XML文档规范化算法研究

李高仕[1],刘先锋[1],黄海燕[2]
(1.湖南师范大学,湖南 长沙 410081; 2.湖南科技学院,湖南 永州 425100)

摘 要:函数依赖对关系数据库和XML文档都是一种重要的语义表达。文中对XML文档中存在的函数依赖、部分函数依赖和传递函数依赖进行分析,对规范部分函数依赖提出XML第二范式,对规范部分函数依赖和传递函数依赖提出了XML第三范式,给出了相应算法,并进行了无损联接性和函数依赖保持性证明,对可终止性和时间复杂度进行了分析。

关键词:XML;DTD;函数依赖;规范化;算法
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2008)07-0048-05

Study on Normalization Algorithms for XML Documents

LI Gao-shi[1], LIU Xian-feng[1], HUANG Hai-yan[2]
(1. Hunan Normal University, Changsha 410081, China;
2. Hunan University of Science and Engineering, Yongzhou 425100, China)

Abstract: Functional dependency is an important semantic representation both in relational databases and XML documents. Analyzed functional dependency, partial functional dependency and transferrer functional dependency in XML documents, and presented XML the second and the third normal format, then put forward two related normalization algorithms, which is lossless join and preserve functional dependency. At last, terminability and time complexity of the algorithms were analyzed.

Key words: XML; DTD; functional dependency; normalization; algorithm

0 引 言

XML(eXtensible Markup Language)[1]已经成为Internet上主要的数据表示和交换标准之一。XML DTD(Document Type Definition)[2]和XML-Schema[3]等都是XML文档模式。在实际应用中,DTD是XML文档使用最多也是应用最成熟的一种模式,而XML-Schema是DTD的扩展和延伸。但是由于设计上的缺陷,DTD可能存在一些异常数据依赖,从而导致数据冗余和更新、插入和删除异常。由于Web的开放性,XML数据操作异常的危害性要远远大于关系数据操作异常的危害性。因此,对于引起XML数据操作异常的原因及消除操作异常的方法和算法进行研究很有必要。

1 背景知识

定义1(文档类型定义,DTD)[4] DTD定义为$D=(E,A,P,R,r)$,其中:(1)E表示元素类型的有限集合;(2)A表示属性的有限集合;(3)P表示从E到元素类型定义的映射:对于每一个$r\in E$,$P(r)=S$或$P(r)$是如下定义的正则表达式$\alpha:\alpha::=S\mid\varepsilon\mid r\mid\alpha\mid\alpha,\alpha\mid\alpha*$,其中,$S$表示字符串类型(元素类型声明中的#PCDATA),ε是空序列(元素类型声明中的EMPTY),$r\in E$,"|","," 和"*"分别表示并、连接和Kleene闭包;(4)R表示从E到A的幂集$\rho(A)$的映射,对于任意$r\in E$,如果$@l\in R(r)$,称属性$@l$是r的定义;(5)$r\in E$,是根元素类型,不失一般性,对于任意$r\in E$,假定r不出现在$P(r)$中。

定义2(DTD路径)[4] 给定一个DTD $D=(E,A,P,R,r)$,D中的路径定义为$p=v_1\cdots v_n$,其中$v_1=r$,$v_i\in P(v_{i-1})$,$i\in[2,n-l]$,$v_n\in P(v_{n-1})$或$v_n=@l$,$@l\in R(v_{n-1})$。且$\text{length}(p)=n$,$\text{last}(p)=v_n$。用$\text{Paths}(D)$表示D中所有路径构成的集合;$\text{EPaths}(D)=\{p\mid p\in\text{Paths}(D)$且$\text{last}(p)\in E\}$,表示以元素类型结尾的路径构成的集合;$\text{APaths}(D)=\{p\mid p\in\text{Paths}(D)$且$\text{last}(p)\in A\}$,表示以属性结尾的路径构成的集合。

定义3(XML树)[4] 给定DTD $D=(E,A,P,$

收稿日期:2007-10-10
基金项目:国家自然科学基金(10571052)
作者简介:李高仕(1974-),男(苗族),湖南邵阳人,硕士研究生,研究方向为XML半结构化数据、数据库;刘先锋,教授,研究方向为数据挖掘、电子商务技术。

