基于键的 XML 模式到关系模式的规范化转换*)

王梅娟　鲍培明　赵改连
(南京师范大学计算机系　南京 210097)

摘　要　本文针对 XML 模式向关系模式转换过程中形成的冗余和不规范，提出一种基于 XML 键的 XML 模式到关系模式的规范化转换算法。该算法以键为基础，根据 XML 函数依赖的一组推理规则，扩充函数依赖的集合，避免 XML 模式向关系模式转换过程中语义丢失的情况；再利用一组消解规则，消除冗余的函数依赖，使其满足极小函数依赖集；最后以键为中心划分关系属性，得到关系表，并且保证得到的关系模式满足 3NF。

关键词　XML，关系，键，函数依赖

Translating XML to Normal Relational Schema Based on XML Key

WANG Mei-Juan　BAO Pei-Ming　ZHAO Gai-Lian
(Department of Computer Science, Nanjing Normal University, Nanjing 210097)

Abstract　This paper proposes a Normalized Translation Algorithm for translating XML to relational schema based on XML key, which tackles the redundancy and un-standard expression in the translation. According to a group of deduce rules about XML functional dependency, we expand the FD-Set to avoid losing semantics of original XML. Then we use a group of clear rules to clear up the redundant FD and enable the final FD-Set is minimum. Finally, we take the XML keys as the center to divide the relation attributes, obtain the relation table, and guarantee the relation schema satisfies 3NF.

Keywords　XML, Relation, Key, Functional dependency

1　引言

XML 是一种可扩展的半结构化数据模型，本文借鉴关系数据库中“键”的概念，与 XML 模式特点相结合，提出了一种基于 XML 键的 XML 模式到关系模式的转换算法。该算法首先定义出 XML 中函数依赖的键；其次利用规则重新整理函数依赖集，先针对转换过程中的函数依赖丢失情况，根据一组推理规则扩充完整函数依赖集，再针对扩充后的函数依赖集过于庞大和冗余，利用一组消解规则得到极小的函数依赖集，并给出说明，证明它是极小的；最后归纳得到以键为中心的关系属性划分，对有 XML 键的结点为其建立关系表，生成满足 3NF 的关系模式。

文[5]的映射转换算法是将关系模式中的表分为对象表和关系表两种，首先对 DTD 树模型按层次结构划分转换得到对象表，再针对每一条函数依赖描述的关系生成关系表，虽然该算法得到的关系模式满足 BCNF，但是表的数量和重复的属性非常多，存在大量的冗余；文[9]是先分析 DTD 中存在的部分函数依赖形式和传递函数依赖形式，分别利用提升规则和创建新结点规则进行 DTD 的规范化，再转换为满足 BCNF 的关系模式，过程烦琐且不能保证 DTD 的规范性。本文算法在结合两篇文献的基础上进行改进，以 XML 键为中心进行属性划分，因此比文[5]按层次结构划分得到的关系模式要简洁得多，既减少了表的冗余，也减少了重复属性的冗余；同时，本文的算法是在转换过程中进行规范化，不需要先对 DTD 进行规范化，因此只需要对文档扫描一次，与文[9]相比降低了时间复杂度。

另外，文[5]和文[9]得到的关系模式虽然满足了关系模式的规范化，但是转换过程不满足无损分解；而本文从 XML 模式到关系模式的转换是无损分解过程，可以通过极小函数依赖集中的结点路径还原为原 XML 模式。

2　函数依赖

2.1　DTD 和 XML 文档的定义

定义 1　DTD(文档类型定义)，$D=(E,A,P,R,r)$，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 是从 E 到元素类型定义的映射：对于每一个 $\tau\in E$，$P(\tau)$是正则表达式 $\alpha::=S|\varepsilon|\tau|\alpha,\alpha|\alpha^*$，其中：$S$ 表示字符串类型；ε 是空序列；$\tau\in E$；“,”和“*”分别表示连接和 Kleene 闭包；(4)R 是从 E 到 A 的幂集的映射；(5)$r\in E$ 是树根的元素类型，假定对于任何的 $\tau\in E$，r 不出现在 $P(\tau)$中。

