第 32 卷 第 19 期 计 算 机 工 程 2006 年 10 月
Vol.32 № 19 Computer Engineering October 2006
·软件技术与数据库· 文章编号：1000—3428(2006)19—0091—03 文献标识码：A 中图分类号：TP311

一种用于存储与查询半结构化数据的新方法

叶飞跃，蒙德龙，员红娟
(上海大学计算机工程与科学学院，上海 200072)

摘 要：由于半结构化数据缺乏模式信息，因而半结构化数据的存储与查询将是一个十分重要且具有挑战性的研究课题。利用关系数据库存储半结构化数据可以重用数据库的查询优化器和事务处理机制，能够保证半结构化数据的一致性和完整性。该文提出一种实现半结构化数据存储与查询的新方法，该方法使用关系数据库系统来实现半结构化数据的存储与查询。给出了把基于半结构化数据的查询重写为基于关系的查询的算法，同时介绍一个可视化查询程序。
关键词：半结构化数据；查询重写；OEM

New Approach for Storing and Querying Semistructured Data

YE Feiyue, MENG Delong, YUAN Hongjuan
(School of Computer Engineering and Science, Shanghai University, Shanghai 200072)

【Abstract】 Storing and querying semistructured data will be a very important and challenging research because it lacks schema information. Using RDBMS to store semistructured data can reuse database's query optimizer and transaction manager, which can ensure semistructured data's consistency and integrality. This paper presents a new approach for storing and querying semistructured data. An algorithm is given to rewrite from semistructured data-based query to relation-based one, and a visual query program is introduced.
【Key words】 Semistructured data; Query rewriting; OEM

半结构化数据是指那些结构隐含或无规则、不严谨的自我描述型数据[1]。这样的数据介于严格结构化数据(如关系数据库和对象数据库中的数据)和完全无结构的数据(如声音、图像文件)之间[2]。半结构化数据的来源主要是：互联网和数据集成。传统的数据存储、处理、查询技术不能直接应用于这种数据类型，本文提出了一种基于关系的半结构化数据的存储方法，并在此基础上给出了一种半结构化数据的查询语言，且给出了把这种查询语言重写为 SQL 语句的算法。为了克服直接输入半结构化数据查询语句容易出错的缺点，我们还编写了一个可视化查询程序，用于半结构化数据的可视化查询。

半结构化数据的模式描述方法很多，有基于图的描述形式和基于逻辑的描述形式[2]。在这里采用的是比较有代表性的基于图的 OEM(Object Exchange Model)模型。

1 基本概念

1.1 OEM 模型

OEM 是斯坦福大学(Standford University)Papakonstatinon 等人提出的用来描述半结构化数据的数据模型[3]。OEM 模型的主要特征是自描述性。该模型由表示对象的结点和带标签(label)的有向边构成。每个 OEM 对象可以用一个 4 元组来表示：(oid,label,type,value)。其中 oid 是对象标识。label 是对象的标签描述，表示对象之间的关系。type 是对象类型，对象类型有两类：原子对象和复杂对象。原子对象是不可再分的基本类型，如 int，string，real 等。复杂对象是对象引用的集合，每一个对象引用指向另一个对象。不失一般性，借用文献[4]中的 OEM 模型图来举例说明文中用到的相关概念及算法。OEM 模型可以用一个带根有向图 G(r,V,E)来表示，其中 r 表示根结点；V 表示对象集；E 是有向边的集合，边上的标签表示对象之间的关系，记作<O_i,label,O_j>，如<1,Player,14>,<0,Club,20>等。若图 G 中存在<O_1,l_1,O_2>，<O_2,l_2,O_3>,…,<O_n,l_n,O_1>，则称 OEM 模型图 G 中存在环路。

