文章编号:1001 - 9081(2009)08 - 2303 - 03

基于轴节点的 XML Schema 到关系模式的映射

任廷艳,余建桥
(西南大学 计算机与信息科学学院,重庆 400715)
(yudian86@163.com)

摘 要:DTD 模式不支持复杂元素类型定义,在引入 Schema 形式化定义的基础上,给出 XML 上的复杂元素和函数依赖的定义,提出一种基于轴节点的映射算法。该算法根据轴节点和 XML 函数依赖生成关系表,能保持 XML 文档的内容和结构信息,保持函数依赖,减少存储冗余,并且证明映射后的关系模式满足 3NF。

关键词:关系模式;函数依赖;轴节点
中图分类号: TP311.5 **文献标志码**:A

Mapping from XML schema to relation mode based on pivot node

REN Ting-yan, YU Jian-qiao
(*Faculty of Computer and Information Science, Southwest University, Chongqing* 400715, *China*)

Abstract: DTD does not support the definition of the complex elements. Based on the formal definition of XML schema, the complex elements and the functional dependence were defined in XML. A new mapping method based on pivot node was proposed; it obtained the relation tables according to the pivot node and the XML functional dependences. The constraints that were represented by the XML functional dependences, as well as the content and the structure, were preserved at the same time. Much storage redundancy can be reduced. Furthermore, the relations mapped from XML were proved in Third Normal Form (3NF).

Key words: relation mode; functional dependency; pivot node

0 引言

XML 已经成为 Internet 上的主要数据交换标准之一。目前,在 XML 模式转换为关系模式的研究中大多采用 DTD 作为 XML 的模式描述语言,比如:文献[1]基于函数依赖到关系模式的算法,对 DTD 树模型按层次结构转换生成相应的对象表,再对特定函数依赖生成关系表;但转换过程中存在多次遍历模式树的不同节点和边,算法较复杂;文献[2]基于键的 XML 模式到关系模式的规范化转换,以 XML 键为中心对属性进行划分,生成关系表,其中寻找 XML 键的过程过于繁琐,对所有节点都要进行操作;以及对象关系模式算法[3],信息保留算法[4]等。但是在 DTD 功能上存在不足:采用了非 XML 的语法规则、不支持复杂数据类型和用户自定义类型、扩展性较差等,缺乏对 XML 文档的内容及其语义的约束机制,在模式映射过程中会丢失语义约束信息。而 XML Schema 采用与 DTD 完全不同的语法,是格式良好的 XML 文档,数据描述能力更强,已逐渐替代 DTD 成为 XML 模式描述语言的标准。

针对以上问题,本文提出了一种 XML Schema 到关系模式映射方法。在 Schema 形式化定义中给出了复杂元素的类型集合,定义了与复杂元素相关的轴节点,提出引入轴节点的映射算法。该算法对轴节点和结构良好的函数依赖建立关系表,在映射过程中保持了 XML 文档内容、结构及函数依赖,生成满足 3NF 的关系模式。

1 Schema 和 XML 文档

XML Schema[5]作为 XML 文档的模式语言,其作用是描述 XML 文档的合法结构、内容和限制。在 Schema 中数据类型分为两类:一类是简单数据类型,包括内置的 44 种数据类型和用户自定义的简单数据类型;一类是复杂数据类型,由用户自定义。简单数据类型仅包含信息,不包含子元素或子属性。复杂数据类型可以包含信息,至少包含子元素或子属性中的一种,并用 minOccurs 和 maxOccurs 表示子元素出现的次数。

下面首先引入文献[6]中忽略元素顺序的 Schema 形式化定义。

定义 1 Schema 是一个三元组 (E,T,r):

1) E 是有限的元素标记集合(这里对元素和属性做相同的处理,用 @ 标记属性,元素名和属性名不相同);

2) T 是有限的元素的类型集合,对于每一个 $e \in E$,对应有 $\tau \in T$,记作 $(e:\tau)$,τ 表示为:$\tau ::= \text{str} \mid \text{int} \mid \text{float} \mid \text{SetOf}\ \tau \mid \text{Rcd}[e_1:\tau_1,\cdots,e_n:\tau_n]$,其中 Rcd 表示复杂类型元素,SetOf 表示元素的 *maxOccurs* 大于 1;

3) $r \in E$ 是根元素的标记,它的元素类型不能为 SetOf τ。

定义 2 对于 Schema 的元素 $e \in E$,对应有 $\tau = \text{SetOf}\ \tau \mid \text{Rcd}[e_1:\tau_1,\cdots,e_n:\tau_n]$,则称 e 为轴元素(pivot node)。

