第35卷 第12期 计 算 机 工 程 2009年6月
Vol.35 No.12 Computer Engineering June 2009
·软件技术与数据库· 文章编号：1000—3428(2009)12—0056—03 文献标识码：A 中图分类号：TP311.13

基于函数依赖的XML冗余检测算法

沈 刚，罗 军
(重庆大学计算机学院，重庆 400030)

摘 要：XML保持语义下的冗余检测问题对于防止XML文档的更新异常和减少XML冗余信息的存储具有很大意义，是XML规范化理论的关键问题之一。对XML schema、树元组、XML键、XML范式等进行研究，定义基于schema的XML函数依赖形式化模型，并基于该定义模型设计一种新的XAP算法，可有效发现XML文档中的函数依赖和冗余，并对算法的复杂性进行分析。
关键词：XML模式；冗余检测；函数依赖；属性分解

XML Redundancy Detection Algorithm Based on Function Dependency

SHEN Gang, LUO Jun
(College of Computer Science, Chongqing University, Chongqing 400030)

【Abstract】 The problem of XML redundancy detection with semantics is very meaningful to prevent update anomalies and the key problem of XML normalization theory. In this paper, the definition of XML schema and XML function dependency etc. are given and the new XML function dependency and normal form model are proposed and based on this model a new XAP algorithm is designed to discover function dependency and detect redundancy. It also analyzes its complexity.
【Key words】 XML schema; redundancy detection; function dependency; attribute partition

1 概述

随着XML的逐渐流行，XML文档中的冗余也随之成为人们越来越关心的一个问题。冗余信息不仅浪费了存储空间，增加了数据传输和维护操作的成本，而且导致潜在的更新异常，引起数据库的不一致。

定义XML数据冗余时的问题形象地体现在图1所示的XML示例文档中。这个文档维护了各个书店售书的相关信息，包括一个按所在省(province)分类的书籍仓库(warehouse)信息。每一个书店(store)记录了店名(name)的信息以及正在柜上出售的数据信息，每一本书(book)则记录了它的ISBN号、作者(au)、书名(title)以及价格(price)等信息。这个例子有2个很显然的语义上的约束，如果2本书的ISBN号相同，那么他们的书名和作者一定相同；如果2本书的书名和作者相同，那么它们的ISBN号一定相同。这2个约束都会使得文档中的某些信息成为冗余。与关系模式里类似冗余的特征不同，XML冗余的一个重要区别是引入了集合的概念：冗余的往往是单一对象的集合，而不仅仅是单一的元素，例如一本书的作者就可能有多个。第3个约束稍隐蔽一些，同一家书店的2本书的ISBN号相同，那么它们的价钱相同。例如，价钱为59.9元的…269号图书，在希望(Xiwang)连锁店有冗余存在，2本书在不同的省。这样的冗余的特殊之处在于，约束所涉及到的元素不全是基于book节点的子孙节点，而是跨层存在的。将找到的典型约束归纳如下：

约束1 如果2本书的ISBN号相同，那么它们的书名相同。

约束2 如果同一家书店(书店名相同)的2本书的ISBN号相同，那么它们的价格相同。

约束3 如果2本书的ISBN号相同，那么它们的作者相同。

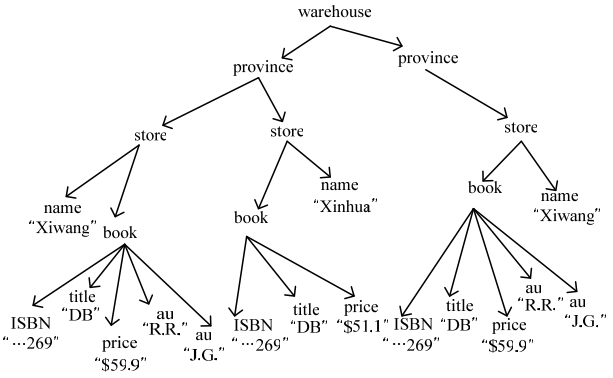

图1 XML示例文档

2 相关工作

虽然有关XML数据模式设计的研究已经取得了一些初步成果，但还没有形成统一的规范和完整的理论体系。文献[1]采用基于树元组的方法并且第1个形式化地定义了XML函数依赖和范式。文献[2]利用文献[3]中XML键的概念，采用基于路径的方法从而建立XML函数依赖的概念。文献[4]通过定义XML函数依赖和推理来指导基于DTD的XML文档设计，但还不够完善，比如对于集合(set)元素的处理。

