维普资讯 http://www.cqvip.com

XML DTD 规范化处理应用研究*)

丘 威[1] 张立臣[2]

(嘉应学院计算机系 梅州 514015)[1] (广东工业大学计算机学院 广州 510090)[2]

摘 要 从消除 XML DTD 文档内数据冗余的角度出发研究了文档的规范化问题,首先引入 XML DTD 上路径和函数依赖的定义,并提出定义 XML 上的数据冗余;其次基于函数依赖,提出了规范化的 DTD 概念和 XML DTD 规范化处理规则;最后给出了一个将 XML DTD 转化为规范化的处理算法。

关键词 规范化,XML DTD,函数依赖,规则

Research and Application of Normalization for XML DTD

QIU Wei[1] ZHANG Li-Chen[2]

(Department of Computer Science and Technology, Jiaying University, Meizhou 514015)[1]

(Faculty of Computer Science, Guangdong University of Technology, Guangzhou 510090)[2]

Abstract The normalization problem of XML DTD is studied, as viewed from avoidance of redundant information in documents. First, the paper gives the definition of functndional dependencies and path for XML DTD and the concepts of reduancy is provided. Second, based on the functional dependency, the concept of normalized DTD and normalization rules for XML schema are provided. Last, an algorithm for converting XML DTD into a corresponding normalized one is provided.

Keywords Normalization, XML DTD, Functional dependency, Rule

1 引言

XML 已经成为 Internet 上的主要数据交换标准之一。正如关系数据库有其模式一样,XML 文档也有自己的模式。XML 模式是伴随着 XML1.0 规范的制定而推出的。从模式的第一个方案到目前为止[1],W3C 成员共提交了 5 个模式规范,分别是 XML-Data,DCD(Document Content Description for XML),SOX(Schema for Object-oriented XML),DDML(Document Definition Markup Language)和 XML Schema[2]。直到现在,关于模式还没有一个正式的标准,它仍在不断修改完善中。但是 DTD 良好的一致性、扩展性、易用性、互换性作为 XML 模式的应用备受研究者的青睐。在实际应用中,DTD 是 XML 文档使用最多和应用最成熟的模式。正如关系数据库一样,如果 XML 的模式设计不好,在 XML 文档中同样也有数据冗余和操作异常现象。所以,规范化理论也是设计 XML 模式和半结构化数据库的主要核心组成部分和基础理论[3]。虽然人们已经对关系数据库的规范化理论进行了深入研究,但对于 XML 的规范化问题却并非如此。

下面首先通过一个例子来说明一个设计不好的 DTD 是如何在 XML 文档中引起数据冗余的,以及如何通过合适的 DTD 变换和规范化处理来消除这种数据冗余。

例 1 DTD $D_1(E_1, A_1, P_1, R_1, r_1)$ 描述的是一所高校的选课信息,表示课程(course)、学生(student)和教师(teacher)实体之间的关系:

```
〈! ELEMENT courses(course * )〉
〈! ELEMENT course(title,takenby)〉
    〈! ATTLIST course
            cno CDATA #REQUIRED〉
〈! ELEMENT title(#PCDATA)〉
〈! ELEMENT takenby(student * )〉
〈! ELEMENT student(sname,teacher)〉
    〈! ATTLIST student
            sno CDATA #REQUIRED〉
〈! ELEMENT sname(#PCDATA)〉
〈! ELEMENT teacher(tname)〉
〈! ATTLIST teacher
        tno CDATA #REQUIRED〉
〈! ELEMENT tname(#PCDATA)〉
```

例 1 中的 *DTD* D_1 对应的一个 XML 文档树如图 1 所示。

基于树型结构,我们给出路径和树元组的定义。

定义 1 Paths(路径) 给定 DTD $D(E,A,P,R,r)$ 和满足 D 的 XML 树 $T(V,lab,ele,att,val,root)$,$T$ 中的路径定义为 $p=v_1,\cdots,v_n$,其中 $v_1=root, v_i\in ele(v_{i-1}), i\in[2,n-1]$;如果 $lab(v_n)\in E$,那么 $v_n\in ele(v_{n-1})$,称路径 p 为节点类型路径;如果 $lab(v_n)\in A$,那么 $v_n\in att(v_{n-1})$ 或者 $P(lab(v_{n-1}))=S, v_n=S$,称路径 P 为值类型路径[4]。

令 $last(p)=v_n$,表示路径 p 中最后一个节点;$Paths(T)=\{p|p$ 是 T 中的路径$\}$,表示 T 中所有的路径的集合;$EPaths(T)=\{p|p\in Paths(T)$ 且 $last(p)\in E\}$,表示节点类型路径的集合;$VPaths(T)=\{p|p\in Paths(T)$ 且 $last(p)\in A$ 或 $last(p)=S\}$,表示值类型路径的集合。事实上,*XML* 树 $T\models D$,T 中的路径也是 D 中相应路径的映像。