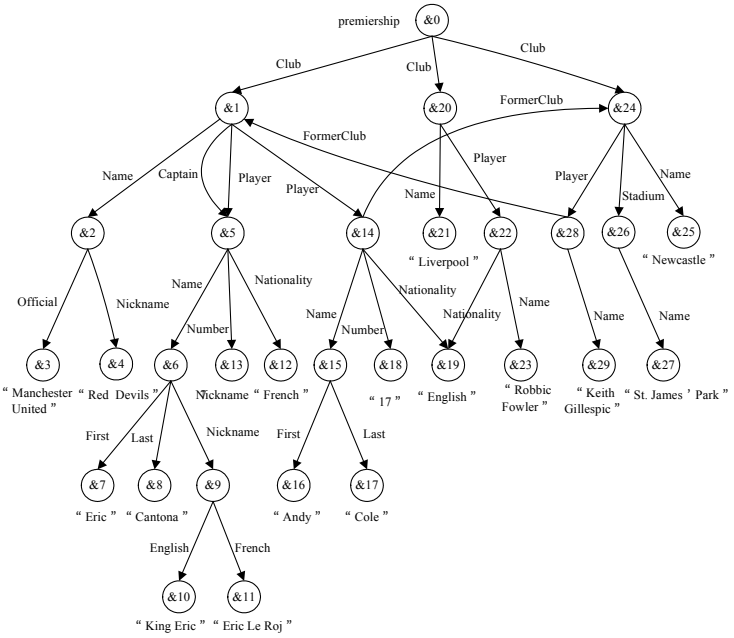

图 1 OEM 模型

作者简介：叶飞跃(1959－)，男，教授，主研方向：数据库；蒙德龙、员红娟，硕士
收稿日期：2005-12-24 **E-mail：**mengdl2004@163.com

1.2 标签路径、数据路径和路径实例

标签路径 lp 是以符号‘．’分隔开的标签序列，记作 lp=l_1．l_2…l_n,n 是 lp 的长度。如 lpl=Club．Name．Official 是一条长度为 3 的标签路径。

数据路径 dp 是以符号‘．’分隔开的对象和标签交替出现的序列，记作 dp=O_0．l_1．O_1．l_2…l_n,O_n，n 是 dp 的长度。例如：dp1=0．Club．1．Name．2．Official．3。

称数据路径 dp=O_0．l_1．O_1．l_2…l_n,O_n 为对应于标签路径 lp=l_1．l_2…l_n 的路径实例。对于同一标签路径可能存在多个路径实例。图 1 中有 4 条数据路径是标签路径 lp2=Club．Player．Name 的路径实例。它们分别是：

dp1=0．Club．1．Player．5．Name．6

dp2=0．Club．1．Player．14．Name．15

dp3=0．Club．20．Player．22．Name．23

dp4=0．Club．24．Player．28．Name．29

1.3 同类对象

所谓同类对象是指那些对应于同一标签路径的所有路径实例中的最后一个对象。由同类对象组成的集合称为同类对象集。例如对应于标签路径 lp2 的同类对象集是{6,15,23,29}。

2 半结构化数据的存储

目前提出并实现了半结构化数据的存储技术主要有：文本文件方式，RDB 方式及 OODB 等方式[5]。文本文件存储方式的缺点是存储难度较大，不利于数据的检索和管理。对于 OODB 方式，如果在事先不知道数据的类型信息时，数据加载的代价可能是很高的；如果类型发生了变化，将导致代价极高的模式更新。而半结构化数据的数据类型和模式是不固定的，它随着数据的更新而发生变化。

利用 RDBMS 来存储半结构化数据具有如下的优点：当前的关系数据库技术已十分成熟，商用的关系数据库都具有高性能的查询引擎、良好的可扩展性、安全性和健壮性。利用关系数据库存储半结构化数据可以重用数据库的查询优化器和事务处理机制，能够保证半结构化数据的一致性和完整性。但是，由于数据模型上的差异，因此利用关系数据库来存储半结构化数据也给数据库技术带来了许多新的挑战。

半结构化数据模型本质上是基于带根有向图[6]，用关系表来存储半结构化数据必须保证不破坏或丢失原来数据的结构信息和数据信息，因此，除了存储对象值之外，还要存储对象之间的关系。本文提出一种基于边的半结构化数据存储方法，使用 3 个关系表存储半结构化数据，半结构化数据的结构信息蕴涵在关系表中。表结构及其意义表示如下(下划线表示主键)：

root(rOID,chOID,label)。这个表存储 OEM 的根结点 ID、根结点的子结点 ID 以及根对象名。

edges(pOID,chOID,label,flag)。这个表存储 OEM 中的边，flag 字段用于表示 chOID 的对象类型，如果是原子对象则用 leaf 表示，否则用 ref 表示。

values(OID,type,Val_int,Val_string,Val_float,…)。这个表存储原子对象的类型和对象值。

图 1 所示的半结构化数据用关系数据库存储表示，如表 1~表 3 所示。

表 1 root 表

rOID	chOID	label
0	1	premiership
0	20	premiership
0	24	premiership

表 2 edges 表

pOID	chOID	label	flag
0	1	Club	ref
0	20	Club	ref
…	…	…	…

表 3 values 表

OID	type	Val_int	Val_string	Val_float
3	string		Manchexter United	
4	string		Red Devils	
…	…	…	…	…