从 DTD 的定义可知，DTD 中的元素类型和属性组织成一个树形结构。为了描述 DTD 结构上的特点，引入文[4]中路径的概念：

定义 2　DTD 的路径，给定 DTD $D=(E,A,P,R,r)$，D 中的路径 p 定义为 $p=w_1,\cdots,w_n$，其中 $w_1=r$；$w_i\in P(w_{i-1})$，$i\in[2,n-1]$；如果 $w_n\in A$，那么 $w_n\in R(w_{n-1})$；否则 $w_n\in P(w_{n-1})$。并且令 $\mathrm{paths}(D)=\{p\mid p$ 是 D 中的路径$\}$。如果 $\mathrm{paths}(D)$是无穷的，那么 D 叫做递归的，否则称为非递归的。本文仅考虑非递归的 DTD，用 $\mathrm{last}(p)$表示路径 p 的最后一个符号，用 $p\subseteq_{\mathrm{path}}q$ 表示路径 p 是路径 q 的一部分或它们相等。

给定 DTD D，符合 DTD 的 XML 文档实际上是 D 的一个映像，XML 文档 T 的定义如下：

定义 3　XML 文档(符合 D 的 XML 树，表示成 $T\vdash$

*)基金项目：江苏省高校自然科学基金(04KJB520075)。**王梅娟**　硕士，主要研究方向为 XML 模式规范化与关系模式转换。**鲍培明**　副教授，研究生导师。**赵改连**　硕士。

$D)$，令 $D=(E,A,P,R,r)$，称 $T=(V,lab,ele,att,val,root)$ 为符合 D 的 XML 树，其中：(1)V 是结点的有限集合；(2)lab 是从 V 到 $E\cup A\cup\{S\}$ 的映射(符号"$\cup$"表示逻辑或的关系，下同)；(3)ele 是从 V 到 V^* 的部分映射，满足对任意 $v\in V$，都有 $ele(v)=\{v_1,\cdots,v_n\}$，当且仅当 $lab(v_i)(i\in[1,n])$ 在 $P(lab(v))$ 中有定义；(4)att 是从 V 到 A 的部分映射，满足对任意 $v\in V$，都有 $att(v)=R(lab(v))$，当且仅当 $lab(v)\in E$ 和 $R(lab(v))$ 在 D 中有定义；(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 $P(lab(v))=S$ 或者 $lab(v)\in A$；(6)$lab(root)=r$ 叫做 T 的树根。

2.2 函数依赖

定义 4 结点值相等，XML 树中的两个结点 $x、y\in V$，它们的值相等可以表示为 $x=_v y$，当且仅当满足下述条件：

(1)$lab(x)=lab(y)$；

(2)如果 $lab(x),lab(y)\in A$ 或者 $lab(x)=lab(y)=S$，那么 $val(x)=val(y)$；

(3)如果 $lab(x),lab(y)\in E$，那么①对于任意属性 $a\in att(x)$，存在 $b\in att(y)$，满足 $a=_v b$，反之亦然；②如果 $ele(x)=v_1,\cdots,v_k$，那么 $ele(y)=w_1,\cdots,w_k$，且对任意的 $i\in[1,k]$，都有 $v_i=_v w_i$，反之亦然。

如果 n 是 XML 树 T 中的一个结点，p 是 T 中的一条路径，那么从结点 n 开始，经路径 p 所能到达的结点集合记为 $n[[p]]$，特别地，$root[[p]]$ 简记为 $[[p]]$。

定义 5 函数依赖，给定 DTD D，D 上的函数依赖(FD) $\varphi:(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_y)$，对于满足 D 的 XML 树 T，称 T 满足函数依赖 φ 当且仅当对 T 中任意结点 $n\in[[S_h]]$(如果 S_h 为空，则令 $n=r$)，如果结点 $x_{1i}\in n[[S_{xi}]],x_{2i}\in n[[S_{xi}]]$ 且 $x_{1i}=_v x_{2i}$ 对所有 $i\in[1,n]$ 都成立，那么就存在结点 $y_1\in n[[S_y]]$ 和 $y_2\in n[[S_y]]$，并且 $y_1=_v y_2$ 成立。