定义 2 Tree Tuples(树元组)[5] 给定 DTD $D(E,A,P,R,r)$ 和满足 D 的 XML 树 $T(V,lab,ele,att,val,root)$,树元组 t 定义为 $Paths(T)$ 到 $V\cup S\cup\{\perp\}$ 的映射,满足:

· 如果 $p\in EPaths(T)$,那么 $t[p]\in V\cup\{\perp\}$,且 $t[r]\neq$

*)国家自然科学基金(No. 60474072、No. 60174050)、广东省自然科学基金(No. 04009465、No. 010059)、广东省高校自然科学研究项目(No. Z03024)基金资助。丘 威 讲师,硕士,研究方向为数据库技术和软件工程;张立臣 教授,博士,主要研究方向:软件工程。

维普资讯 http://www.cqvip.com

⊥；

- 如果 $p\in VPaths(T)$，那么 $t[p]\in S\cup\{\perp\}$；
- 如果 $t[p_1]=t[p_2]$且 $t[p_1]\in V$，那么 $p_1=p_2$；
- 如果 $t[p_1]=\perp$且 p_1 是 p_2 的前缀，那么 $t[p_2]=\perp$；
- $\{p\in Paths(T)\mid t[p]\neq\perp\}$是有限的。

其中 S 和⊥分别表示字符串值和空值。令 $\mathcal{T}[T]=\{t\mid t\in T\}$表示树元组的集合，元组 $t[p]$可记作 $t.p$[6]。

例 2 图 1 文档树的部分树元组为：

$t[courses.course]=course$，元素类型

$t[courses.course.@cno]=\{co1,co2\}$，属性值类型

$t[courses.course.title.S]=\{DB,DW\}$，$S$ 字符串类型

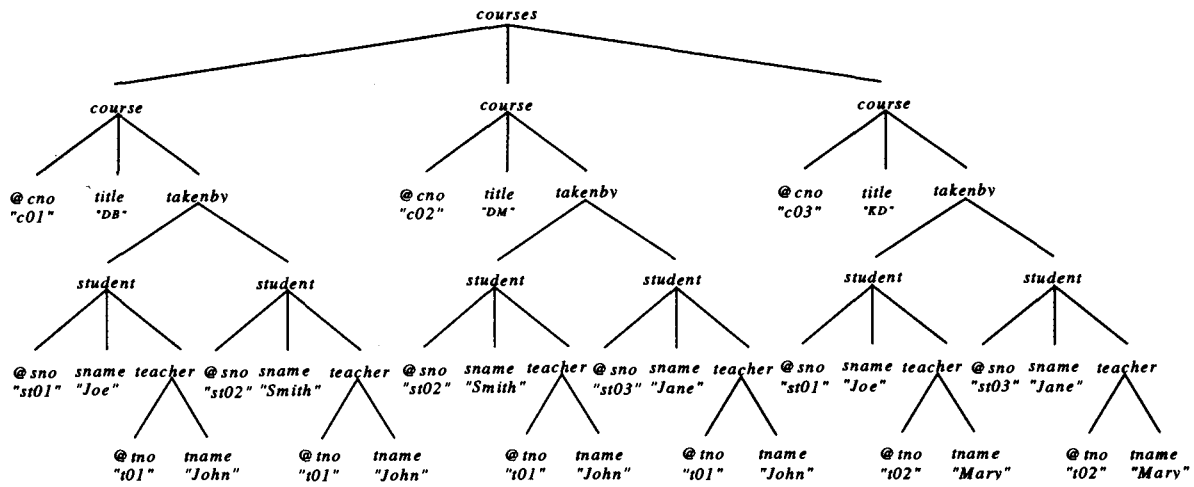

图 1 符合 D_1 的一个 XML 文档

2 XML DTD 规范化处理规则

2.1 XML 函数依赖的概念

这里简单引用 XML 函数依赖的概念。

定义 3 XML Functional Dependency(XML 函数依赖，FD_{XML})[5] 给定 DTD D 和满足 D 的 XML 树 T，D 上函数依赖 FD_{XML}定义为如下形式：

$\varphi:(H,[p_{x1},\cdots,p_{xn}\rightarrow p_{y1},\cdots,p_{ym}])$

其中：

- H 是 XML 函数依赖 FD_{XML} φ 的头部路径，即从 XML 文档的根节点开始的路径，是完全限定路径表达式，定义约束保持的范围；
- $p_{x1},\cdots,p_{xn}$是 XML 函数依赖 FD_{XML} φ 的左部路径，$p_{xi}\in Paths(T)$，每个 p_{xi} 是左部(Left-Hand-Side，LHS)实体类型，即 $last(p_{xi})\in E\cup A\cup S$；
- $p_{y1},\cdots,p_{ym}$是 XML 函数依赖 FD_{XML} φ 的右部路径，$p_{yj}\in Paths(T)$，每个 p_{yj} 是右部(Right-Hand-Side，RHS)实体类型，即 $last(p_{yj})\in E\cup A\cup S$。