作者简介：沈 刚(1984－)，男，硕士研究生，主研方向：数据库，XML规范化；罗 军，副教授
收稿日期：2008-12-25 **E-mail**：gangshen001@163.com

现有的算法和策略大都采用仿照关系数据库的思路[5]，在 XML 领域提出相似的键、函数依赖、范式等理论方法，从指导 XML 文档设计的角度来进行 XML 规范化研究。这些理论虽对指导 XML 规范化设计有很大意义，但对于已存在的 XML 文档中的冗余则几乎没有涉及。

3 基本定义

本文先给出了 schema 的形式化定义，但 schema 语法不足以描述 XML 文档的所有语义信息，所以需要引入 XML 函数依赖的概念，以进一步扩充 XML 文档语义表达能力。在关系数据库中，一个数据依赖是关于属性值之间的相关关系的一个命题，它规定了任何一个数据库的合法状态所必需满足的一个语义完整性的约束条件，即从语义上决定了冗余的存在。而函数依赖是最常见、最重要的一种数据依赖，它是关系数据库设计理论的主要内容之一，是范式研究的基础。同样的语义约束，在 XML 文档中也是同样存在的。

在下面的定义中，约定关键字 Rcd 用来标记比较复杂的结构型元素，关键字 SetOf 用来标记集合型元素，集合型元素并不一定是结构型元素，如图 1 中的 author 是集合型元素，但不是结构型元素。

定义 1 XML schema 是一个三元组(E, T, r)。其中，E 是有限的元素名称的集合；T 是元素类型的集合，$\forall e \in E$ 都有唯一的一个 $t \in T$ 与之相对应，t 满足正则表达式 t::=str|int|float|SetOf t|Rcd[$e_1 : t_1 \cdots e_n : t_n$]；r 是根节点元素，即 $r \in E$，且 r 一定不是集合型元素。

图 1 的 XML 文档对应的 schema 如下：

```
Warehouse: Rcd
  Province: SetOf Rcd
      Store: SetOf Rcd
          Name: str
          Book: SetOf Rcd
              ISBN: str
              Author: SetOf str
              Title: str
              Price: str
```

定义 2 树模型四元组 $T(N,P,V,n_r)$。其中，N 是数据节点的集合，$\forall n \in N$，在 T 中都有唯一的标记 e 与之相对应；$n_r \in N$ 是根节点；P 是父节点与子节点之间的边的集合；V 是节点值的集合。

定义 3 通用树元组：从原 XML 文档树中抽取部分节点组成，且任意 2 个节点不允许来自同一个 schema 元素，但 set 元素要全部列出。

定义 4 元组类：树 T 中一类通用树元组的集合。通常情况下表示聚合在某条路径的所有树元组的集合。

定义 5 XML 函数依赖：三元组 C_p, LHS, RHS 形式为 $LHS \rightarrow RHS$ w.r.t. C_p

其中，C_p 表示一个元组类；LHS, RHS 是树中的路径。表示的意义为，对任意 2 个 C_p 中的树元组，同样的 LHS 决定并约束 RHS。

定义 6 XML 键：二元组 C_p, LHS，要求树 T 满足函数依赖 C_p, LHS, ./@key。例如下面的这个键< C_{state}, {./name}>。

定义 7 XML 数据冗余：XML 文档数包含冗余当且仅当树 T 满足函数依赖 C_p, LHS, RHS，但是不满足 XML 键 C_p, LHS。

4 冗余检测的XAP算法

4.1 XML数据的表示

基于第 3 节中的 XML 函数依赖的概念，可以找到一个检测函数依赖的方法：先把原始的 XML 数据树表示成二维表的模式，然后结合已有的关系数据库理论里的函数依赖检测算法。后面的算法将分别针对表内、表间以及集合元素中的函数依赖进行检测。

4.2 表内的函数依赖检测

算法 1 表内函数依赖检测算法 Intra-RFDD

输入：关系模式 R 以及其中的字段 $a_1, a_2, \cdots, a_n$

输出：关系模式中所蕴含的函数依赖集合 FDs

算法过程：

(1)初始化：计算生成各字段的投影 $\prod_{a_1}$, $\prod_{a_2}, \cdots, \prod_{a_n}$，$FDs=\phi$，自定义队列 Queue $Q=\phi$，属性集 $A=\phi$。