其中：(1)$S_h\in paths(D)$ 叫做函数依赖 φ 的头部路径，定义了 φ 在 D 上成立的范围。该路径的最后一个符号是一个元素类型的名称，即 $last(S_h)\in E$。如果 $S_h\neq\phi$，那么 φ 叫做相对函数依赖，表示 φ 只有在以路径 S_h 最后一个元素类型 $last(S_h)$ 为根的子树范围内才成立(在 DTD 树中，每一个元素对应的结点都是唯一的)；如果 $S_h=\phi$，φ 可表示为 $[S_{x1},\cdots,S_{xn}]\rightarrow S_y$，叫做绝对函数依赖，表示 φ 在整个 D 上均成立。(2)$[S_{x1},\cdots,S_{xn}]$ 叫做 φ 的左部路径。其中对每一个 $i\in[1,n]$，都有 $S_{xi}\in paths(D)$，$S_h\subseteq_{path}S_{xi}$，且 $last(S_{xi})$ 可能是一个元素名称、元素的一个字符串值(表示成 S)或者是一个属性的名称。(3)S_y 叫做 φ 的右部路径。其中 $S_y=\varepsilon$ 或者 $S_y\in paths(D)$，$S_h\subseteq_{path}S_y$，且 $last(S_y)$ 的取值可能是一个元素名称、元素的一个字符串值或者是一个属性的名称。

ε 表示空，当 $S_{yj}=\varepsilon$ 时，函数依赖本身是没有意义的，此时函数依赖的左部路径都是 XML 键，用于建立键-键联系，这一问题在后面叙述。

例 1 假设有一个 DTD $D1$ 及其函数依赖：

$\varphi1$：([courses. course. @cno]→courses. course)；//课程号唯一确定一门课程；

$\varphi2$：([courses. course. @cno]→courses. course. student. credit)；//课程号唯一确定一门课程的学分；

$\varphi3$：([courses. course. student. @sno]→courses. course. student. sname)；//学号唯一确定其姓名；

$\varphi4$：(courses. course，[courses. course. student. @sno]→courses. course. student)；//在同一门课程下，学生的学号唯一确定一个学生；

$\varphi5$：([courses. course. @cno，courses. course. student. @sno]→courses. course. student. grade)；//课程号和学生学号唯一确定该学生的成绩。

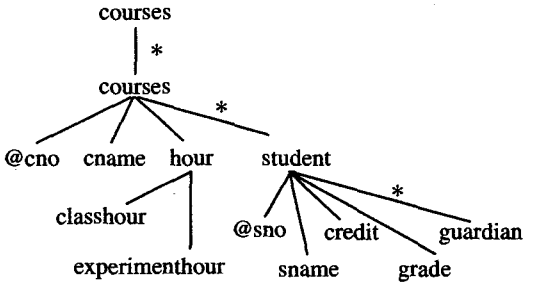

图 1 DTD $D1$ 的树结构

3 XML 键

XML 模式中约束主要是通过函数依赖来实现的，而关系中约束的主要概念是键，本文定义 XML 键的主要目的就是将 XML 模式中的约束表示规范化。XML 键的定义如下：

定义 6 XML 键，对于 DTD 中的一个元素结点，如果存在某个叶结点(叶结点一定在该元素结点的子树中)唯一地决定该元素结点，则称该叶结点为该元素结点的 XML 键。

虽然我们给出了 XML 键的定义，但对于一个 DTD 并非所有的键都是一目了然或者直接由已知约束条件得到，因此我们需要通过一个算法找到其所有的 XML 键。

算法 1 寻找 XML 键

扫描 DTD 中的根结点和每个"*"对应的结点：

(1)若该结点无子结点，则为该结点增加两个一对一的子结点@nID 和存放该结点值的 value，并标记@nID 为结点 n 的 XML 键；

(2)若该结点有子结点，则扫描以该结点为根的子树，将除去"*"对应子结点为根的子树后的所有叶结点存放在 leaves 数组中：

ⅰ. 若 leaves 数组空，则不为该结点建立 XML 键，即不为该结点生成关系表；

ⅱ. 若 leaves 数组中存在一叶结点可以唯一确定该结点，则标记该叶结点为该结点的 XML 键；

ⅲ. 否则，为该结点增加一对一的子结点@nID，并标记@nID 为该结点的 XML 键。

例 2 为例 1 中的 D1 寻找 XML 键

(1)对于 courses 结点，它是根结点，虽然它有子结点，但除去以"*"对应的子结点为根的子树后，以它为根的子树不含有叶结点，因此不为其建立 XML 键；

(2)对于"*"对应的 course 结点，与之对应的 leaves 数组中的叶结点集合是{@ cno，cname，classhour，experimenthour}，由 FD$\varphi1$ 可以知道@cno 是 course 结点的 XML 键；同理可知@sno 是 student 结点的 XML 键；