R, r),称 T = (V,lab, ele,att, val, root) 为满足(或符合)D 的XML树,记作 $T \models D$,其中:(1)V 是节点的有限集合;(2)lab是从 V 到 $E \cup A \cup S$ 的映射,对于任意节点 $v \in V$:如果 lab(v) = r且 r $\in E$,v 称为元素节点;如果 lab(v) $\in A$,v 称为属性节点;如果 lab(v) = S,v 称为文本节点;(3)ele是从 V 到 $V*$ 的部分映射,满足对任意 $v \in V$,都有 ele(v) = [$v_1 \cdots v_n$] 且 lab(v_i) $\in P$(r),($i \in [l, n]$);(4)att是从 V 到 A 的部分映射,满足对任意 $v \in V$ 和 $@l \in A$,att(v,$@l$) 有定义,当且仅当 lab(v) = r,r $\in E$ 和 $@l \in R$(r);(5) val是从 V 到字符串值的部分映射,满足对任意 $v \in V$,val(v) 有定义,当且仅当 lab(v) = S 或者 lab(v) $\in A$;(6)root是 V 中唯一节点,lab(root) = r 称为 T 的根。

定义4(XML路径)[4] 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}),$i \in [2, n-l]$;如果 lab(v_n) $\in E$,那么 $v_n \in$ ele(v_{n-1}),称路径 p 为元素节点类型路径;如果 lab(v_{n-1}) $\in A$,那么 $v_n \in$ att(v_{n-1}) 或者 P(lab(v_{n-1})) = S,那么 $v_n = S$,称路径 P 为值类型路径。

2 XML文档中存在的函数依赖

函数依赖属于XML语义范畴,是指XML树节点数据之间的约束关系或者说是依赖关系。

定义5(函数依赖)[5] 设 R(W) 是一关系模式,X, $Y \subseteq W$,关系模式 R 上的一个函数依赖是形如 $f: X \rightarrow Y$ 的一个命题,它的含义是:对于 R 的任意一个可能的实例 r,如果对任意 t_1, $t_2 \in r$, $t_1[X] = t_2[X]$,则必有 $t_1[Y] = t_2[Y]$。$X \rightarrow Y$ 读作"X 函数决定 Y"或"Y 函数决定于 X"。

定义6(XML函数依赖,FD_{XML})[6] 给定DTD D = (E, A, P, R, r),任意的XML文档树 $T \models D$,Paths(D) 为 D 中所有路径的集合,Paths(T) 为文档树 T 上的路径集合,D 上函数依赖 FD_{XML} 定义为:φ:(H, [$p_{x1}, \cdots, p_{xn} \rightarrow p_{yi}, \cdots, p_{ym}$])。其中:(1)$H$ 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,是一个完全限定路径表达式,定义约束保持的范围;(2)$p_{x1}, \cdots, p_{xn}$ 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}) $\in E \cup A \cup S$;(3)$p_{yi}, \cdots, p_{ym}$ 是XML函数依赖 FD_{XML} φ 的右部路径,$p_{yi} \in$ Paths(T),每个 p_{yj} 是右部(Right - Hand - Side,RHS)实体类型,即 last(p_{yi}) $\in E \cup A \cup S$。

定义7(XML部分函数依赖,PFD_{XML})[7] 给定DTD D = (E, A, P, R, r),任意的XML文档树 $T \models D$,Paths(T) 为文档树 T 上的路径集合,D 上函数依赖 φ:(H, [$p_{x1}, \cdots, p_{xn} \rightarrow p_{yi}, \cdots, p_{ym}$]),如果存在另外一个函数依赖 φ':(H', [$H.p_{z1}, \cdots, H.p_{zk} \rightarrow p_{y1}, \cdots, p_{ym}$]),满足 $p_{x1}, \cdots, p_{xn} \rightarrow H.p_{z1}, \cdots, H.p_{zk}$ 和 $H.p_{z1}, \cdots, H.p_{zk} \rightarrow p_{x1}, \cdots, p_{xn} \circ H' \subseteq_{paths} H \subset_{paths} P_{xi} \subseteq_{paths} P_{yl}$,$1 \leqslant i \leqslant n$,$1 \leqslant l \leqslant m$,称 φ 为部分函数依赖,记作 PFD_{XML}。

