第30卷 第10期 计 算 机 工 程 2004年5月
Vol.30 №10 Computer Engineering May 2004
·基金项目论文· 文章编号：1000—3428(2004)10—0057—03 文献标识码：A 中图分类号：TP274

一种基于XML的半结构化数据存储方法

吴共庆，陈恩红
(中国科学技术大学计算机系，合肥 230027)

摘 要：提出了一种基于XML存储半结构化数据的方法，设计并实现了相应存储与解析算法。鉴于OEM模型是一种图状模型，而通常的XML数据模型是树状模型，为此需解决图状模型数据映射为树状模型数据这一关键问题，利用XML元素和属性的语义信息可从语义级别解决该问题。

关键词：半结构化数据；OEM模型；XML；DOM

An Approach of Storing Semi-structured Data with XML

WU Gongqing, CHEN Enhong
(Computer Departmemt,University of Science and Technology of China, Hefei 230027)

【**Abstract**】 The paper proposes an approach to storing semi-structured data based on XML, and develops algorithms for storing and parsing. Because OEM is a data model with graph structure while common data model for XML is tree structured, the key problem is that how to map data with graph structure into data with tree structure. The problem can be solved by utilizing semantic information of element and attribute of XML data.

【**Key words**】 Semi-structured data; OEM model; XML; DOM

互联网目前已经成为全球信息传递与共享的重要资源和途径，Internet上的信息通常被视为是半结构化或无结构的数据，半结构化数据是无预先指定数据模式数据，它有一定的结构，但又不遵循关系型或面向对象模型。半结构化数据的特点是数据结构不规则或不完整，表现为数据不遵循固定模式、结构隐含、模式信息量大、模式变化快、模式和数据统一存放等[1]。随着Internet的发展和异构信息源集成技术以及存储技术的进步，网络中涌现出大量诸如Web页面和XML文档等半结构化数据资源。过去几年对半结构化数据的研究集中在网络上半结构化数据的自动抽取、数据模型、查询语言、模式抽取、半结构化数据库管理系统等方面[2]。一个常见的半结构化数据模型是OEM(Object Exchange Model)模型，OEM是一个自描述嵌套对象模型，可将其视为带标记有向图[3]。

XML(eXtensible Markup Language)是一种元标记语言，用户可利用它来定义自己的标记，这些标记必须根据某些通用的原理来创建，但在标记的意义上，也具有相当的灵活性。自1998年W3C推出XML1.0规范以来，XML技术有了很大发展。XML数据是具有良好结构的半结构化数据，XML的一大优点就是可通过它来创建高度结构化的标记语言。因XML数据具有良好可移植性，XML被广泛应用于数据描述和数据交换等领域。利用DOM(Document Object Model)解析器和SAX(Simple API for XML)解析器等工具，用户可对XML数据进行处理,在XML应用中通过理解XML数据中标记的语义可在语义级处理数据[4]。

OEM模型数据的外存储通常依赖于某种难以解析和处理的自定义格式[5]，为基于XML研究一般半结构化数据，本文提出了一种基于XML存储半结构化数据的方法，设计并实现了相应的存储与解析算法。OEM模型是一种图状数据模型，而通常的XML数据的数据模型(如DOM)是树状模型，为此需解决图状模型数据映射为树状模型数据这一关键问题。尽管OEM模型与XML数据的树状数据模型存在差异，但它们之间有很多相同点[6]。给定OEM模型数据，经过遍历可得一宽度优先生成树，这部分数据可方便地由XML数据来表示，而OEM模型数据中除宽度优先生成树以外的引用边，可定义相关元素和属性在语义级进行描述。

1 基本概念和问题描述

1.1 OEM模型

OEM模型中所有的实体均为对象，每个对象有唯一标识符oid，对象分为原子对象和复合对象，原子对象具有integer、real、string等类型的值，复合对象为对象引用集，其中每个引用表示为(label，oid)，label为一字符串。可将OEM模型数据视为一个带标记有向图,其中图的节点表示对象,图的边表示对象引用。下述定义形式化定义了OEM模型数据，图1给出了一个OEM模型数据示例。