(3)对于"*"对应的 guardian 结点是 student 结点的子结点，它不包含子树，因此为其增加子结点 gID 和 gvalue，并标记 gID 为 guardian 结点的 XML 键，使其唯一确定 guardian。

4 基于推理的函数依赖扩展、消解

单纯定义的约束并不完整，因为 XML 是一种层次的结构，很多函数依赖无法通过简单的映射反映到关系表中，即使

加上隐含的约束关系，也无法避免在转换过程中依赖信息的丢失。文[7]虽然给出部分推理规则，但是并不完整，本文在文[7]的基础上分解出函数依赖的推理规则，使 XML 向关系模式转换的过程中保持语义约束传播，确保函数依赖信息不会丢失；又在文[7]中的部分规则基础上补充一些和 XML 键的约束有关的消解规则，使最终用于转换的函数依赖为极小的函数依赖集。

4.1 XML 函数依赖的推理规则

定理 1 给定 DTD $D=(E,A,P,R,r)$，对 D 上的函数依赖 $\varphi:(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_y)$，下述推理规则成立：

规则 1(唯一路径)：如果 $y\in P(last(S_{xi}))\subseteq E$，且 $y^*\notin P(last(S_{xi}))$，即子元素 y 在 $last(S_{xi})$ 结点下出现一次，或者 $y\in R(last(S_{xi}))\subseteq A$，那么函数依赖 $\varphi:(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_{xi}.y)$ 成立。

规则 2(唯一子元素)：如果满足条件 $e\in P(last(S_y))\subseteq E$，但是 $e^*\notin P(last(S_y))$，也即子元素名称 e 只能在 $last(S_y)$ 结点下出现一次，那么函数依赖 $\varphi:(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_y.e)$ 成立。

规则 3(唯一属性)：如果 $a\in R(last(S_y))\subseteq A$，那么函数依赖 $\varphi:(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_y.@a)$ 成立。

规则 4(唯一父结点) 如果满足条件 $last(S_y)\in E$，并且 $last(S_y)\neq r.S_y\text{-}last(S_y)$ 且 $S_h\subseteq_{path} r.S_y\text{-}last(S_y)$，那么函数依赖 $\varphi:(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow[S_y\text{-}last(S_y)])$ 成立。

规则 5(头部路径的包含性) 如果在 D 上 $(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_y)$ 成立，并且 $S_h\subseteq_{path}S'_h$，满足 $S'_h\subseteq_{path}S_{xi}$，$i\in[1,n]$ 和 $S'_h\subseteq_{path}S_y$，那么函数依赖 $\varphi:(S'_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_y)$ 成立。

规则 6(传递性) 如果 $(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_y)$ 和 $(S'_h,[S_y]\rightarrow S'_y)$ 成立，且满足 $S'_h\subseteq_{path}S_h$ 和 $S_h\subseteq_{path}S'_h$，那么函数依赖 $\varphi:(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S'_y)$ 成立。

规则 7(混合并规则) 如果在 D 上 $(S'_h,[S'_{x1},\cdots,S'_{xn}]\rightarrow S_h)$ 和 $(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_y)$ 成立，那么函数依赖 $\varphi:(S'_h,[S'_{x1},\cdots,S'_{xn},S_{x1},\cdots,S_{xn}]\rightarrow S_y)$ 成立。

规则 8(键-键联系规则) 对于每个 XML 键 $S_{xi}.k_i$ 约束的元素 $last(S_{xi})$，沿 S_{xi} 路径向上扫描到第一个有 XML 键约束的元素，若该元素的 XML 键为 $S_{xj}.k_j$，则添加函数依赖 $\varphi:([S_{xi}.k_i,S_{xj}.k_j]\rightarrow\varepsilon)$。