这种存储方法只用 3 个关系表就把基于图结构的半结构化数据信息和结构信息全部存储下来，对于半结构化数据的标签路径、数据路径都蕴涵在关系表中，对半结构化数据的查询就可以转化为对关系表的查询。

3 查询

半结构化数据的特点是数据的结构不规则或不完整，其模型都基于带根有向图，因此，半结构化数据的查询过程本质上可以看作是从根结点开始对图的搜索过程[6]。由于我们采用的是基于关系的存储方式，因此对图的搜索过程要转化为对关系表的查询过程，转化过程由后面的查询重写来实现。

3.1 查询语言

在用于异构数据源集成的丛模型(PM)系统中，使用的查询语言可以看作是 Lorel 的一个子集，查询类似于 SQL 语句的结构，即

```
SELECT select-list
FROM from-list
WHERE condition
```

例如，考虑查找运动员“Robbic Fowler”的国籍的查询，查询可以表示如下：

```
Q1：SELECT x.nationality
FROM premlership.club.player x
WHERE x.name=“Robbic Fowler”
```

上面的查询语句不能直接用于 RDBMS，必须把它改写为 RDBMS 能够识别的 SQL 语句。

3.2 查询重写

查询重写就是要把基于路径的半结构化数据查询改写为基于关系表的 SQL 查询。结合前面介绍的半结构化数据的存储方法，提出使用 IN 子句的查询重写方法，其基本思想是利用 IN 子句对标签路径的同类对象集进行集合查找和集合交操作。算法由下面 3 部分组成：(1)查询得到满足 Q1 中 from 子句的同类对象集；(2)在满足第 1 步的同类对象集中查询满足 where 子句的对象集；(3)在满足第 2 步的对象集中查询满足整个查询的结果集。算法如下：

```
输入：半结构化数据查询语句 Q
输出：基于关系的 SQL 查询语句
s1=“select DISTINCT rOID from Root”;
lp=getFromLp(Q);//取 from 子句中的标签路径
for label∈lp
s1=“select chOID from edges where pOID IN (”
+s1+“) AND (label=”+label+“)”;
rp=getWhereLp(Q);//取 where 子句中的标签路径
s2=“select OID from values where val_string”+whereCondition;
for label∈rp
s2=“select pOID from edges where chOID IN (”+s2+“)”
+“AND (label=”+label+“)”;
s3=“select DISTINCT pOID from edges where pOID IN (”
+s1+“) AND pOID IN (”+s2+“)”;//查询交集
rp=getSelectLp(Q);//取 select 子句中的标签路径
```

```
s4="select pOID from edges where pOID IN ("+s3+")";
for label∈rp
s4="select chOID from edges where pOID IN ("
+s4+") AND (label="+label+")";
sql="select Val_string from values where OID IN"+s4;
//在对象值表上查询
return sql
```

例：上面的 Q1 查询使用重写算法改写后，得到如下所示的 SQL 查询，其中左边一列是语句行号，右边的语句是改写后得到的基于关系的 SQL 查询。

```
1   select Val_string from values
2   where OID IN
3   (select chOID from edges
4     where pOID IN
5     (select DISTINCT pOID from edges
6           where pOID IN
7           (select chOID from edges
8                 where pOID IN
9                 (select chOID from edges
10                where pOID IN
11                      (select DISTINCT rOID from root)
12                      AND (label='Club')
13                )
14                AND (label='Player')
15          )
16          AND pOID IN
17          (select pOID from edges
18                where chOID IN
19                (select OID from values
20                      where val_string='Robbic Fowler'
21                )
22          )
23    )
24    AND (label='Nationality')
25  )
```

说明：

(1)7~15 行查询得到满足查询 Q1 中 from 子句的同类对象集，即{5,14,22,28}；

(2)17~22 行查询得到满足 Q1 中 where 条件子句 x.name="Robbic Powler"的对象，即{22}；

(3)第 5 行和 16 行实现(1)和(2)两步的交集，得到查询 Q1 中满足条件的对象集，即{5,14,22,28}∩{22}={22}；

(4)3~25 行查询得到满足第(3)步的 x.Nationality 的对象集{19}；

(5)最后通过第 1、2 两行在关系表 values 中查询得到满足条件的运动员的国籍"English"。

3.3 可视化查询界面

半结构化数据没有固定的模式，随着数据的变化模式也会发生变化，这使得书写半结构化数据的查询语句非常不方便，而且容易出错。为了克服这个缺点，我们编写了一个可视化查询程序，程序界面见图 2。

