计算机科学1995Vol. 22 No. 3

→1NF 关系数据模型及其进展

聂培尧　褚东升

（山东财政学院信息管理系　济南250014）

摘　要　本文主要介绍→1NF 关系模型的基本概念，并讨论嵌套关系代数、嵌套范式及嵌套关系的唯一性与最小化等有关→1NF 的当前研究概况、结果与进展。

关键词　NF^2模型，嵌套关系代数，嵌套范式，唯一性，最小化。

1. 引　言

Codd 定义的关系数据库是关系的集合，其中每一关系至少要求满足第一范式（1NF），即关系中的元组必须是原子的（不可再分解的）。这种传统的关系为“平面关系（flat relation）”结构，从而限制了一些非传统的数据库应用，如 CAD/CAM 数据库、文字与图形数据库、信息检索系统以及统计数据库等。近几年来在关系模型的扩充研究方面发展十分迅速，很多研究人员提出了允许关系中的元组亦为关系的方法来对“平面”关系模型进行扩充，所得到的关系通常称为非第一范式关系（→1NF），或者嵌套关系（Nested Relation）或 NF^2关系（Non-First normal Form）。

本文将主要介绍→1NF 关系数据库模型的一些基本概念、研究的基本概况及其进展。在下一节中我们将介绍嵌套关系模型及其与之相关的数据结构和运算问题；第三节中将介绍嵌套范式问题；第四节我们将进一步介绍并讨论嵌套关系的唯一性及最小化问题。

2. 嵌套关系模型

早在1977年，Makinouchi[2]就提出了对关系模型去掉第一范式约束进行扩充的假设。Jaeschke 和 Schek[9]所扩充的普通关系模型允许其关系带有集合值（Set-valued）的属性，并且还新引进了两个构造运算符 nest 和 unnest（仅限嵌套深度为一层的嵌套关系）。Thomas 和 Fischer[6]对 Jaeschke 和 Schek 的模型进行了扩充并允许嵌套关系为任意深度（但为某一固定数）。Roth、Korth 和 Silberschatz[17]又对 Thomas 和 Fischer 的嵌套关系模型定义了类似于查询语言的演算。自那时起，对这种扩充后的嵌入关系模型又引进了大量的类 SQL 查询语言[18,19]、面向图形的查询语言[20]以及类 Datalog 语言[21,22]。同样，很多人[12,14,23,24,25]开始实现了嵌套式关系数据库模型，有些人以现存的传统关系 DBMS 为基础并在其上进行扩充以实现其嵌套式关系 DBMS；另一些人则是实现一个全新的嵌套式 DBMS。

2.1 数据结构问题

在不考虑关系必须满足第一范式的条件时可对关系模型进行多种形式的扩充。有些方案[13,26]可以允许关系的属性为任意域。嵌套关系模型的思想则是允许在其属性出现的任何地方出现关系。下面给出了利用类 PASCAL 语言描述的嵌套关系，其中在属性 attri-name$_i$ 上的嵌套深度可以任意。

```
type nested-relation-type=
    SET OF RECORD
    ┆
    attri-name1:atomic-type1;
    ┆
    attri-namei:SET OF RECORD
        ┆
        END;
    ┆
    END;
```

文[27]中给出了嵌套关系的形式定义，这里不再详述。形式化嵌套关系的一个关键问题是关系的两部分视图，即模式和值。通常模式的形式由一关系名后跟用括号括起来的若干属性列表构成。嵌套关系模式也可以用树来表示，其值可以用嵌套表来表示。下面图1中给出了一简单的嵌套关系的模式树及值表，其线性表表示为：

D_{EPT}（DNO，DNAME，BUDGET，E_{MP}（ENO，ENAME，SALARY））

聂培尧　副教授；副系主任，主要从事数据库系统及 MIS 的研究。

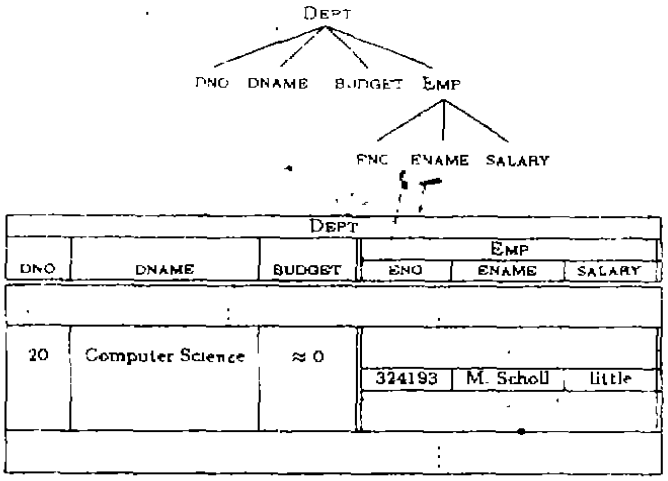

图1

2.2 操作问题

在设计嵌套关系操作语言时，一般是：当在语言中某处遇到一关系时，应允许在该语言中使用查询表达式。本文不准备研究诸如嵌套关系 SQL[18,19,28] 那样的"高级语言"，而主要介绍对关系代数进行扩充以适用于嵌套关系操作的问题。