(2)对于表内的每一字段，逐个搜索可能的函数依赖关系：

```
for i=1,2,…,n: Q.enqueue({a_i});
while Q≠ϕ   A=Q.dequeue();
If ∏_{a_i} 不存在 L_s = generateLHS(A, FDs);
Foreach A_1 ∈ L_s, r=A- A_1;
If ∏ A_l== ∏ A, FDs.add(A_l→r);
Function generateLHS(A, FDs)
L_s=0; Foreach   a ∈ A, 令 A_l = A – a;
Foreach L→r ∈ FDs, If a==r and A_l→r ∈ L, continue;
Else if A ∈ L, continue;
Else L_s.add(A_l)          //找到 LHS
Return L_s
```

上述算法以图 2 中的表 R_book 为例：

$\prod_{price}$ for c_{book} = { {t6,t20}, {t13} }

$\prod_{@key}$ for c_{book} = { {t6}, {t20}, {t13} }

$\prod_{price,@key}$ for c_{book} ={{t6},{t20}, {t13} }

那么由函数依赖的定义：$LHS \rightarrow RHS$ w.r.t. c_p is satisfied iff: $\prod_{LHS \cup RHS} = \prod_{LHS}$，可以找到下面这个函数依赖关系：{./ISBN} → ./title, w.r.t. $c_{/warehouse/province/store/book}$，即上文提到的约束 1：如果 2 本书的 ISBN 号相同，那么它们的书名相同。

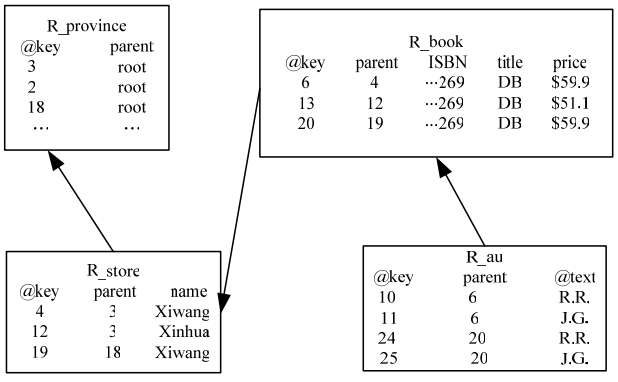

图 2 XML 数据的关系表示

4.3 表间的函数依赖检测

算法 2 表间函数依赖检测算法 Inter-RFDD

输入：父关系 P'和子关系 P，2 个关系的表内函数依赖集合 FDs(由算法 1 得出)

输出：关系模式中所蕴含的函数依赖集 FDs

算法过程：

(1)初始化：$FDs=\phi$, Queue $Q=\phi$, $PTs=\phi$。

(2)表间函数依赖存在的必要条件：

对于父表 P'的任意属性 X 及 $LHS \cup X \rightarrow RHS$ w.r.t. C_p，必须满足 $LHS \cup \{./parent\} \rightarrow RHS$ w.r.t. C_p；

对于子表 P 的任意属性 X 及 $LHS \cup X \rightarrow RHS$ w.r.t. C_p，$LHS \rightarrow RHS$ w.r.t. C_p 必须不能满足。

(3)对于满足上述条件的父子表，从底向上地搜索可能存在的函数依赖关系：

For i=1,2,…,n: Q.enqueue($\{a_i\}$);
Foreach pt ∈ PTs:
If pt' ≠ NULL resultPTs.add(pt');
While Q ≠ 0; A=Q.dequeue();
If $\prod$ A 不存在 L_s=generateLSH2(A, curFDs);
Foreach $A_L \in L_s$: If $\prod A_l == \prod A$,
curFDs.add($A_l \rightarrow A - A_l$), continue;
If pt ≠ NULL, resultPTs.add(pt);
Return resultPTs;