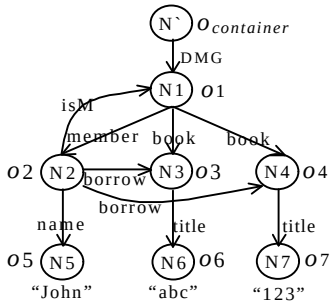

图1 OEM模型数据D

定 义 一个OEM模型数据为字符串集合N_{oid}和N_{label}上五元式$D = (V = V_a \cup V_c, f_{oid}, Val_a, R=\{R_V\}, f_{label})$。其中，$V_a$为原子对象对应的有限节点集，原子对象具有integer、real、string等类型的值，V_c为复合对象对应的有限节点集，$V_a \cap V_c = \varnothing$；$f_{oid}$为V到$N_{oid}$上的单射；$Val_a$为$V_a$到原子对象值域上的单射；$R_V = \{<x, y>|P(x, y) \wedge S(x) \wedge (x, y \in V)\}$，谓词$P(x, y)$表示从x到y的一

基金项目：国家自然科学基金资助项目(60005004)；安徽省自然科学基金资助项目(01042302)
作者简介：吴共庆(1975－)，男，硕士生，主研方向为知识发现；陈恩红，博士、副教授
收稿日期：2003-03-15 E-mail：ahwugq@ah163.com

© 1994-2009 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

条单向通路，谓词S(x)表示x为复合对象；f_{label}为R_V到N_{label}上的映射。

任意e∈ R_V，e表示OEM图中的边。 ∀ o∈ V_c,记复合对象o的值为 $Val_c(o)$， $Val_c(o)=\{y|(x=o)\wedge(<x, y>\in R_V)\}$。 ∀ o ∈ V，若$\{x|<x,o>\in R_V)\}=\varnothing$，则称o为OEM模型数据的根对象。

按上述定义，图1所示的OEM图可表示为D = (V = $V_a\cup V_c$, f_{oid}, Val_a, R={R_V}, f_{label})。其中，V_a＝{o5, o6, o7}，V_c＝{o1, o2, o3,o4, $o_{container}$}，$o_{container}$为D的根对象。f_{oid} ($o_{container}$) = "N0"， f_{oid}(o1) = "N1"， f_{oid}(o2) = "N2"， f_{oid}(o3) = "N3"， f_{oid}(o4) = "N4"，f_{oid} (o5) = "N5"，f_{oid} (o6) = "N6"， f_{oid} (o7) = "N7"。Val_a(o5) = "John" , Val_a (o5) = "abc", Val_a (o5) = "123"。R_V ＝ {<$o_{container}$, o1>, <o1, o2>, <o1, o3>, <o1, o4>,<o2, o1>,<o2, o3>, <o2, o4>, <o2,o5>, <o3, o6>, <o4, o7>}。f_{label}(<$o_{container}$, o1>) = "DMG"，f_{label}(<o1, o2>) = "member"，f_{label}(<o1, o3>) ="book", f_{label}(<o1, o4>) = "book"，f_{label}(<o2, o1>) = "isM", f_{label}(<o2, o3>) = "borrow"，f_{label}(<o2, o4>) = "borrow",f_{label}(<o2,o5>) = "name",f_{label}(<o3,o6>) = "title", f_{label}(<o4, o7>) = "title"。

1.2 XML相关概念

描述XML的基本部件是元素和属性，它们负责确定XML文件的逻辑结构，元素用于表示一个信息对象，而属性用于表示该对象的性质。

XML的一个主要目标就是创建包含自定义标记(元素和属性)的标记词表，这也引发了一个重要问题，即自定义词表中的名字冲突。名字空间提供一种在XML词表中的元素和属性间建立唯一性的机制，解决了名字冲突问题。没有名字空间，在无数开发出来以及Web上使用的XML词表间是不可能保证唯一性的。