对于树 T 中任意两个树元组 t_1 和 t_2，在头部路径 H 约束保持范围内，若存在 $t_1[p_{xi}]=t_2[p_{xi}]$，$(1\leqslant i\leqslant n)$，一定有 $t_1[p_{yj}]=t_2[p_{yj}]$，$(1\leqslant j\leqslant m)$。

令$\sum=\{\varphi\mid\varphi$是满足 DTD D 和 XML 树 T 上的函数依赖 $FD_{XML}\}$，表示所有 XML 函数依赖的集合。如果 $T\models\varphi$，那么 $T\models\sum$；如果 $T\models\sum$且 $T\models D$，那么 $T\models(D,\sum)$；如果 $T\models\varphi$且 $T\models(D,\sum)$，那么称$(D,\sum)$导出 φ，记为$(D,\sum)\vdash\varphi$。令$(D,\sum)^+=\{\varphi\mid(D,\sum)\vdash\varphi\}$。

定义 4 Partial FD_{XML}(部分函数依赖，PFD_{XML}) 给定 DTD D 和满足 D 的 XML 树 T，D 上函数依赖 $\varphi:(H,[p_{x1},\cdots,p_{xn}\rightarrow p_y])$，如果存在另一个函数依赖 $\varphi':(H',[H.p_{z1},\cdots,H.p_{zm}\rightarrow p_y])$满足$[p_{x1},\cdots,p_{xn}]\rightarrow[H.p_{z1},\cdots,H.p_{zm}]$不成立，且$[H.p_{z1},\cdots,H.p_{zm}]\rightarrow[p_{x1},\cdots,p_{xn}]$也不成立，$H'\subseteq_{Paths}H\subset_{Paths}p_{xi}\subseteq_{Paths}p_y$，$1\leqslant i\leqslant n$，称 φ 为部分函数依赖，记作 PFD_{XML}。

定义 5 Transitive FD_{XML}(传递函数依赖，TFD_{XML}) 给定 DTD D 和满足 D 的 XML 树 T，D 上函数依赖 $\varphi:(H,[p_{x1},\cdots,p_{xn}\rightarrow p_y])$，如果存在函数依赖 $\varphi':(H,[p_{x1},\cdots,p_{xn}\rightarrow p_{z1},\cdots,p_{zm}])$和 $\varphi'':(H',[p_{z1},\cdots,p_{zm}\rightarrow p_y])$，满足$(H,[p_{z1},\cdots,p_{zm}]\rightarrow[p_{x1},\cdots,p_{xn}])$和 $p_y\rightarrow[p_{z1},\cdots,p_{zm}]$不成立，$H'\subseteq H$，称 φ 为传递函数依赖，记作 TFD_{XML}。

定义 5 的直观解释是：当把语义上非直接联系的元素类型组织成树状结构时，就会有部分函数依赖。这样，在 XML 文档中就会有数据冗余及异常现象。也就是说，p_y 是在以 $last(H')$为根的子树内和$[H.p_{z1},\cdots,H.p_{zm}\rightarrow p_y]$有直接的函数依赖关系，即函数依赖$(H'[H.p_{z1},\cdots,H.p_{zm}\rightarrow p_y])$成立，而和$[p_{x1},\cdots p_{xn}\rightarrow p_y]$的函数依赖关系是通过 H 所约束的，也就是通过 DTD 中元素类型之间的树状结构所形成的一种部分函数依赖关系[7]。

由于存在上述的数据依赖关系，造成 XML 文档 D_1 包含冗余的信息：选修了同一门课程(course)的每一个学生(student)，其子元素 teacher 的信息都要重复存储；学生“*st*01”的姓名(sname)“Joe”在他所选修的每一门课程中都要重复存储。如果这些信息发生改变，在相应的子元素下面都要同时更新，否则在 XML 文档中就会产生数据不一致，这种现象称之为更新异常。另外，该模式也必然存在数据的操作异常现象：(1)更新异常。如果要更新学生 joe 的信息，则必须同时更新 XML 文档中在每一名老师下所有的有关 joe 的信息，否则会造成该系学生信息的不一致；(2)插入异常。如果要在该系中存入一名新生的信息，就必须在每一名老师下都插入该学生的信息，否则会造成该系学生信息的不一致；(3)删除异常。如果要在该系中删除一名学生的信息，就必须在每一名老师下都删除该学生的信息，否则也会造成该系学生信息的不一致。这种异常现象正是由于 DTD D_1 在设计上存在问题。为了避免这种异常现象，需要对 DTD 模式进行规范化处理。XML 中存在数据冗余，会导致种种异常现象产