上述算法用来检测表 R_book 和表 R_store 的表间函数依赖，就是下面的过程：

$\prod_{ISBN}$ = { {t6, t13, t20}}

$\prod_{price}$ = { {t6, t20}, {t13}}

PT = { t4 ≠ t12, t19 ≠ t12}

$\prod_{name}$ = { {t4, t19}, {t12} }

{../name, ./ISBN} → ./price w.r.t. c_{book}

于是可以找到下面这个依赖关系：

{../name, ./ISBN} → ./price, w.r.t. $c_{/warehouse/province/store/book}$

即上文提到的约束 2：如果同一家书店(书店名相同)的 2 本书的 ISBN 号相同，那么它们的价格相同。

4.4 set元素的处理

对于涉及 set 元素的函数依赖检测，可以采取分解转化的方法，以图 2 中的表 R_au 结合表 R_book 为例：

$\prod_{au}$ for R_book = { {t6, t13, t20}}

$\prod_{@text}$ for R_au = { {t10, t24}, {t11, t25} }

$\prod_{@text}$ for R_book = { {t6, t20}, {t6, t20} }

$\prod_{au}$ for R_book = { {t6, t20}, {t13} }

5 算法复杂度分析

简要分析一下上述 2 个算法的时间复杂度。在算法 1 中，假设有 n 个关系模式，每张表里有 k 个属性，那么该算法最坏情况下的时间复杂度为 $O(n \cdot k \cdot 2^k)$；在算法 2 中，假设关系 R 的子孙关系的属性总数为 d，那么最坏情况下每个关系模式的时间复杂度为 $O(n \cdot k \cdot 2^k + n \cdot d \cdot 2^{k+d})$，如果是 m 个关系模式，则总的时间复杂度为 $O(m \cdot (k+d) \cdot 2^{k+d})$。

6 结束语

本文研究了 XML 函数依赖以及 XML 键的形式化定义，基于以上定义提出如何检测现有的 XML 文档中的函数依赖的算法。以后的工作将在此基础上对 XML 的冗余消除算法、XML 范式及规范化理论等问题进行研究。

参考文献

[1] Arenas M, Libkin L. Normal Form for XML Documents[J]. ACM Transactions on Database Systems, 2004, 29(1): 195-232.

[2] Vincent M, Liu Jixue, Liu Chengfei. Strong Functional Dependencies and Their Application to Normal Forms in XML[J]. ACM Transactions on Database Systems, 2004, 29(3): 445-462.

[3] Buneman P, Davidson S, Fan Wenfei, et al. Keys for XML[C]//Proc. of the 10th International WWW Conference. Hong Kong, China: [s. n.], 2001: 201-210.

[4] 谈子敬, 庞引明, 施伯乐. XML 上的函数依赖推理[J]. 软件学报, 2003, 14(9): 1564-1570.

[5] Codd E F. A Relational Model of Data for Large Shared Databanks[J]. Communications of the ACM, 1970, 13(6): 377-387.

编辑　任吉慧

(上接第 55 页)

3 结束语

本文把 P2P 数据集成的方法思想引入到网格环境中，把其中每个节点当作一个 Peer，同时建立起了语义映射连接，因此，每个节点的地位都是平等的，解决了旧有数据集成的瓶颈问题。同时给出一个数据集成模型及其分解-再组成算法。通过该算法，只要对系统中的 1 个节点给出 1 个查询语句，系统就能自动地给出全部相关数据源的全部的访问，大大方便了用户的使用。

参考文献

[1] Sheth A P, Larson J A. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Database[J]. ACM Computing Surveys, 1990, 22(3): 183-236.

[2] Levy A Y, Rajaraman A, Ordille J J. Querying Heterogeneous Information Sources Using Source Descriptions[C]//Proceedings of the 22nd VLDB Conference. Bombay, India: [s. n.], 1996.

[3] Antonioletti M. OGSA-DAI: Two Years on[C]//Proc. of the Future of Grid Data Environments Workshop. Berlin, Germany: [s. n.], 2004.

[4] Alpdemir M N, Mukherjee A, Gounaris A, et al. OGSA-DQP: A Service for Distributed Querying on the Grid[C]//Proc.of EDBT' 04. Ctete, Greece: [s. n.], 2004.

[5] Halevy A Y, Suciu D, Tatarinov I. Schema Mediation in Peer Data Management Systems[C]//Proc. of the 19th International Conf. on Data Engineering. Bangalore, India: [s. n.], 2003.

[6] Clark J, DeRose S. XML Path Language (XPath) Version 1.0, W3C Recommendation[Z]. (1999-11-01). http://www.w3.org/TR/XPath.

编辑　顾姣健