W3C DOM混合了树型和对象模型，它提供了一种访问和操纵XML文件的方法，没有DOM，XML就仅仅是一种可以通过多种方法访问的存储系统。通过DOM，XML渐渐成为一种真正统一的、跨平台的、应用无关的数据描述语言。W3C DOM使用双重引用来描述文件，模型中的所有东西都是一个节点和一个命名了的对象类型，有两种方法可以从DOM中取得节点：一种方法是遍历树；另外一种方法是用名字来访问节点。任何支持DOM的对象都必须能够装入XML文件，另外，它还必须能够用适当的DOM规范的属性和方法来提供所有的接口，和它接口的应用程序必须能够通过指定的语法来访问整个XML文件[7]。

1.3 基于XML的半结构化数据存储问题描述

问题描述：给定OEM模型数据，如何建立树状数据模型(DOM)描述图状数据模型(OEM)数据的机制，并实现基于XML的半结构化数据存储与解析还原。

2 存储映射与解析算法

2.1 基于XML的半结构化数据存储映射算法

给定OEM模型数据D，图2所示存储映射算法将D映射为相应的XML文档。该算法要求OEM图具有唯一的根对象，可在原OEM图添加一新根对象和相应的边来达到要求。记OEM图中从根对象开始的宽度优先生成树以外的边为宽度优先引用边。算法1的第(1)、(2)步对所有的宽度优先引用边做上标志。(3)、(4)步写入XML处理指令，其中write(F_{xml}, s)表示将字符串s写入XML文档F_{xml}。(5)、(6)、(8)步写入根对象对应的XML数据以及有关名字空间信息。第(7)步中对根对象的所有子对象调用算法2生成XML数据。

算法1:OEM模型数据到XML文档的存储映射算法

输入:OEM数据D = (V = $V_a\cup V_c$, f_{oid}, V_{ala}, R={R_V}, f_{label})

输出:XML文档F_{xml}

(1) for each <x, y>∈R_V do flag[x, y]:=false;
(2) 从OEM图根对象r开始宽度优先遍历;
for each <x, y>∈R_V do
if <x,y>为宽度优先引用边 then flag[x, y]:=true;
(3) s:="<?xml version="+""+"1.0" +""
+"encoding="+""+"utf-8"+""+"?>";
(4) write(F_{xml}, s);
(5) s:="<Ocontainer xmlns:RS="
+""+"URL/RSSchema#"+""+">";
(6) write(F_{xml}, s);
(7) for each x in Val_c(r) do genXML(r, x, F_{xml});
(8) s:="</Ocontainer>"; write(F_{xml}, s);

图2 OEM模型数据到XML文档的存储映射算法

图3所示算法2实现了基于深度优先遍历生成XML数据过程。算法的(1)~(4)步生成原子对象的XML数据。(5)~(7)步生成复合对象的XML数据，其中(6)、(7)、(15)、(16)步生成复合对象的XML元素和属性，(8)~(14)步生成复合对象中嵌套子对象的XML数据，(9)~(12)步生成宽度生成引用边的XML数据，第(13)步递归调用genXML生成嵌套子对象的XML数据。

算法2: genXML(p, c, F_{xml}) //深度优先遍历生成XML数据

输入: OEM图中对象p, c

输出: XML数据

```
(1)  if c∈ Va then{ //c为原子对象
(2)   s:='<'+flabel(p, c)+" RS:OID="+""+foid(c)+""+'>'+Val_a (c)+
'</'+flabel(p, c)+'>';
(3)   write(Fxml, s);
(4)  }
(5)  else{ //c为复合对象
(6)   s:='<'+ flabel(p, c)+ " RS:OID="+""+foid(c)+""+'>';
(7)   write(Fxml, s);
(8)   for each cc in Val_c(c) do{
(9)    if flag[c, cc] then{ //< c, cc >为引用边, 生成引用边的
      //XML数据
(10)    s:="<RS:OREF RS:lname="+""+flabel(c, cc)+""+" RS:
OID="+ foid(cc)+"/>";
(11)    write(Fxml, s);
(12)   }
(13)   else genXML(c, cc, Fxml); //生成嵌套对象的XML数据
(14)  }
(15)  s:="</"+ flabel(p, c)+'>';
(16)  write(Fxml, s);
(17) }
```