例 3 由例 1 中的函数依赖可以结合定理 1 得到扩充完整的函数依赖集，下面给出部分：

$\varphi6:$([courses. course]→courses. @cno)；//规则 1

$\varphi7:$([course. course. @cno]→courses. course. cname)；//由 $\varphi1$ 和规则 2 得到

$\varphi8:$([courses. course. @cno]→courses. course. hour)；//由 $\varphi1$ 和规则 2 得到

$\varphi9:$(courses. course,[courses. course. student. @sno]→courses. course. student. sname)；//由 $\varphi4$ 和规则 2 得到

$\varphi10:$([course. course. @ cno] → courses. course. @ cno)；//规则 3

$\varphi11:$([courses. course]→courses. course. student. credit)；//由 $\varphi9$、$\varphi2$ 和规则 6 得到

$\varphi12:$([courses. course. student. @sno, course. course. @cno]→ε)；//规则 8

$\varphi13:$([courses. course. student. guardian. @gID, courses. course. student. @sno]→ε)；//规则 8

4.2 XML 函数依赖的消解规则

定理 2 给定 DTD $D=(E,A,P,R,r)$，在 D 上的函数依赖 $\varphi:(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_y)$，下述消解规则成立：

规则 1 函数依赖 $\varphi:(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_y)$ 成立，如果 $S_y=S_h$ 或 $S_y=S_{xi}$，则消解 φ；

规则 2 函数依赖 $\varphi:(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_y)$ 成立，如果 S_h 或 S_{xi} 中均不包含 XML 键，则消解 φ；

规则 3 函数依赖 $\varphi:(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_y)$ 成立，如果 S_y 不是叶结点，则消解 φ；

规则 4 函数依赖 $\varphi1:(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_y)$ 和 $\varphi2:(S_h,[S_{x1},\cdots,S_{xi-1},S_{xi+1},\cdots,S_{xn}]\rightarrow S_y)$ 成立，则消解 $\varphi1$；

规则 5 函数依赖 $\varphi1:(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_y)$ 和 $\varphi2:([S_{x1},\cdots,S_{xn}]\rightarrow S_y)$ 成立，则消解 $\varphi1$；

规则 6 函数依赖 $\varphi1:(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_y)$ 和 $\varphi2:(S_h,[S_{x1},\cdots,S_{xi}.e,\cdots,S_{xn},S_{xn}]\rightarrow S_y)$ 成立，$e\in P(last(S_{xi}))\subseteq E$ 且 $e^*\notin P(last(S_{xi}))$，或 $e\in R(last(S_{xi}))\subseteq A$，则消解 $\varphi1$。

规则 7 函数依赖 $\varphi1:(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow\varepsilon)$ 和 $\varphi2:(S_h,[S_{x1},\cdots,S_{xn}]\rightarrow S_y)$ 成立，则消解 $\varphi1$；

例 4 对例 3 中得到的函数依赖闭包结合定理 2 可以消解重复和冗余的语义约束，下面给出部分被消解掉的函数依赖：

$\varphi10:$([courses. course. @ cno] → courses. course. @ cno)；//规则 1

$\varphi7:$([courses. course]→courses. course. cname)；//$\varphi2$ 和 $\varphi7$ 根据规则 2 可以消解 $\varphi7$

$\varphi8:$([courses. course. @cno]→courses. course. hour)；//规则 3

$\varphi9:$(courses. course,[courses. course. student. @sno]→courses. course. student. sname)；//$\varphi3$ 和 $\varphi9$ 根据规则 5 可以消解 $\varphi9$

$\varphi11:$([courses. course]→courses. course. student. credit)；//$\varphi2$ 和 $\varphi11$ 根据规则 6 可以消解 $\varphi11$

$\varphi12:$([courses. course. student. @sno, course. course. @cno]→ε)；//$\varphi5$ 和 $\varphi12$ 根据规则 7 可以消解 $\varphi12$

最终，我们可以得到一个消解后的函数依赖集 Σ：

[courses. course. @cno]→{courses. course. student, credit;
courses. course. cname;
courses. course. hour, classhour;
courses. course. hour, experimenthour}
[courses. course. student. @sno]→courses. course. student. sname}
[courses. course. @cno, courses. course. student. @sno]→{courses. course. student. grade}
[courses. course. student. guardian. gID]→
{courses. course. student. guardian. gvalue}
[courses. course. student. guardian. @gID,
courses. course. student. @sno]→{ε}