2.2.1 "平面"关系代数 传统的"平面"关系代数由五种基本运算符构成，即并(∪)、差(\)、关系积(×)、投影(π)，选择(σ)，并具有使用这些基本运算符组合表达式的能力。以下的递归规则定义了"平面"关系的关系代数：

(1)若 R 是一关系(名)，则 R 为一代数表达式；

(2)若 E_1、E_2为代数表达式，则 $E_1 \cup E_2$，$E_1 \setminus E_2$及 $E_1 \times E_2$也为代数表达式；

(3)若 E 是一代数表达式，则 π[L](E)和 σ[F](E)也为代数表达式，这里的 L 为包含在 E 的模式中的属性名的投影列表；F 为选择公式，其构成为由 E 的模式中的属性和常数经过逻辑连接符(∧，∨，¬)及算术比较符构成。

2.2.2 NEST 与 UNNEST 操作 NEST 和 UNNEST 这两个运算符最早由文[8,9]中引入，记为 υ 和 μ，用以处理"平面"关系与嵌套关系之间的相互变换。嵌套(NEST)的作用类似于扩充 SQL 中的"GROUP BY"子句。嵌套的结果可由非嵌套(UNNEST)消除，UNNEST 操作总是 NEST 的逆操作，但一般情况下其逆运算并不成立[8,9]。下面为 NEST 和 UNNEST 的一个例子：

$N_{ESTED}D_{EPT}$(DNO，DNAME，BUDGET，E_{MP}(ENO，ENAME，SALARY))：=
υ[(ENO，ENAME，SALARY)=E_{MP}]
(F_{LAT} D_{EPT}(ENO，ENAME，SALARY，DNO，DNAME，BUDGET))
$F_{LAT}D_{EPT}$(ENO，ENAME，SALARY，DNO，DNAME，BUDGET)：=
μ[E_{MP}](N_{ESTED} D_{EPT}(DNO，DNAME，BUDGET，E_{MP}(ENO，ENAME，SALARY))

应当指出的是，除了分解(投影)操作外，嵌套是在数据库设计期间获得"好"的关系模式的一种途径，它总是在不断地消除冗余。这也就是为什么嵌套关系被认为是最适合于按某些范式对数据库进行模式设计[3,6]的原因所在。

2.2.3 嵌套关系代数 嵌套关系可以认为是一类层次结构[7]，与纯层次结构模型的区别在于：在纯层次结构中，原子属性的组合构成该结构每一层的关键字，但这种约束在一般的嵌套结构中是不存在的。

对于嵌套关系的数据结构，可以允许在关系的属性上嵌套关系，同样，也可以在代数表达式中嵌套代数表达式以替代属性。这样可得到双重嵌套代数。第一嵌套为通常由函数分解实现的；第二种嵌套为一种新的方法，是在"平面"关系中任何出现属性的地方，即在投影列表和选择公式中，可以使用代数表达式。

为了把"平面"关系代数扩充到嵌套关系代数，可先对"平面"关系代数作以下两点的修改：

(1)对"平面"关系代数中的投影运算，如 π[L](E)中的投影列表 L 进行扩充使其可以包含代数表达式，用 $N_i:=E_i$ 来替代原来的属性名，投影得到的子关系 N_i 定义为取对 E_i 赋值的结果值；

(2)对"平面"关系代数的选择运算 σ[F](E)中的选择公式 F 进行扩充使其可包含集合比较谓词 Θ 及代数表达式 $E_i \Theta E_j$。

经过以上修改，再增加两个重构运算符 υ 和 μ，即可把"平面"关系代数扩充为嵌套结构的关系代数。因此，扩充后所得到的嵌套代数可由如下七种运算符构成：

1)并($\cup^e$)：应保证经该操作所得到的结果结构为层次的；

2)差($-^e$)：与并运算具有相同的要求；

3)投影(π^e)：扩充后的要求如同(1)；

4)选择(σ^e)：虽然非常类似于"平面"关系模型中的运算，但在嵌套关系模型中具有更强的表达能力，并且可在结构的任何层次上执行。为了方便，文[36]中提出了用于过滤元组的模板(template)。除了用于比较原子值的标准条件运算符(=，≠，>，<，≥，<，≤)外，如前扩充(2)所述，还需要用于比较集合以及集合元素的集合比较谓词 Θ={∈，∉，⊂，⊄，⊆，⊈}；