定义8(XML传递函数依赖集,TFD_{XML}) 给定DTD D = (E, A, P, R, r),任意的XML文档树 $T \models D$,Paths(T) 为文档树 T 上的路径集合,D 上函数依赖 φ:(H, [$p_{x1}, \cdots, p_{xn} \rightarrow p_{y1}, \cdots, p_{ym}$]),如果存在另外两个函数依赖 φ':(H, [$p_{x1}, \cdots, p_{xn} \rightarrow p_{z1}, \cdots, p_{zk}$]) 和 φ'':(H', [$H.p_{z1}, \cdots, H.p_{zk} \rightarrow p_{y1}, \cdots, p_{ym}$]),满足($H$, [$p_{z1}, \cdots, p_{zk} \rightarrow p_{x1}, \cdots, p_{xn}$]) 和 $p_{y1}, \cdots, p_{ym} \rightarrow p_{z1}, \cdots, p_{zk} \circ H' \subseteq_{paths} H$,称 φ 为传递函数依赖,记作 TFD_{XML}。

3 规范化

如何判定一个DTD是否是设计良好的呢?同关系数据库一样,为了区分XML模式的优劣,给出XML模式的不同级别的范式。

3.1 XML的范式

定义9(XML第一范式,X1NF)给定DTD D 和 D 上的函数依赖集 $\sum$,如果元素节点、属性节点和文本节点的值是不可分解的原子值,那么称 D 为XML第一范式的,记作 $D \in$ X1NF。如果某个XML中每个DTD模式都是第一范式的,则称该数据库模式是属于第一范式的。

定义10(XML第二范式,X2NF)给定DTD D 和 D 上的函数依赖集 $\sum$,如果 $D \in$ X1NF,并且在(D, $\sum$)$^+$ 中不存在部分函数依赖,则称 D 是属于XML第二范式的,如果某个XML中每个DTD模式都是第二范式的,则称该数据库模式是属于第二范式的。

定义11(XML第三范式,X3NF)给定DTD D 和 D 上的函数依赖集 $\sum$,如果 $D \in$ X1NF,并且在(D, $\sum$)$^+$ 中不存在部分函数依赖和传递函数依赖,则称 D 是属于XML第三范式的,如果某个XML数据库模式中每个DTD模式都是第三范式的,则称该数据库模式是属于第三范式的。

3.2 规范化

定义12(关系模式分解)[5] 关系模式 R(U,F) 的一个分解 ρ 是若干个关系模式的一个集合:ρ =

$\{R_1(U_1,F_1),R_2(U_2,F_2),\cdots,R_n(U_n,F_n)\}$。其中：

1) $U=\bigcup_{i=1}^{n}U_i$，即关系模式 R 的属性全集 U 是分解后所有小关系模式的属性集 U_i 的并。

2) 对每个 $i,j(1\leqslant i,j\leqslant n)$ 有 $U_i\not\subseteq U_j$。3) $F_i(i=1,2,\cdots,n)$ 是 F 在 U_i 上的投影，即 $F_i=\{X\rightarrow Y\mid X\rightarrow Y\in F^+\wedge XY\in U_i\}$。

定义 13(DTD 的无损联接分解)DTD 的无损联接分解可以转化为相应的关系模式的无损联接分解来加以简化，也就是说给定 DTD $D(E,A,P,R,r)$ 及其相应的关系模式 R_D，另一个 DTD $D'(E',A',P',R',r)$ 是从 D 经过某种规则转换得到的，其相应的关系模式是 $(R_1,\cdots,R_n)$，如果关系模式 $(R_1,\cdots,R_n)$ 是对 R_D 的无损联接分解，那么称 D' 是对 D 的无损联接分解。