维普资讯 http://www.cqvip.com

生。而它们的产生是由于文档内部数据之间存在某种函数依赖关系。首先需要一种表示这种依赖关系的方法，引进关系模式中的概念，我们称其为 XML 上的函数依赖，显然它比平面或是嵌套关系中的函数依赖都复杂得多，也超出了已有的 DTD 和 XML 模式中键可以表达的范畴。在此基础上，可以形式地讨论怎样才是一个消除了冗余的 DTD，即规范化的 DTD。最后再给 DTD 一个规范化算法，它是基于函数依赖的推理规则集。函数依赖和规范化的理论源于关系数据库更广泛的研究，包括嵌套函数依赖和包括嵌套关系上的范式。DTD 和 XML SCHEMA 提供了 XML 上基本的键定义能力。文[7]在 DTD 规范的基础上，扩展了键的定义能力，可以表述相对键。文[9]更进一步通过分析两类特殊路径的表达式，对文档讨论了键的逻辑蕴涵问题及其计算复杂度分析。对于“相等”，文[7,8]中采取了“交不空”的定义方式，而不是严格意义上的相等，且它们的讨论仅限于键。文[11]通过将 XML 文档映射到关系，讨论了 XML 的范式(XNF)，给出了一个将 DTD 转化为符合 XNF 形式的算法。它的主要问题是模型中的“相等”有二义性。对于元素，相等是结合的重点；对于属性，相等则是值相等。

2.2 XML DTD 规范化处理规则[7]

由于存在上述的数据依赖关系，造成 XML 文档 D_1 包含冗余信息：教师“john”的信息在同一课程(course)的每一个学生(student)的子元素内都要重复存储。如果这些信息发生改变，在相应的子元素下面就会产生数据不一致，这种现象称为更新异常。这种异常现象正是由于 DTD D_1 在设计上存在的问题。为了避免这种异常现象，需要对 DTD 模式进行规范化处理。处理规则如下：

(1)提升规则

给定 DTD，$D(E,A,P,R,r)$ D 上的 PFD_{XML} φ:$(H,[p_{z1}.\sigma_{z1},\cdots,p_{zm}.\sigma_{zm}\rightarrow p_y.\sigma_y])$，在$(D,\Sigma)^+$ 中一定存在另一个 FD_{XML} φ':$(H',[H.p_{z1}.\sigma_{z1},\cdots,H.p_{zm}.\sigma_{z1}\rightarrow p_y.\sigma_y])$，其中 $H'\subseteq_{Paths}H\subset_{Paths}p_{xi}\subseteq_{Paths}p_y$，$1\leqslant i\leqslant n$，$\sigma$ 是属性或者具有形式 $\tau.S$(τ 是元素类型)。提升 $last(p_y)$ 节点，使其成为 $last(H)$ 的子节点，构造一个新的如图 2 所示的 DTD $D'(E',A',P',R',r)$，消除部分函数依赖，以减少由部分函数依赖引起的数据冗余现象。

形式上，DTD $D'(E',A',P',R',r)$，其中 $E'=E$；$A'=A$；$P'(last(H))=\{P(last(H)),last(p_y)\}$；$R'=R$。$P'$的其它定义同 D 中 P 的定义[4,12]。在 Σ 中形如 $p_y.\sigma_y$ 形式的函数依赖在 Σ' 中被转换成 $H.p_y.\sigma_y$ 形式。如：

$(H',[H.p_{z1}.\sigma_{z1},\cdots,H.p_{zm}.\sigma_{z1}\rightarrow p_y.\sigma_y])$

转换成

$(H',[H.p_{z1}.\sigma_{z1},\cdots,H.p_{zm}.\sigma_{z1}\rightarrow H.p_y.\sigma_y])$