对于任一 Schema,通过类型集合 T 可以找出其中所有的轴元素。

定义 3 从 Schema 的定义可知,Schema 中的元素和属性

收稿日期:2009 - 02 - 26;**修回日期**:2009 - 04 - 13。

作者简介:任廷艳(1983 -),女,贵州都匀人,硕士研究生,主要研究方向:XML 数据库; 余建桥(1957 -),男,重庆北碚人,教授,主要研究方向:数据库、信息系统。

组织成一个树结构,称为 Schema 的模式树。在模式树中每个节点用三元组表示(e, τ,@key),其中 @key 为 Schema 模式树中相应节点的前序遍历编号,其中轴元素对应生成的节点为轴节点。

定义4 符合给定Schema $S(E,T,r)$ 的XML文档的树模型 $T=(N,P,V,n_r)$:

1)N 是有限的节点集合;

2)$n_r \in N$ 表示根节点;

3)P 是 N 中节点到后续节点的映射,对于一个 $p=(n',n) \in P, n \in N(n \neq n_r), n' \neq n, n'$ 为父节点,n 为子节点;

4)V 表示赋值的集合,对于每一个叶节点 $n \in N$,有 $v=(n,s)$,s 为简单类型的值。

基于 Schema 的树型结构,给出路径的定义。

定义5 Schema S 上的路径表达式 $paths(e_k)$ 为 $/e_1/e_2/\cdots/e_k$,其中 $e_i \in E, e_1=r, e_i$ 有相关的元素类型 $\tau_i::=\mathrm{Rcd}[\cdots,e_{i+1}:\tau_{i+1},\cdots], i \in [1,k-1]$。

使用 $first(path(e_k))$ 和 $last(path(e_k))$ 分别表示路径表达式的第一个和最后一个元素或属性。

2 函数依赖

由于XML文档是一个带有层次的树形结构,在XML中函数依赖会涉及到XML树中成立的范围,其次函数依赖还要考虑简单数据类型和节点类型是否相等。

首先"相等"是函数依赖中的一个重要概念,在XML文档树上有如下定义。

定义6 节点值相等。对于文档数 $T_1(N_1,P_1,V_1,n_{r1})$ 的节点 n_1 和文档数 $T_2(N_2,P_2,V_2,n_{r2})$ 的节点 n_2,两者元素值相等,记作 $n_1 =_{ev} n_2$,当且仅当:

1)n_1, n_2 同时存在且有相同的标志;

2)存在集合 M,有 $(n_1', n_2') \in M, n_1' =_{ev} n_2', n_1', n_2'$ 是 n_1, n_2 的子元素,n_1, n_2 所有的子元素都满足集合 M;

3)$(n_1,s) \in V_1$ 当且仅当 $(n_2,s) \in V_2$。

定义7 路径值相等。对于文档数 $T_1(N_1,P_1,V_1,n_{r1})$ 的路径 p_1 和文档数 $T_2(N_2,P_2,V_2,n_{r2})$ 的路径 p_2 路径值相等,记作 $T_1.p_1 =_{pv} T_2.p_2$,当且仅当存在一个集合 M':

1)对于 M' 中的 $m'=(n_1,n_2), n_1 \in N_1, n_2 \in N_2$, $path(n_1)=p_1$, $path(n_2)=p_2$,并且 $n_1 =_{ev} n_2$;

2)所有 p_1, p_2 中的元素都满足 M'。

定义8 XML上的函数依赖。

给定 Schema $S(E,T,r)$ 和 XML 文档树 T, 函数依赖 FD_{XML} 定义为如下形式

$FD_{\mathrm{XML}}: (Q[P_{x1}, P_{x2}, \cdots, P_{xn}] \rightarrow P_y)$

其中:

1)FD_{XML} 为 XML 函数依赖名;

2)Q 是范围路径, 定义约束保持的范围。若为 ε,为绝对的 XML 函数依赖,否则为相对的 XML 函数依赖;

3)$P_{x1}, \cdots, P_{xn}$ 是左部路径, $P_{xi} \in paths(S)$, $i \in [1, \cdots, n]$;

4)P_y 是右部路径, $P_y \in paths(S)$。

结合轴元素的定义,我们对 FD_{XML} 进行划分,定义一种重要的函数依赖。

定义9 结构良好的函数依赖。

对于 $FD_{\mathrm{XML}}: (Q[P_{x1}, P_{x2}, \cdots, P_{xn}] \rightarrow P_y)$,如果:

1)$last(P_{xi})$ 不为轴元素,$i \in [1, \cdots, n]$;

2)$last(P_y)$ 不为轴元素。

把满足上述条件的 FD_{XML} 称为结构良好的函数依赖。结构良好的函数依赖在构建关系表时不会引起歧义,是有意义的。在本文中我们只考虑结构良好的函数依赖到关系模式的映射。

3 XML到关系模式的映射

通过上述定义,这里给出了基于 Schema 的保持函数依赖的映射算法。

算法1 XML模式转换为关系模式

输入:Schema S 和 S 上的函数依赖集 FD_{XML}。

输出:关系模式 R。

步骤如下:

1) 根据给定的 Schema S 创建一棵 Schema 模式树。

2) 找出函数依赖集 FD_{XML} 中结构良好的函数依赖。为每个结构良好的函数依赖 $(Q[P_{x1}, P_{x2}, \cdots, P_{xn}] \rightarrow P_y)$ 创建关系,其中属性从 $P_{x1}, P_{x2}, \cdots, P_{xm}$, P_y 中提取, 即 $last(P_{xi})$, $last(P_y)$,指定从相应的 $P_{x1}, P_{x2}, \cdots, P_{xm}$ 提取的属性为码,并从图中删除相应的节点 $last(P_y)$。

3) 为根节点创建关系,属性为空。

4) 扫描 Schema 树,识别树中需要创建相应独立关系的轴节点 D。

5) 选定当前未处理的轴节点 D,为其创建新的关系表。

a) 结构图查找 D 所有的儿子节点,如果儿子节点为叶子节点,则将儿子节点内联到当前表中;如果儿子节点为轴节点,则不做处理。

b) 该节点 D 无子节点,或 D 的所有儿子节点都是轴节点,内联 D 自身到当前表中。

6) 继续读取尚未处理的轴节点,直至遍历整个结构图。

7) 在每个由轴节点生成的关系表中添加属性 id,记录其对应节点的 @key 值。添加属性 $parid$(父节点的 @key) 作为外键来记录父子关系。

4 算法分析

在映射过程中需要对轴节点和结构良好的函数依赖进行转换。若函数依赖集 FD_{XML} 中函数依赖个数为 n_1,Schema 模式树中的节点为 n_2,算法的执行只需扫描一遍函数依赖集 FD_{XML} 和 Schema 模式树,最多进行 n_1+n_2 次到关系模式的转换即可完成 XML 模式到关系模式的映射,因此算法 1 的时间复杂度为 $O(n)$。

在算法中通过步骤(2) 为每个 FD_{XML} 创建关系,保持了XML文档的函数依赖;映射过程中对XML文档中的属性和元素作相同的处理,XML 文档的内容得到保持;并在生成的关系模式中添加主键 id 和外键 $parid$,确保了文档的结构不会改变。

命题 给定 Schema S,S 上的函数依赖集 FD_{XML} 和满足 S 的 XML 文档 T:使用算法 1 得到相应的关系模式集 $R(R_1,R_2,\cdots,R_n)$ 满足 3NF。

证明

1) 对于 R_i 中每个属性都是 Schema S 中的简单类型元素(属性),是不可分的数据项,满足 1NF。

2) 在由轴节点生成的关系中主键都为 id,每一个非主属性都完全函数依赖以 id;同样由结构良好的函数依赖生成的关系中,指定从相应的 P_{x1},P_{x2},…,P_{xm} 提取的属性为码,每一个非主属性都完全函数依赖以码;满足 2NF。

3) 根据算法 1 映射过程 2) 可知,当为每个结构良好的函数依赖生成关系后,在模式树中删除了相应的节点 $last(P_y)$,不存在传递依赖。每一个非主属性既不部分依赖于码也不传递依赖于码,每个 FD_{XML} 产生的关系满足 3NF。

综上可知,R 满足 3NF。

下面通过一个例子来说明算法的执行过程。该算法不仅对本例成立,实验证明对于任何有效的 Schema 均成立。

例 1　学校开设一组课程,每个课程有其课程编号和名称以及选修该门课程的多个学生,对于每个学生,记录其唯一标志的学号、姓名、课程的学分、课程成绩、监护人。

按照定义 1 给出的 Schema 形式化表示如下:

```
school: Rcd
  course: SetOf Rcd
    @cno: int
    cname: str
    student: SetOf Rcd
      @sno: int
      sname: str
      credit: int
      grade: float
      guardian: SetOf str
```