DTD D 转化为关系的具体方法为：每个元素变成一个表(关系模式)；每个 XML 属性变成列(关系型属性)；一个元素的每个子元素变成表中的列，这个表与子元素指定表之间有一个外键约束；所有同名元素引用一个关系(并且任何属性或子元素关系合并)。

定义 14(依赖保持性) 设关系模式 R 的一个分解为 $\rho=\{\{R_1,F_1\},\{R_2,F_2\},\cdots,\{R_n,F_n\}\}$，$F$ 是 R 上的依赖集，如果对于所有的 $i=1,2,\cdots,\prod R_i(F)$ 中的全部函数依赖的并集逻辑地蕴涵 F 中的全部依赖，则称分解 ρ 具有依赖保持性。

定义 15(DTD 的依赖保持分解)DTD 的依赖保持分解可以转化为相应的关系模式的依赖保持分解来加以简化，也就是说给定 DTD $D(E,A,P,R,r)$ 及其相应的关系模式 R_D 和函数依赖集 F，另外一个 DTD $D'(E',A',P',R',r)$ 是从 D 经过某种规则转换得到的，其相应的关系模式是 $R_1,\cdots,R_n$，函数依赖集为 $F_1,\cdots,F_n$，如果对于所有的 $i=1,2,\cdots,\prod R_i(F)$ 中的全部函数依赖的并集逻辑地蕴涵 F 中的全部依赖，即 $F^+=(F_1\cup\cdots\cup F_n)^+$，称关系模式 $(R_1,\cdots,R_n)$ 是对 R_D 的依赖保持分解，D' 是对 D 的依赖保持分解。

3.3 规范化规则

规则 1 是用来消除 XML 文档中的部分函数依赖的。它是通过在 XML 树中构造一个新的节点来消除 PFD_{XML}，从而达到消除由 PFD_{XML} 所引起的数据冗余的目的。

规则 1(元素提升规则) 给定 DTD $D=(E,A,P,R,r)$，假定存在 $PFD_{XML}\varphi:(H,[p_{x1}.\delta_{x1},\cdots,p_{xn}.\delta_{xn}\rightarrow p_y.\delta_y])$，易知存在 $FD\ \varphi'\in(D,\sum)^+:(H',[H.p_{z1}.\delta_{z1},\cdots,H.p_{zm}.\delta_{z1}\rightarrow p_y.\delta_y])$。其中 $H'\subseteq_{paths}H\subset_{paths}P_{xi}\subseteq_{paths}P_{yl},1\leqslant i\leqslant n,\delta$ 是属性或者具有形式 r.S(r 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 1 所示的 DTD $D'(E',A',P',r)$，消除部分函数依赖。