可以看出，Σ 是极小的。因为由消解规则可以得到，除去 Σ 中的任意一个函数依赖 $X\rightarrow A$，则 $\Sigma-\{X\rightarrow A\}$ 不能再推导出 $X\rightarrow A$，即不能从函数依赖集中剩下的函数依赖中通过规则推理出去除的函数依赖，因此函数依赖集 Σ 是极小函数依赖集。例如，除去 courses. course. @ cno→ courses. course. cname 则它无法从剩下的函数依赖中推导出。

5 基于键的 XML 到关系模式的转换

目前关于 XML 模式到关系模式的转换研究往往是只能考虑到结构的映射，无法完成函数依赖的映射；文[5]以层次

(下转第 138 页)

虎溪信息门户所有的业务数据都通过 Hibernate 来实现并通过 POJO 来调用。在数据库方面，通过 Hibernate 的映射关系实现与各主流数据库的完全兼容。Hibernate 类是根据模式(model)类对应生成的。这样就可以在多层系统中允许模式类是可作序列化处理的，而 Hibernate 类则不必。系统生成的持久层方法有 add，update，delete，find，remove，count。同时生成协助类(Helper classes)，可以用来调用持久层方法。协助类调用 Hibernate 的方法来对数据库进行更新操作，但是也可以改写配置文件，使用自己专用的类来完成，但这种类要求继承默认的持久层类。换言之，用户完全可以定制自己的持久层数据，可以是一个正统的数据库，或者是 LDAP 服务器或者其它。

4.2.4 SOAP，RMI，and Tunneling

客户程序可以通过传统的或者无线设备来访问门户。而开发者能够通过开放的 SOAP、RMI 以及客户隧道(tunnel)类来实现对门户的访问。

结束语 本文从信息门户的整体架构出发，提出了设计门户系统的技术手段。信息门户对各种资源进行了整合和管理，提供了统一的访问入口，从而改变了现有信息系统的访问模式。同时，信息门户还为用户提供了个性化的使用界面，使以前面向单一应用的服务模式变为面向人的服务模式，为今后信息系统的发展提出了新的开发思路。门户的构建与设计是一个比较复杂的过程。灵活运用各种技术，充分了解企业的内部需求，才能构建出合理实用的信息门户。

参 考 文 献

1 Terry Lee . Portlet 应用开发(JSR168)
2 Eamoi. Portal 架构
3 王映雪，沈陪华，陈怀楚. 清华大学信息系统结构发展策略. 中山大学学报，2001
4 BEA Portal management schema
5 Martel C, Vignollet L. Educational Web portal baeed on pereonal
6 Java Community Process JSR 168. http://www.jcp.org/about-Java/communityprocess/final/jsr168/(Accessed Sept. 22, 2004)
7 Java Technology: Java 2 Platform, Enterprise Edition (J2EE). http://java.sun.com/j2ee(Accessed Sept. 22, 2004)
8 Web Services Activity. http://www.w3.org/2002/ws/(Accessed Oct. 8, 2004)
9 Myerson J. Human-facing Web services. http://www-900.ibm.com/developerWorks/cn/webservices/ws-adapt/part1/index_eng.shtml(Accessed Oct. 8, 2004)
10 Lakos A. Personalised Library Portals and Organisational Change. http://www.lib.uwaterloo.ca/~aalakos/index.html. Mar, 2001(Accessed Oct. 11, 2004)
11 Apache JetSpeed. http://jakarta.apache.org/jetspeed/site/index.html(Accessed Oct. 8, 2004)

(上接第 97 页)

为基础，虽然同时考虑了结构和约束但是造成了大量的属性冗余。本文以 XML 键为基础，提出了一种新的转换算法，该算法既考虑到了约束关系，又可以根据第 4 节得到的极小函数依赖集转换过程中的实现规范化，避免了文[9]的分步操作，更重要的是减少了表中大量属性结点的冗余：