图3 深度优先遍历生成XML数据算法genXML

根据算法1，图1中OEM图映射为图4所示的XML文档，该文档采用了图5中的RSSchema.xml定义的元素和属性描述宽度优先引用边，RSSchema.xml中的OID属性用于描述对象标识符，OREF元素用于描述对象引用，OREF的lname属性用于描述该引用边上的标记，OREF的OID属性用于描述所引用对象的标识符。算法1生成的XML文档不仅体现出原OEM图宽度优先生成树的层次树结构，而且语义级无损表示了原图的结构和内容。

© 1994-2009 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

```
<?xml version="1.0" encoding="utf-8"?>
<Ocontainer xmlns:RS="URL/RSSchema#">
<DMG RS:OID="N1">
      <member RS:OID="N2">
          <OREF RS:lname="isM"   RS:OID="N1"/>
          <OREF RS:lname="borrow" RS:OID="N3"/>
          <OREF RS:lname="borrow" RS:OID="N4"/>
          <name RS:OID="N5">John</name>
      </member>
      <book RS:OID="N3">
          <title RS:OID="N6">abc</title>
      </book>
      <book RS:OID="N4">
          <title RS:OID="N7">123</title>
      </book>
    </DMG>
</Ocontainer>
```

图4 OEM模型数据D对应的XML文档

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:attribute name="OID" type="xs:ID"/>
  <xs:element name="OREF" type="xs:IDREF">
    <xs:attribute ref="OID"/>
    <xs:attribute name="lname" type="xs:string"/>
   </xs:element>
</xs:schema>
```

图5 RSSchema.xml

2.2 基于XML的半结构化数据存储解析算法

算法3**解析**XML**文档为**OEM**图，第**(1)**步利用**DOM**解析器将**XML**文档解析为一棵**DOM**树，并定位该树的文档根节点。第**(2)**步创建**OEM**图的根对象。**(6)~(19)**步生成原**OEM**图的宽度优先生成树,其中第**(12)**步生成一原子对象，此处为描述算法方便起见将所有的原子对象均以字符串方式处理，第**(14)**步生成一复合对象。**(20)~(25)**步生成原**OEM**图的宽度优先引用边。算法使用了一个**Npair**查找表以创建**OEM**图中的边，其中查找表的每一条目**(a,b)**，**a**表示**DOM**树中的节点，**b**表示**OEM**图中根据**a**创建的对象。

算法3: parseXML2OEM //**解析**XML**文档为**OEM**图

输入: **存储**OEM**图的**XML**文档** F_{xml}

输出: OEM**模型数据** $D=(V=V_a \cup V_c, f_{oid}, Val_a, R=\{R_V\}, f_{label})$

(1) **解析** F_{xml}**得一**DOM**树**, r**为**DOM**树的文档根节点**;

(2) **创建节点**r', **并置** $f_{oid}(r')$**为节点**r**的**RS:OID**属性值**; $V_c:=V_c \cup \{r'\}$;

(3) **将节点对**(r, r')**放入**Npair**查找表**;

(4) **初始化队列**Q、Q';

(5) for each **节点**r**的所有无**RS:OREF**属性的元素子节点**c do c**进队列**Q;

(6) while **队列**Q**不空** do{ //**生成宽度优先生成树**

(7) **队首节点**f**出队列**Q;

(8) **创建节点**f', **并置** $f_{oid}(f')$**为节点**f**的**RS:OID**属性值**;

(9) **将节点对**(f, f')**放入**Npair**查找表**;

(10) p**为节点**f**的父节点**, **从**Npair**查找表中查找节点**p**对应的**OEM**图节点**p;

(11) VR := VR ∪ {<p', f'>}; **并置** $f_{label}(p', f')$**为节点**f**的元素标记**;

(12) if **节点**f**存在文本子节点**t then{ $V_a:=V_a \cup \{f'\}$; **并置** $Val_a(f')$**为节点**t**的值**;}

(13) else {

(14) $V_c:=V_c \cup \{f'\}$;