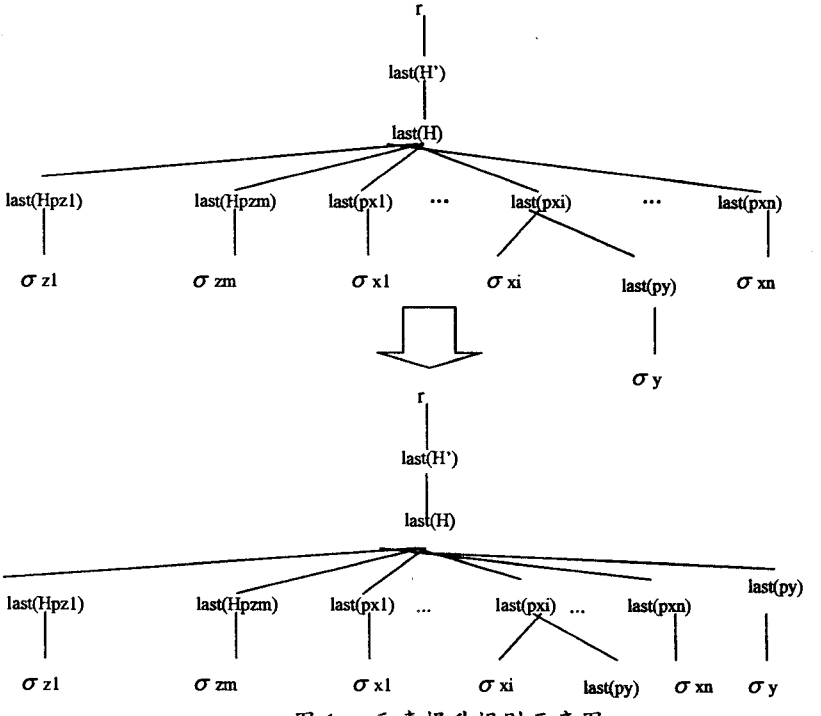

图 1 元素提升规则示意图

规则 2 是用来消除 XML 文档中的传递函数依赖的。它的基本思想是通过构造一个新的元素类型来消除 TFD_{XML}，从而达到消除由 TFD_{XML} 所引起的数据冗余的目的。

规则 2(元素创建规则) 给定 DTD $D=(E,A,P,R,r)$，假定存在 $TFD_{XML}\varphi\in(D,\sum)^+:\varphi:(H,[p_{x1}.\delta_{x1},\cdots,p_{xn}.\delta_{xn}\rightarrow p_y.\delta_y])$，易知存在两个 $FD\ \varphi'\in(D,\sum)^+:(H,[p_{x1}.\delta_{x1},\cdots,p_{xn}.\delta_{xn}\rightarrow p_{z1}.\delta_{z1},\cdots,p_{zm}.\delta_{zm}])$ 和 $\varphi''\in(D,\sum)^+:(H',[p_{z1}.\delta_{z1},\cdots,p_{zm}.\delta_{zm}\rightarrow p_y.\delta_y])$，$H'\subseteq H,\delta$ 是属性或者具有形式 r.S(r 是元素类型)。通过创建元素类型 V_{new}，并移动具有传递函数依赖的节点来构造一个新的 DTD $D'=(E',A',P',R',r)$，如图 2 所示。从而消除由它所引起的数据冗余。

4 规范化算法

算法 1 X2NF - DECOMPOSITION(DTD 分解成

X2NF)

输入:一个规范化的 DTD $D(E,A,P,R,r)$ 和函数依赖集 $\sum$。

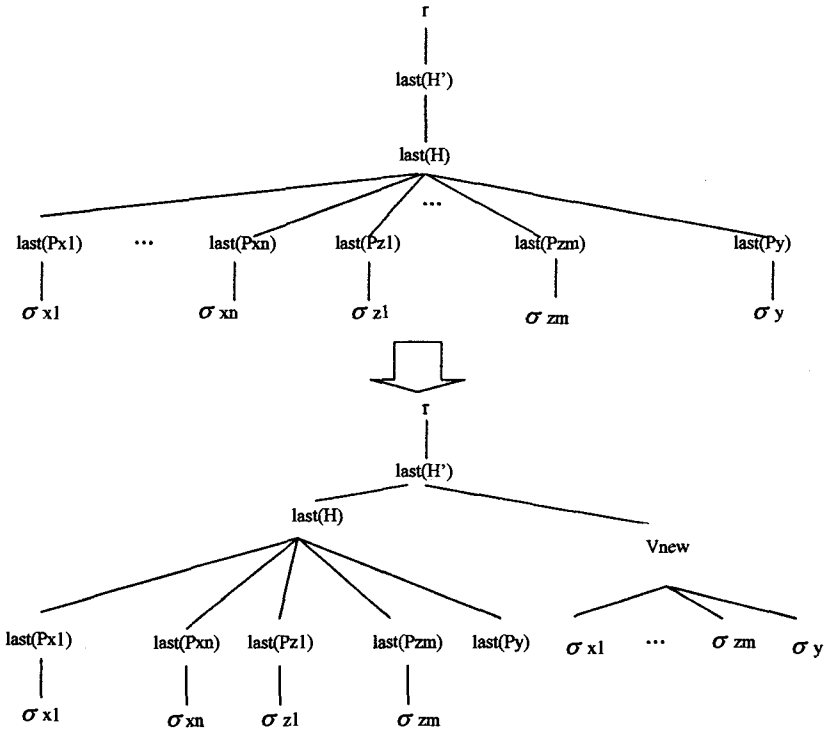

图 2　元素创建规则示意图

输出:满足 X2NF 的 D 的分解 DTD D'。

(1) 初始化:$D' := D, \sum' := \sum$;