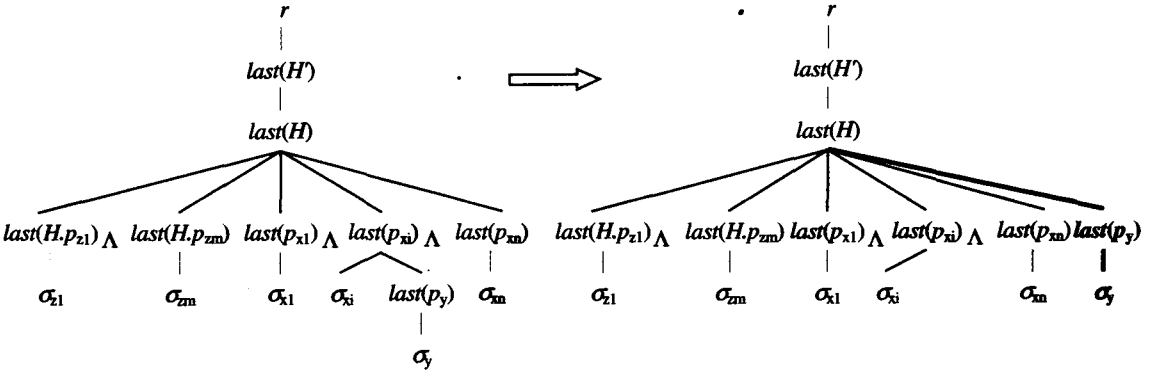

图 2 提升规则示意图

例 3 DTD D_1 经提升规则可以转化为如下 DTD D_2，图 1 可转换为图 3 所示。

```
<!ELEMENT courses (course*)>
<!ELEMENT course (title, takenby, teacher)>
    <!ATTLIST course
        cno CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT takenby (student*)>
<!ELEMENT student (sname)>
    <!ATTLIST student
        sno CDATA #REQUIRED>
<!ELEMENT sname (#PCDATA)>
<!ELEMENT teacher (tname)>
    <!ATTLIST teacher
        tno CDATA #REQUIRED>
<!ELEMENT tname (#PCDATA)>
```

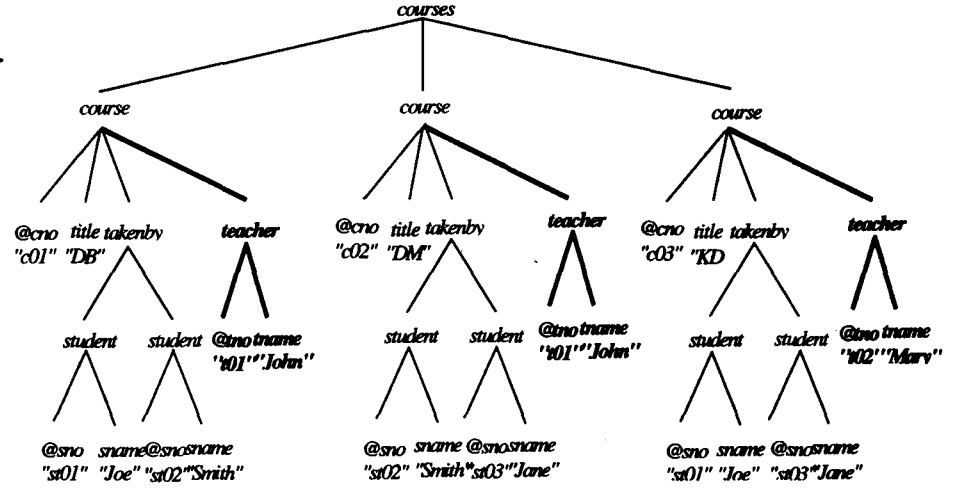

图 3 符合 D_2 的一个 XML 文档

(2)移动规则

给定 DTD $D(E,A,P,R,r)$，D 上的 TFD_{XML} φ:$(H,[p_{z1}.$

维普资讯 http://www.cqvip.com

$\sigma_{x1},\cdots,p_{zm}.\sigma_{zm}\rightarrow p_y.\sigma_y]$),在$(D,\Sigma)^+$中一定存在两个 FD_{XML} φ':$(H,[p_{x1}.\sigma_{x1},\cdots,p_{zm}.\sigma_{zm}\rightarrow p_{z1}.\sigma_{z1},\cdots,p_{zm}.\sigma_{zm}])$和 φ'':$(H',[p_{z1}.\sigma_{z1},\cdots,p_{zm}.\sigma_{zm}\rightarrow p_y.\sigma_y])$,$H'\subseteq H$,$\sigma$是属性或者具有形式$\tau.S$($\tau$是元素类型)。通过创建元素类型$V_{new}$,并移动具有传递依赖的节点来构造一个新的如图 4 所示的 DTD $D'(E',A',P',R',r)$,消除传递函数依赖,以减少由传递函数依赖引起的数据冗余现象。