5)联结($\bowtie^e$)：实现两个结构上的主元属性的自然联结和条件联结，既可实现原子值属性间的联结，也可以实现集合值属性的联结；

6)NEST(υ)：在嵌套关系结构中的任意层次上产生集合值的属性。使用嵌套运算符 υ 和联结运算符还可以在一嵌套关系模型中产生多个集合值的属

性；

7)UNNEST(μ)：为υ的逆操作，可解除由υ产生的集合值属性。

下面我们给出几个使用嵌套代数表达式进行查询的例子：

(1)给出名为"Computer Science"的那些部门，要求在这些部门中至少有一个雇员的薪水大于30000：

σ^{e}[DNAME="Comp. Sc. "∧ σ^{e}[SALARY>30000](E_{MP})≠∅](D_{EPT})

(2)给出名为"Computer Science"的那些部门，要求在这些部门中全体雇员的薪水均大于30000，称为"高薪雇员"，用RE表示：

π^{e}[DNO,DNAME,BUDGET,RE:=σ^{e}[SALARY>30000](E_{MP})](σ^{e}[DNAME="Comp. Sc. "](D_{EPT}))

(3)查询内容同前，但若无雇员其薪水超过30000，则去掉"CS"部门的元组：

σ^{e}[RE≠∅](π^{e}[DNO,DNAME,BUDGET,RE:=σ^{e}[SALARY>30000](E_{MP})](σ^{e}[DNAME="Comp. Sc. "](D_{EPT}))

注意，该查询中的附加选择("σ^{e}[RE≠∅]")与内层选择的结果有关。

应当指出的是，上面(3)中的查询与以下的平面关系的选择-投影-联结查询相等价：

σ[DNAME="Comp. Sc. "](D_{EPT})⋈σ[SALARY>30000](E_{MP})

在(3)中，当执行联结运算时，对那些没有雇员其薪水超过30000的部门将自动被消除。另外，等价于(2)中的嵌套关系查询的"平面"关系表达式可对自然联结代之以外层联结获得。以上的例子说明了在典型的嵌套关系代数查询中，在投影或选择运算内可以嵌套任意的表达式，但应特别注意这种"内部"运算中如何定义有效变量的问题[36]。

在嵌套很深的代数表达式中使用高层属性(在模式树中)的情况在文[27]中称为"动态常量"。这一概念已被证明是非常有用的，因为嵌套、差、关系积和联结等操作可仅仅使用带有动态常量的嵌套选择和投影定义之。例如，对每一部门元组，从子关系E_{MP}中选择出那些经理的子元组(假设在D_{EPT}中其属性名为MNO)：

π^{e}[DNO,DNAME,σ^{e}[ENO=MNO](E_{MP})](D_{EPT})

这里MNO是施加于E_{MP}的内部选择的"动态常量"。

也可以按照类似于查询代数的表示法对嵌套关系定义其修改运算(Update)，当然这种运算也是嵌套的。例如，可以根据插入一元组到一子关系中来改变一个元组。这样，代数对嵌套关系的层次结构提供了一强有力的操作集。为了对嵌套关系层次中的更深层的子关系进行运算，其相应的代数表达式可嵌套到投影或选择运算的内部。特别是在这种情况下使用"平面"关系代数操作和重新嵌套操作可避免对关系的UNNEST操作。因此，这种嵌套关系代数也可以作为一高效的DBMS内核的操作接口来提供嵌套关系的存贮结构。

3. 嵌套关系范式

Özsoyoglu和Yuan于1985年在文[3]中给出了第一个→1NF关系规范化的综合算法。他们研究了其模式为树状结构的嵌套关系，称为模式树，并对这类关系引进了一种范式，称为嵌套范式NNF(Nested Normal Form)。一模式树是这样一种树，其树中的结点标以非嵌套属性的互不相交的集合；而树的边则表示树中结点属性间的多值依赖(MVDs)。可用以把一个1NF关系表示成为如上所讨论的具有所期望性质的→1NF关系。图2给出了E_{MP}模式的模式树及相关的MVDs。