(2) FOR 每一个 PFD_{XML} φ: $(H, [p_{x1}.\sigma_{x1}, \cdots, p_{xn}.\sigma_{xn} \rightarrow p_y.\sigma_y]) \in \sum$,do 根据元素提升规则构建新的函数依赖 φ_{new}: $(H, [p_{x1}.\sigma_{x1}, \cdots, p_{xn}.\sigma_{xn} \rightarrow H.p_y.\sigma_y])$; $\sum' = (\sum' - \{\varphi\}) \cup \{\varphi_{\text{new}}\}$;根据 $\sum'$ 构建新的 DTD D_{new}, $D' := D_{\text{new}}$;

(3) 合并 D' 中相同元素类型,消除冗余元素类型;

(4) RETURN(D')。

算法 1DTD D' 是对 DTD D 的无损联接分解和依赖保持分解。

证明:算法 1 运用的是元素提升规则,没有改变原文档的元素类型和属性,所得的 DTD D' 对应的关系模式 R_2 同原来的 DTD D 对应的关系模式 R_1 相等,即 $R_1 = R_2$,且 R_2 属于 X2NF,所以 R_2 是 R_1 的无损联接分解和依赖保持分解,即 DTD D' 是对 DTD D 的无损联接分解和依赖保持分解。

算法 2 X3NF - DECOMPOSITION(DTD 分解成 X3NF)

输入:一个规范化的 DTD $D(E,A,P,R,r)$ 和函数依赖集 $\sum$。

输出:满足 X3NF 的 D 的分解 DTD D'。

(1) 初始化:$D' := D, \sum' := \sum$;

(2) FOR 每一个 TFD_{XML} φ: $(H, [p_{x1}.\sigma_{x1}, \cdots, p_{xn}.\sigma_{xn} \rightarrow p_y.\sigma_y]) \in \sum$,do 根据元素创建规则构建新的函数依赖 φ_{new}: $(H', [H'.V_{\text{new}}.\sigma_{y1}, \cdots, H'.V_{\text{new}}.\sigma_{ym} \rightarrow H'V_{\text{new}}.\sigma_y])$; $\sum' = (\sum' - \{\varphi\}) \cup \{\varphi_{\text{new}}\}$;根据 $\sum'$ 构建新的 DTD D_{new}, $D' := D_{\text{new}}$;

(3) 合并 D' 中相同元素类型,消除冗余元素类型;

(4)RETURN (D')。

算法 2 具有无损联接性。

引理 1[8]设 $\rho = \{R_1, R_2\}$ 是关系模式 R 的一个分解,F 是 R 的一个函数依赖集,则对于 F,具有无损联接性的充分必要条件是 $R_1 \cap R_2 \rightarrow R_1 - R_2 \in F^+$,或 $R_1 \cap R_2 \rightarrow R_2 - R_1 \in F^+$。即如果两个关系模式间的公共属性集至少包含其中一个关系模式的关键字,则此分解必定具有无损联接性。证明见参考文献[8]。

证明:算法 2 运用的是创建移动规则,形成新元素类型。将形如 $R_1(\sigma_{x1}, \cdots, \sigma_{xn}, \sigma_{y1}, \cdots, \sigma_{ym}, \sigma_y, X)$ 形式(其中 X 表示其他的属性集合),转换为包括 $R_{11}(\sigma_{x1}, \cdots, \sigma_{xn}, \sigma_{y1}, \cdots, \sigma_{ym}, X)$ 和 $R_{12}(\sigma_{y1}, \cdots, \sigma_{ym}, \sigma_y)$ 两种形式。由引理 1 知,R_{11} 和 R_{12} 是对 R_1 的无损联接分解。