在可视化查询界面上，左边的面板显示半结构化数据的模式图，用鼠标点击模式图中的标签名，对应的标签名就会出现在右边的 label 文本框中，然后再点击 Add 按钮，就可以把相应的标签添加到对应的查询子句中去。重复以上过程，直到所需的标签路径和查询条件全部生成为止。最后点击界面下方的 Create 按钮，程序就会按要求生成一条完整的查询语句。如果要执行这个查询，则只要点一下 Query 按钮即可。可见使用这个可视化查询界面，用户几乎不用输入什么内容，只要用鼠标在这个界面上点几下就可以生成一条半结构化数据查询语句，非常方便。

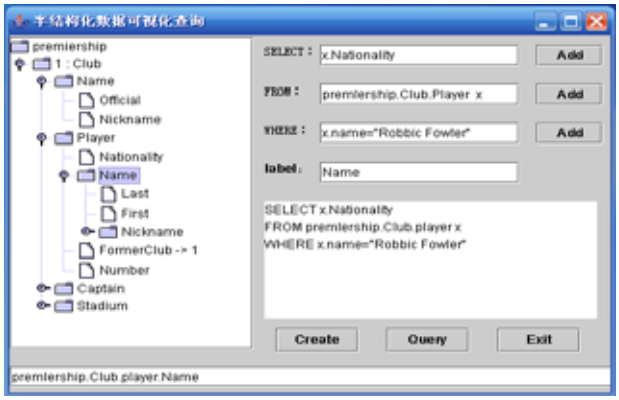

图 2 可视化查询界面

4 性能分析

查询重写过程就是把对路径的搜索改写为对关系表的查询。查询的路径长度决定重写后 SQL 查询的嵌套层数，路径越长，重写后得到的查询语句的嵌套层数就越多。在 DBMS 中，嵌套查询语句的执行是从最内层 select 查询开始的，并逐层向外执行。查询主要是在表 egdes 上进行，为简化分析，假定每一层 select 查询的时间开销为 t，则嵌套 m 层的查询的时间开销为 mt。设有如下查询语句：

```
SELECT x.lrp
FROM lp x
WHERE x.crp
```

令 lrp 的路径长度为 n1，lp 的路径长度为 n2，crp 的路径长度为 n3。则重写后的查询语句的 select 嵌套层数为

(1+n2)+(1+n3)+1+n1=n1+n2+n3+3

对于上面的查询 Q1，n1=1，n2=3，n3=1，重写后的查询语句的 select 嵌套层数等于 8，查询开销为 8t。可见，查询的时间开销与查询语句中的 select、from 及 where 子句中的标签路径的长度之和成正比。通过在关系表中建立索引，可以减少查询中的 t 值，从而加快查询速度。

5 结束语

半结构化数据的存储与查询及相关技术是异构数据源集成和网络资源共享的一个研究重点。本文提出的半结构化数据的存储与查询方法已经应用到我们的丛结构模型系统中，实践证明该方法是可行和有效的。在下一步工作中，我们的工作重点将是半结构化数据的查询优化。

参考文献

1 Abiteboul S. Querying Semi-structured Data[C]. Proc. of ICDT Delphi, Greece, 1997.

2 王 静, 孟小峰. 半结构化数据的模式研究综述[J]. 计算机科学, 2001, 28(2): 6-10.

3 Papakonstanntinou Y, Garica M H, Widom J. Object Exchange Across Heterogeneous Information Sources[C]. Proc. of the IEEE ICDF. IEEE Computer Society Press, 1995: 251-260.

4 Svetlozar N, Jeffrey U, Janet W. Representative Objects: Concise Representations of Semistructured, Hierarchical Data[C]. Proc.of ICDE, 1997: 79-90.

5 聂培尧, 李战怀, 胡正国. 一种基于 XML 的半结构数据的 ORDB 存储方法[J]. 计算机工程与应用, 2003, 39(14): 190-193,199.

6 陈 滢, 王能斌. 半结构化数据查询的处理和优化[J]. 软件学报, 1999, 10(8): 883-890.