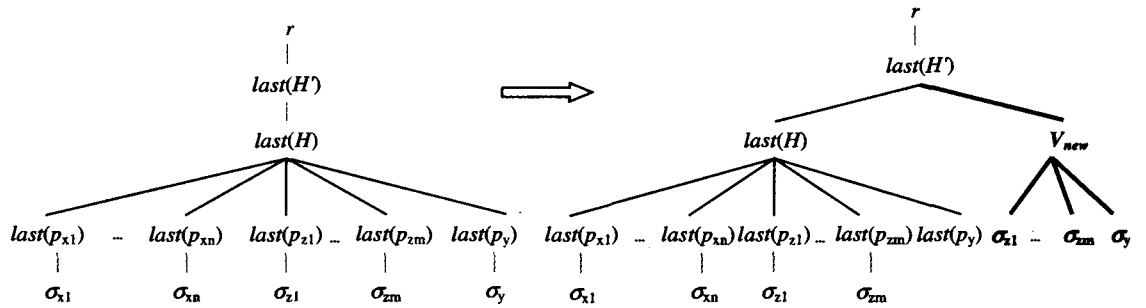

图 4　创建移动规则示意图

形式上,DTD $D'(E',A',P',R',r)$,其中:$E'=E\cup\{lab(V_{new})\}$;$A'=A$;$P'(last(H'))=\{P(last(H)),lab(V_{new})^*\}$;$P'(lab(V_{new}))=[\sigma_{z1},\cdots,\sigma_{zm},\sigma_y]$;$P'(last(p_y))=P(last(p_y))\backslash[\sigma_y]$;$R'(lab(V_{new}))=\{@\sigma_{z1},\cdots,@\sigma_{zm},@\sigma_y\}$;$R'(last(p_y))=R(last(p_y))-\{@\sigma_y\}$;$P'$和$R'$的其它定义同$D$中$P$和$R$的定义[12,13]。在$\Sigma$中形如$p_y.\sigma_y$形式的函数依赖在$\Sigma'$中不再成立,但增加新的函数依赖$\varphi_{new}$:$(H',[H'.V_{new}.\sigma_{y1},\cdots,H'.V_{new}.\sigma_{ym}]\rightarrow H'.V_{new}.\sigma_y)$。

例 4　图 3 经创建移动规则可转换为图 5 所示,其 DTD 为 D_3。

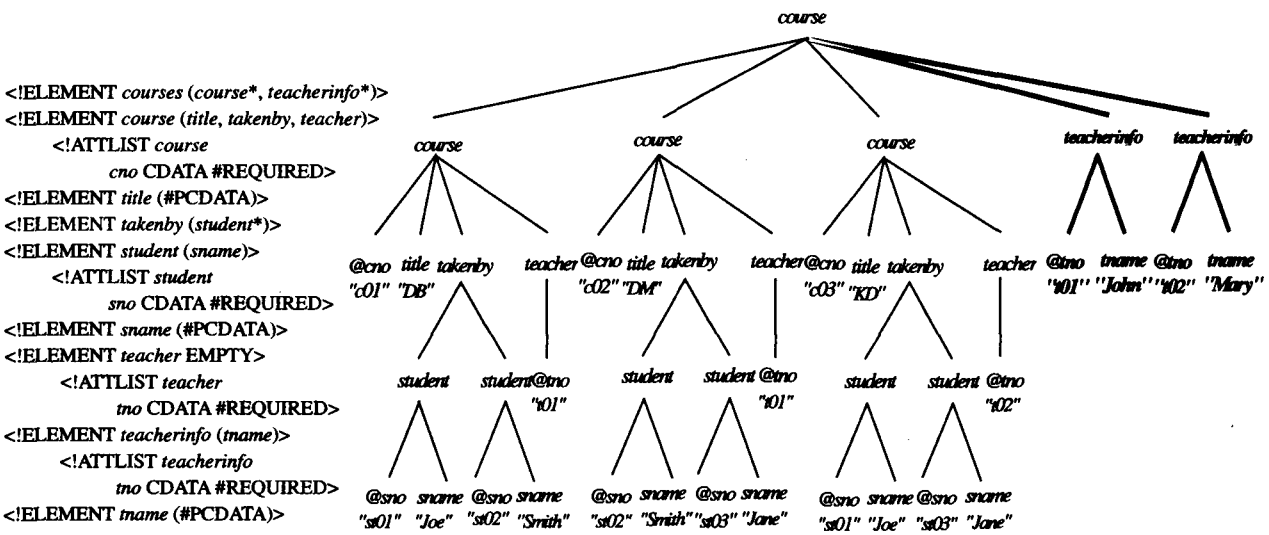

图 5　符合 D_3 的一个 XML 文档

3　XML DTD 的冗余和规范化

3.1　XML DTD 文档中的数据冗余与消除

定义 6　XML 文档中的数据冗余

给定 D,D 上成立的函数依赖集Σ,符合 D 的文档 X 和非平凡的函数依赖 $R_1,R_2(Q_1,Q_2,\cdots,Q_n\rightarrow P_1,P_2,\cdots,P_k)\in\Sigma^+$。设 S 为$Q_1,\cdots,Q_n,P_1,\cdots,P_k$ 的最大公共前缀,令 $Q_i=S/Q_i',P_j=S/P_j'$($i\in[1,n]$,$j\in[1,k]$)。若$\exists v\in\langle R_1\rangle$,$\exists v_1,v_2\in\langle v\{R_2\}\rangle$,$\exists u_1\in\langle v_1\{S\}\rangle$,$\exists u_2\in\langle v_2\{S\}\rangle$($u_1\neq u_2$),有 $u_1\{Q_i'\}\equiv u_2\{Q_i'\}$对于 $i\in[1,n]$都成立,则根据函数依赖的定义,此时必有 $u_1\{P_j'\}\equiv u_2\{P_j'\}$对于 $j\in[1,k]$都成立,这就是文档 X 中的冗余。若非平凡函数依赖 σ 在文档中导致了数据冗余,则称其为异常函数依赖。