(15) for each **节点**f**的所有元素子节点**c do

(16) if **节点**c**无**RS:OREF**属性** then c**进队列**Q; // c**用于生成** // **宽度优先生成树**

(17) else **则节点**c**进队列**Q'; // c**用于生成宽度优先引用边**

(18) }

(19) }

(20) while **队列**Q'**不空** do{ //**生成宽度优先引用边**

(21) **队首节点**f**出队列**Q';

(22) p**为**f**的父节点，从**Npair**查找表中查找节点**p**对应的**OEM**图节点**p';

(23) **查找**OEM**图**D**中对象标识符为**f**的**RS:OID**属性值的节点**f';

(24) VR := VR ∪ {<p', f'>}; **并置** $f_{label}(p', f')$ **为节点**f**的**RS:lname**属性值**;

(25) }

2.3 算法分析

令‖A‖**表示集合**A**的基。给定半结构化数据**D=(V=$V_a \cup V_c$, f_{oid}, Val_a, R=$\{R_V\}$,f_{label})，设**n=‖V‖，e=‖R_V‖，以邻接表表示**OEM**图，则图的遍历操作时间复杂度为**O(n+e)。

算法2**实质上是一个遍历宽度优先生成树的过程，遍历树的时间复杂度为**O(n)，**算法**2**的第**(9)**步计算**flag[c,cc]**的时间复杂度为**$O(n^2)$，**因此算法**2**总时间复杂度为**$O(n^2e)$。

算法1**的第**(1)、(2)**步的时间复杂度均为**$O(n^2e)$，**第**(7)**步的时间复杂度就是算法**2**的时间复杂度即**$O(n^2e)$，**故算法**1**的时间复杂度为**$O(n^2e)$。

算法3**的第**(10)**步在**Npair**查找表中查找操作时间复杂度均为**O(n)，**所以生成宽度优先生成树的时间复杂度为**$O(n^2)$。**第**(22)、(23)**步的查找操作时间复杂度为**O(n)，**所以生成宽度优先引用边的时间复杂度为**O(ne)。**由上述分析可知算法**3**的时间复杂度为**$O(n^2+ne)$。

3 总结

为基于XML**研究一般半结构化数据，本文提出了基于**XML**的半结构化数据存储问题，并给出了一种解决该问题的方法，实现了相应的存储映射和解析算法。该问题的关键是如何利用树状模型数据表达图状模型数据，利用**XML**元素和属性的语义，从语义级解决了树状数据模型描述图状数据的问题。下一步研究工作包括基于**XML**的半结构化数据查询与更新问题的研究，以及基于本文提出的方法对半结构化数据模式发现与变化检测等问题做进一步的研究。**

参考文献

1 Abiteboul S. Querying Semistructured Data. In Proceedings of the International Conference on Database Theory, Delphi, Greece,1997-01:1-18

2 Asai T, Abe K, Kawasoe S, et al. Efficient Substructure Discovery from Large Semi-structured Data. Proc. Second SIAM International Conference on Data Mining 2002 (SDM'02), SIAM, 2002:158-174

3 Papakonstantinou Y ,Molina H G, Widom J. Object Exchange Across Heterogeneous Information Sources. In IEEE International Conference 1 on Data Engineering, 1995-03:251-260

4 Bray T, Paoli J,McQueen C S(Editors). Extensible Markup Language (XML) 1.0(Second Edition).W3C Recommendation, http://www. 1 w3.org/TR/2000/REC-xml-20001006,2000-10

5 Goldman R, Chawathe S, Crespo A,et al. A Standard Textual Interchange Format for the Object Exchange Model (OEM). Technical Report, 1996-10

6 Suciu D. Semistructured Data and XML. In Proceedings of the 5th Int. Conf. of Foundations of Data Organization (FODO'98), 1998-11

7 Morrison M. XML Unleashed. Sams Publishing, 2000

© 1994-2009 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net