其中存在函数依赖:

FD_{XML1}: ([school.course.@cno] → school.course);

FD_{XML2}: ([school.course.@cno] → school.course.student.credit);

FD_{XML3}: ([school.course.student.@sno] → school.course.student.sname);

FD_{XML4}: (school.course.[school.course.student.@sno] → school.course.student);

FD_{XML5}: ([school.course.@cno, school.course.student.@sno] → school.course.student.grade)

其中对于 FD_{XML1},由于 course 为轴节点,根据定义 9 可知不满足结构良好的函数依赖,不处理;同理 FD_{XML4} 中 student 为轴节点,不处理。

例 2　采用算法 1 为例 1 生成关系:

1) 使用 Schema 模式树来表示 Schema 的结构,如图 1 所示。

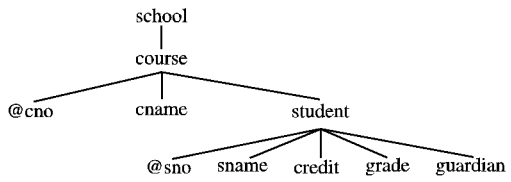

图 1　Schema 模式树

2) 为每个结构良好的函数依赖生成关系表:

RFD_{XML2}: (@cno, credit)

RFD_{XML3}: (@sno, sname)

RFD_{XML5}: (@cno, @sno, grade)

同时删除 Schema 模式树中相应的节点 credit、sname、grade。

3) 根节点生成的关系表:

Rschool();

4) 在图中找出所有的轴节点,有 school、course、student、guardian。

5) ~6) 为轴节点创建关系:

Rcourse(@cno, cname)

Rstudent(@sno)

Rguradian(guradian)

7) 在每个轴节点生成的关系表中添加属性 id, parid,以记录结构信息:

Rschool(id, parid)

Rcourse(id, parid, @cno, cname)

Rstudent(id, parid, @sno)

Rguradian(id, parid, guradian)

最后输出的关系模式 R 如下所示:

Rschool(id, parid)

Rcourse(id, parid, @cno, cname)

Rstudent(id, parid, , @sno)

Rguradian(id, parid, guradian)

RFD_{XML2}: (@cno, credit)

RFD_{XML3}: (@sno, sname)

RFD_{XML5}: (@cno, @sno, grade)

5　结语

本文研究了基于 Schema 的 XML 模式与关系模式的映射问题。引入 Schema 和 XML 文档的定义,依据函数依赖的思想,在 Schema 结构中提出轴节点以区分不同类型的元素,解决了 DTD 不能定义复杂元素的问题,并给出 XML 模式与关系模式映射的算法,该算法解决了 XML 中复杂元素的模式转换问题,并能够有效地保持 XML 模式的内容、结构信息,和函数依赖。

今后的工作是对带递归的 Schema 模式的映射和生成关系中冗余属性的简化,得到规范化的 XML 模式,有助于寻找更优的关系存储策略。

参考文献:

[1] 王庆,周俊梅,吴红伟,等. XML 文档及其函数依赖到关系的映射[J]. 软件学报,2003,14(7):1275 - 1281.

[2] 王梅娟,鲍培明,赵改连. 基于键的 XML 模式到关系模式的规范化转换[J]. 计算机科学,2007,34(3):95 - 100.

[3] MLYNKOVA I, POKORNY J. XML in the world of (object2) relational database systems[R]. Prague, Czech: Charles University, 2003.

[4] BARBOSA D, FREIRE J, ENDELZON A. Designing information preserving mapping schemes for XML[C]// Proceedings of the 31st VLDB Conference. Trondheim Norway: VLDB Endowment, 2005: 109 - 120.

[5] XML Schema Part 0: Primer Second Edition[S/OL]. W3C. [2009 - 02 - 01]. http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/.

[6] YU CONG, JAGADISH H V. XML schema refinement through redundancy detection and normalization [J]. The VLDB Journal, 2008, 17(2): 203 - 223.

[7] 谈子敬,施伯乐. 函数依赖和规范化在关系和 XML 间的传播[J]. 软件学报,2005,16(4):535 - 539.

[8] ARENAS M, LIBKIN L. A normal form for XML documents[J]. ACM Transactions on Database Systems, 2004, 29(1): 195 - 232.