定义 7　给定 D 和 D 上成立的函数依赖集Σ,称 D 是规范化的,当且仅当:

对任给的非平凡函数依赖 $R_1,R_2(Q_1,Q_2,\cdots,Q_n\rightarrow P_1,P_2,\cdots,P_k)\in\Sigma^+$,设 S 为$Q_1,\cdots,Q_n,P_1,\cdots,P_k$ 的最大公共前缀,令 $Q_i=S/Q_i',P_j=S/P_j'$($i\in[1,n]$,$j\in[1,k]$),则 $R_1,R_2/S(Q_1',Q_2',\cdots,Q_n')$是 D 上的键。

由此,规范化的 DTD 消除了文档中的冗余[12]。

文档中冗余的存在是导致对其操作异常的原因。规范化的 DTD,也就是消除了文档中冗余的 DTD。给定 D,D 上成立的函数依赖集Σ,符合 D 的文档和非平凡的函数依赖$(R_1,R_2,Q_1,\cdots Q_n,\rightarrow P_1,\cdots,P_n)\in\Sigma^+$。设 S 为$Q_1,\cdots,Q_n,P_1,\cdots,P_n$ 的最大公共前缀,令 $Q_i=S/Q_i,P_j=S/P_j$。若存在 $v\in\langle R_1\rangle$,存在 $v_1,v_2\in\langle v\{R_2\}\rangle$,存在 $u_1\langle v_1\{S\}\rangle$,存在 $u_2\langle v_2\{S\}\rangle$,$(u_1\neq u_2)$,有 $u_1\{Q_i\}\equiv u_2\{Q_i\}$对于 $i\in[1,n]$ 都成立。根据函数依赖的定义,此时必有 $u_1\{P_j\}\equiv u_2\{P_j\}$对于 $j\in[1,k]$都成立,这就是文档 X 中的冗余。若非平凡函数依赖在文档中导致数据冗余,则称其为异常函数依赖[14]。

3.2　XML DTD 规范化处理算法

给定 D 和 D 上的函数依赖集Σ,目标是 D'和 D'上的函

维普资讯 http://www.cqvip.com

数依赖集Σ'，且D'是规范化的。在本节中我们有如下假定：若有函数依赖$(R_1,R_2,Q_1,\cdots,Q_n,P_1,\cdots,P_k)\in\Sigma^+$，则对$T\in\{Q_1\cdots Q_n,P_1\cdots P_k\}$，若$last(T)\in E$，则$M(e)=S$[10]。

算法 1 DTD的规范化算法

步骤 1. 若D是规范化的，算法结束。

步骤 2. 对Σ^+中的异常最大化函数依赖，根据它的形式不同分别处理：

(1)$(R_1,R_2,\text{empty}\rightarrow P_1\cdots P_k)$

D'：将$last(P_1)\cdots last(P_k)$移动成为$last(R_1)$的子元素(属性)，如图 6。

Σ'：Σ^+中只包含D'上路径表达式的函数依赖。

(2)其他一般情况$(R_1,R_2,Q_1\cdots Q_n,P_1\cdots P_k)$

D'：如图 7。

① 增加一个新的元素类型U为$last(R_1)$的子元素：$M'(last(R_1))=M(last(R_1)),U$。

② 增加一个新的元素类型T为U的子元素，$M'(U)=T^*$。

③ 将$last(P_1)\cdots last(P_k)$移动成为$T$的子元素(属性)。

④ 将$last(Q_1)\cdots last(Q_n)$复制成为$T$的子元素(属性)。

Σ'：

1)Σ^+中只包含D'上路径表达式的函数依赖。

2)若$(R_1,R_2,S_1\cdots S_m\rightarrow T_1\cdots T_L)\in\Sigma^+$，其中，$S_i,T_j\in\{Q_1\cdots Q_n,P_1\cdots P_k\},(i\in[1,m],j\in[1,L])$，则

$$(R_1,U/T,last(S_1)\cdots last(S_m)\rightarrow last(T_1)\cdots last(T_L))\in\Sigma' \quad (1)$$

3)$$(R_1,U/T,last(Q_1)\cdots last(Q_n)\rightarrow id)\in\Sigma' \quad (2)$$

步骤 3. $D=D'$，$\Sigma=\Sigma'$，转步骤 1。

算法中有两种情况会生成新的函数依赖，在上面分别标注式(1)和式(2)。式(2)所生成的显然不是异常函数依赖，而若式(1)所生成的为D'中的异常，则原来与其相对应的也必然是D中的异常函数依赖。在算法的整个过程中，异常函数依赖被逐个消除。由于其数目是有限的，所以算法会中止，同时它的输出是规范化的。