算法 2 具有依赖保持性。

证明:给定 DTD $D = (E, A, P, R, r)$,假定存在 TFD_{XML} $\varphi \in (D, \sum)^+$: $(Sh, [S_{x1}.\sigma_{x1}, \cdots, S_{xn}.\sigma_{xn}] \rightarrow S_y.\sigma_y)$,从传递函数依赖定义知存在 $FD\ \varphi 1 \in (D, \sum)^+$: $(Sh, [S_{x1}.\sigma_{x1}, \cdots, S_{xn}.\sigma_{xn}] \rightarrow [S_{y1}.\sigma_{y1}, \cdots, S_{ym}.\sigma_{ym}])$ 和 $\varphi 2 \in (D, \sum)^+$: $(Sh', [S_{y1}.\sigma_{y1}, \cdots, S_{ym}.\sigma_{ym}] \rightarrow [S_y.\sigma_y])$。转换成相应关系模式 $R_1(\sigma_{x1}, \cdots, \sigma_{xn}, \sigma_{y1}, \cdots, \sigma_{ym}, \sigma_y, X)$($X$ 为其它属性集合),$F^+ = \{\sigma_{x1}, \cdots, \sigma_{xn} \rightarrow \sigma_y, \sigma_{x1}, \cdots, \sigma_{xn} \rightarrow \sigma_{y1}, \cdots, \sigma_{ym}, \sigma_{y1}, \cdots, \sigma_{ym} \rightarrow \sigma_y\}$。根据规则 2,$D$ 转换成 D_1 和 D_2,D_1、D_2 分别转换成相应关系模式 $R_{11}(\sigma_{x1}, \cdots, \sigma_{xn}, \sigma_{y1}, \cdots, \sigma_{ym}, X)$ 和 $F_{11} = \{\sigma_{x1}, \cdots, \sigma_{xn} \rightarrow \sigma_{y1}, \cdots, \sigma_{ym}\}$、$R_{12}(\sigma_{y1}, \cdots, \sigma_{ym}, \sigma_y)$ 和 $F_{12} = \{\sigma_{y1}, \cdots, \sigma_{ym} \rightarrow \sigma_y\}$。$F_{11} \cup F_{12} = \{\sigma_{x1}, \cdots, \sigma_{xn} \rightarrow \sigma_{y1}, \cdots, \sigma_{ym}, \sigma_{y1}, \cdots, \sigma_{ym} \rightarrow \sigma_y\}$。根据传递规则有:$\sigma_{x1}, \cdots, \sigma_{xn} \rightarrow \sigma_y$。故 $(F_{11} \cup F_{12})^+ = \{\sigma_{x1}, \cdots, \sigma_{xn} \rightarrow \sigma_y, \sigma_{x1}, \cdots, \sigma_{xn} \rightarrow \sigma_{y1}, \cdots, \sigma_{ym}, \sigma_{y1}, \cdots, \sigma_{ym} \rightarrow$

$\{_y\} = F^+$。得证。

算法 1 和算法 2 可终止性分析　算法 1 重复运用提升规则，算法 2 重复运用创建规则对给定的 DTDD 进行转换，每转换一次，$\sum$ 中 PFD_{XML} 或 TFD_{XML} 的个数就减少一个，由于 $\sum$ 中 PFD_{XML} 或 TFD_{XML} 的数目是有限的，所以算法 1 和算法 2 可以在有限的时间内终止。

算法 1 和算法 2 的时间复杂度分析　算法 1 和算法 2 的时间复杂度主要是由 FOR 循环体和合并消除冗余两步骤决定。FOR 循环体每执行一步，消除 $\sum$ 中一个 PFD_{XML} 或 TFD_{XML}，设最坏情况下 $\sum$ 中所有 FD_{XML} 都是 PFD_{XML} 或 TFD_{XML}，令 $\sum$ 中函数依赖数目 $n = |\sum|$，所以整个 FOR 循环体要执行 n 步。算法中的第 3 步合并和消除冗余元素类型是检查节点的冗余情况，令 D' 中节点数目 $m = |D'|$，检查节点的冗余每次取 2 个元素，要执行 C_m^2 步（$C_m^2 = m(m-1)/2 = m^2/2 - m/2$），所以合并和消除冗余元素类型的时间复杂度最坏情形下为 $O(m^2)$，所以整个算法的时间复杂度为 $O(m^2 + n)$。

5 结束语

分析了 XML 文档中的部分函数依赖和传递函数依赖，对规范相应函数依赖提出算法 1 和 2，并对两个算法为无损联接分解和依赖保持分解给出了证明，对算法的可终止性和时间复杂度也进行了分析，证明了这两个算法正确、可行。

参考文献：

[1] Aitken P G. XML－the Microsoft Way[M]. [s. l.]: Addison Wesley, Inc. 2002.