算法 2 基于 XML 键的 XML 模式到关系模式的规范化转换算法

ⅰ. 寻找 XML 键。利用算法 1 寻找 XML 键的方法得到 DTD 的所有键集合$\{k_1, k_2, \cdots, k_m\}$；

ⅱ. 推理。利用定理 1 的推理规则得到函数依赖的完整集合Σ'；

ⅲ. 消解。对由推理规则扩充得到的完整集合Σ'利用定理 2 的消解规则得到极小函数依赖集合Σ，该集合内包含的函数依赖形式的左部路径仅含键；

ⅳ. 归纳。对于规则推理和消解后生成的极小函数依赖集Σ，扫描每一条函数依赖，同时划分Σ为$\Sigma_1, \Sigma_2, \cdots, \Sigma_n$，$(n>m)$，分别对应左部路径的键是$\{k_1\}, \{k_2\}, \cdots, \{k_m\}$，$\{k_i \cdots k_j\}$形式的函数依赖，其中$\{k_i \cdots k_j\}$表示左部路径是由两个以上的键决定右部路径的形式；

ⅴ. 转换。对每个Σ_i进行转换得到以 XML 键约束的结点为名的关系表R_i，并把左部路径和右部路径中的每个元素结点 E 和属性 A 均映射为 R_i 中的一个属性列，再将左部路径映射的属性标记为关系表 R_i 的键。

例 5 对于例 1 给出的 XML 模式最终生成满足 3NF 的关系模式，如下所示：

Course(@cno, cname, credit, classhour, experimenthour);
Student(@sno, sname);
Guardian(@gID, gvalue);
Grade(@cno, @sno, grade);
GS(@gID, @sno);

虽然关系表 GS 中仅有两个键而没有任何约束的属性，但如果没有这张表，我们只能根据Σ中的路径还原到 DTD 结构，而无法根据关系数据库中记录还原 XML 文档，因为以 guardian 结点为根的子树在还原到 XML 文档中的时候无法确定归属于哪个 student 结点。

结论及将来工作 本文引用 DTD 和 XML 文档的定义，在此基础上将 XML 模式与关系模式表示形式进行比较，依据函数依赖的思想，在 DTD 结构中提出了 XML 键的概念，并给出算法可以找到所有的 XML 键；再改进关于 XML 函数依赖的一组逻辑推理规则，将其分为推理和消解两个部分，利用定理 1 的推理规则扩充原有的函数依赖集，弥补 XML 模式向关系模式转换过程中的语义丢失；再利用定理 2 的消解规则消除重复冗余的函数依赖，得到基于 XML 键的一组左部路径仅含键形式的极小函数依赖集；最后以键为中心划分关系，基于 XML 键得到保持语义依赖的、并且满足 3NF 规范化的关系模式。

本文转化算法的好处在于它是以 XML 键为中心划分属性，最终可以得到非常简洁的关系模式；同时将规范化在转换过程中进行，避免了文档重复扫描从而降低了时间复杂度。该算法不仅对本文的例子成立，实验证明对任何一个有效的 DTD 均成立。

本文只讨论了 XML 键在 XML 模式向关系模式规范化转换中的应用，今后还可以做的工作是将 XML 键的概念应用到 DTD 模式规范化和索引的建立的工作中，得到规范化的 XML 模式和优化的检索。

参 考 文 献

1 Arenas M, Libkin L. Symp, A normal form for XML document, Principles of Database Systems. Madison, Wisconsin, USA: ACM Press, 2002. 85～96
2 Lee M L, Ling T W, Low W L. Designing functional dependencies for XML. VIII Confrence on Extending Database Technology(EDBT), Prague, 2002
3 王元元. 计算机科学中的逻辑学. 科学出版社，1989
4 吕腾，顾宁，闫萍. XML 文档的范式. 小型微型计算机系统，2004，25(10)
5 王庆，周俊梅，等. XML 文档及其函数依赖到关系的映射. 软件学报，2003，14(7)
6 谈子敬，施伯乐. 函数依赖和规范化在关系和 XML 间的传播. 软件学报，2005，16(4)
7 吕腾，闫萍，王真星. XML 的函数依赖. 小型微型计算机系统，2005，26(5)
8 萨师煊，王珊. 数据库系统概论(第三版). 北京高等教育出版社，2000
9 Tan ZiJing, Xu JianJun, Wang Wei, Shi Baile. Storing Normailized XML Documents in Normalized Relations, CIT'05, IEEE, 2005