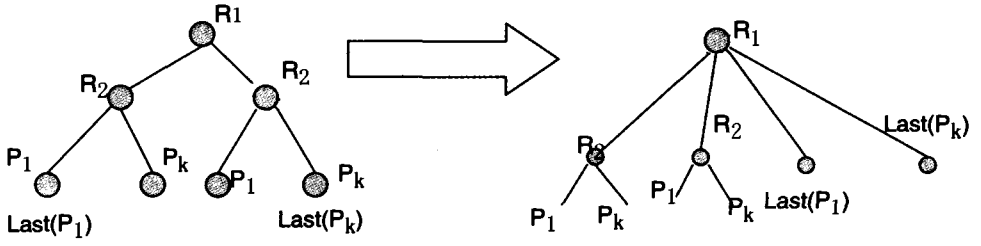

图 6 规范化情况 1

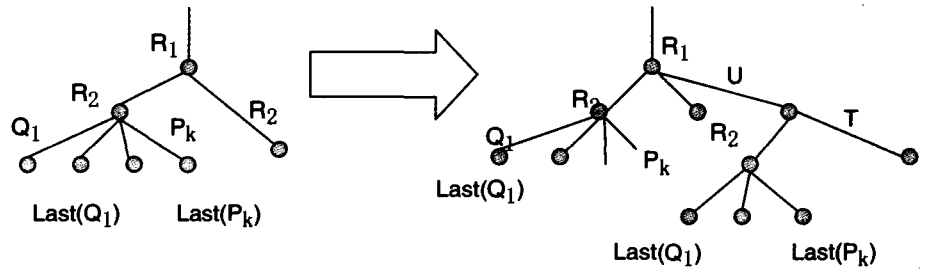

图 7 规范化情况 2

结论与进一步的工作 XML的规范化理论还没有成熟和完善。正如关系数据库中的模式一样，每一种范式都有它的应用范围、特点和局限性，因而不可能解决所有数据冗余问题。通过引入函数依赖的概念，本文给出了一种判定DTD是否设计良好的准则，以及将DTD转化为规范化形式的方法。本文基于XML文档的基本概念给出了XML函数依赖、XML范式的相关定义。提出XML模式规范化转换规则。这将对未来的XML函数依赖保持、XML关键字[12]、XML完整性约束[13,14]、推理规则、XML多值依赖以及XML模式的进一步规范化研究奠定理论基础。

参 考 文 献

1 Extensible Markup Language (XML) 1. 0. Available at http://www. w3. org/TR/1998/ REC-XML-19980210. World Wide WebConsortium. Feb, 1998

2 Vianu V. A Web Odyssey: from Codd to XML. In: Proceedings of ACM PODS, Santa Barbara CA USA, 2001. 148～160

3 谈子敬，施伯乐. DTD的规范化. 计算机研究与发展，2004，41(4)：594～601

4 Arenas M, Libkin L. A Normal Form for XML Documents. In: Symposium on Principles of Database Systems(PODS'02), Madison, Wisconsin, USA: ACM Press, 2002. 85～96

5 Lee Mong-Li , Ling Tok-Wang, Low Wai-Lup. Designing Functional Dependencies for XML. In VIII Conference on Extending Database Technology(EDBT), Prague, March 2002

6 Bosak J, Bray T, Connolly D, et al. W3C XML Specification DTD. http://www. w3. org/XML/1998/06/xmlspec-report-19980910. htm. June 1998

7 张忠平，王超，朱扬勇. 基于约束的XML文档规范化算法. 计算机研究与发展，2005，42(5)：755～764

8 Buneman P, Davidson S B, Fan W F, et al. Reasoning about keys for XML. In: Ghelli G, Grahne G eds. Database Programming Languages, the 8th Int'l Workshop, Frascati: Springer-Verlg, 2001. 133～148

9 Arenas M, Libkin L. A normal form for XML documents. In: Lucian P ed. Proc. of ACM Symp on Principles of Database Systems9PODS). Madison, Wisconsin: ACM Press, 2002. 85～96

10 Fan W, Libkin L. On XML Integerity constraints in the presence of DTDs. In: Proceedings of ACM Symposium on Principles of Database Systems(PODS), Santa Barbara, California, May 2001. 114～125

11 TAN Zi-Jing, SHI Bbi-Le. Propagating Functional Dependency and Normalization Between Relations and XML. Journal of Software. 2005, 16(04): 533～539

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

13 Fan W, Simen J. Integerity constraints for XML. In: Proceedings of ACM Symposium on Principles of Database Systems(PODS), Dallas, Texas, May 2000. 23～34

14 Fan W, Libkin L. On XML Integerity constraints in the presence of DTDs. In: Proceedings of ACM Symposium on Principles of Database Systems(PODS), Santa Barbara, California, May 2001. 114～125