[2] Bray T, Paoli J, Sperberg－McQueen C M. Extensible Markup Language(XML)1.0[EB/OL]. 1998－02. http//www.w3.org/TR/ 199811/REC－XML－19980210.

[3] Biron P V, Maihotra A. XML－Schema Part 2: Datatypes, W3C[EB/OL]. 2000－04. http://www.w3.org/TR/xmlschema2.

[4] Arenas M, Libkin L. A Normal Form for XML Documents [C]//In: Proceedings of ACM Symposium on Principles of Database Systems(PODS). Madison, Wisconsin, USA: [s. n.], 2002:85－96.

[5] 施伯乐，何继潮，崔　靖. 关系数据库理论及应用[M]. 郑州：河南科学技术出版社，1989:355－445.

[6] 谈子敬，施伯乐. DTD 的规范化[J]. 计算机研究与发展，2004,41(4):594－600.

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

[8] 刘方鑫. 数据库原理与技术[M]. 北京：电子工业出版社，2002.

（上接第 44 页）

allel Solving from Nature. NewYork: Springer, 1990:177－188.

[2] 黄高义. 遗传算法实现平面连杆机构运动——轨迹综合[D]. 上海：上海交通大学机械与动力工程学院，2003.

[3] 潘　峰，蔡维由. 改进遗传算法和神经网络在汽轮发电机组故障诊断中的应用[J]. 华中电力，2004,17(2):1－4.

[4] 金朝红，吴汉松，李腊梅，等. 一种基于自适应遗传算法的神经网络学习算法[J]. 微计算机信息，2005,10(1):49－51.

[5] Qi Xiaofeng. Theoretical analysis of evolution algorithms with an infinite population size in continuous space, Part 1,2: basic properties of selection and mutation[J]. IEEE Tran. on Neural Networks, 1994,5(1):102－119,120－129.

[6] 方咸云，方千山，王永初. 双变异率自适应遗传算法研究及其应用[J]. 南昌航空工业学院学报：自然科学版，2006,16(2):17－20.

[7] 毕惟红，任红民，吴庆标. 一种新的遗传算法最优保存策略[J]. 浙江大学学报：理学版，2006,33(3):32－35.

（上接第 47 页）

date generation[C]//In: Proc 2000 ACM－SIGMOD Int Conf Management of Data(SIGMOD'00). TX, Dalas: [s. n.], 2000.

[2] Cheung D W, Han J, Ng V, et al. A fast distributed Algorithm for mining association rules[C]//In: Proc 1996 Int Conf Parallel and Distributed Information Systems. Miami Beach, FL: [s. n.], 1996:31－44.

[3] 铁治欣，陈　奇，俞瑞钊. 关联规则采掘综述[J]. 计算机应用研究，2000,17(1):1－5.

[4] Han Jiawei, Kamber M. 数据挖掘概念和技术[M]. 范　明，孟小峰，等译. 北京：机械工业出版社，2001.

[5] 陈文庆，许　棠. 关联规则挖掘 Apriori 算法的改进与实现[J]. 微机发展，2005,15(8):155－157.

[6] 胡吉明，鲜学丰. 挖掘关联规则中 Apriori 算法的研究与改进[J]. 计算机技术与发展，2006,16(4):99－102.

[7] 朱孝宇，王理冬，汪光阳. 一种改进的 Apriori 挖掘关联规则算法[]. 计算机技术与发展，2006,16(12):89－91.

[8] 何中胜，刘宗田. 一种无候选集产生的并行关联规则挖掘算法[J]. 计算机工程与应用，2004(24):163－165.