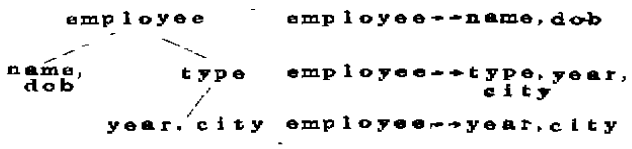

图2

下面形式地讨论模式树。令U为一嵌套属性的集合，令T为一模式树并令S(T)表示T中的所有属性的集合。S(T)为U的子集，令e=(u,v)为T中一条边，则模式树T的每一边表示了S(T)上的一多值依赖。这种由边e所表示的多值依赖为：A(u)→→D(v)，其中A(u)为u的所有前驱(包括u)的并；D(v)为v的所有后继(包括v)的并。由于可能出现S(T)为U的真子集，此时由T的一条边所表示的多值依赖可能为U上的一嵌套多值依赖(EMVD)。我们记由T的边所表示的多值依赖的集合为MVD(T)。

定义3.1 令T为一模式树，$u_1,u_2,\cdots,u_n$为T的全部叶结点，则T的路径集合，记为P(T)，为{A(u_1),A(u_2),…,A(u_n)}。

注意：对于每一叶结点u，A(u)为从T的根结点到T的叶结点u的路径中的全部结点的并。

下面给出了一模式树的某些性质：

性质3.1 若T为一模式树，则1)P(T)为一非循环数据库模式；2)多值依赖的集合MVD(T)与联结依赖⋈(P(T))等价；3)MVD(T)为MVDs的一无冲突集[6]。

令T为关于M的一模式树，这里S(T)⊆U且(u,v)为T中的一条边并且M为MVDs的集合。假设存在一M的关键字X使得存在一Z∈DEP(X)及D(v)=Z∩S(T)，则：如果X⊂A(u)，则称v为关于

X 的 T 中的部分冗余，而 S(T)的上下文中的 MVD X→→D(v)为 S(T)中的一部分依赖。同样，若 T 中存在 v 的某些兄弟结点 $v_1, v_2, \cdots, v_n$ 使得 $W=\bigcup_{i=1}^{n} D(v_i)$，$X \subset A(u) \cup W$，并且 M 在 S(T)的上下文中不蕴涵 X W→→D(v)，则称 v 为 T 中关于 X 的传递冗余，在这种情况下 S(T)的上下文中的 MVD X→→D(v)则称为 S(T)中的传递依赖。注意，上面用到的集合 DEP(X)称为 X 的依赖基[6]。

在嵌套模式的设计过程中，对出现的部分依赖和传递依赖必须从模式树中消除。在最终的设计结果中，应使每一(u,v)的边表示 A(u)→→D(v)。若存在部分依赖，则对某些 u 则具有 A(u)的一真子集 LHS，这样边(u,v)就不能表示那些(部分)依赖，这里的集合 LHS 为 u 的基本关键字集，事实上，部分依赖确实蕴涵了由(u,v)所表示的依赖，我们可以把树分裂成两个或更多棵树的方法来消除部分依赖。若 T 中存在有传递依赖，则在 T 的属性上就存在一多值依赖，而该多值依赖不遵循 T 的某些子树的边所表示的 MVDs。

为了避免对树中关键字的划分，我们引入一基本关键字的概念，令 M 为 U 上的一 MVDs 的集合，$V \subseteq U$，则 V 上的基本关键字集合，记为 FK(V)，定义为：

FK(V)={V∩X | X∈LHS(M)[6]，V∩X≠∅且不存在 Y∈LHS(M)使得 V∩X⊃V∩Y 及 V∩Y≠∅}。

直观地，V 上的基本关键字集合包含了那些最小化的，与 V 相交为非空的那些关键字。

我们希望在设计模式树时使每一结点均为它的子树的基本关键字，因为在文[3]中证明了这样一种树其路径集合是一种第四范式设计，给定属性 U 上的 MVDs 的集合 M，文[3]中给出了一种把 U 分解到不存在部分冗余或传递冗余，并且在树中的结点上不再分裂关键字的一个1NF 模式树的集合的算法。形式地，一个规范模式树可定义如下：

定义3.2 一模式树 T 称为关于 MVDs 集合 M 规范的，如果：(a)M 蕴涵 MVD(T)；(b)T 中不存在部分依赖；(c)T 中不存在传递依赖；(d)T 的根为一关键字且 T 中的每一其它结点 u，若 FK(D(u))≠∅，则 u∈FK(D(u))。

文[3]使用了分解技术构造一个 NNF 设计。当一路径集合不属于4NF 时，可按如下方式进行分解，即：在一结点 V 上选择一基本关键字 K 并把 V 分裂成一棵子树，该子树以 K 为其根，以 K 的基本依赖基中的每一元素为 K 的孩子，重复这一分解过程直至没有进一步的非4NF 路径集合。该方法使用了 MVDs 和 FDs 作为 NNF 分解算法的输入。在文[29]中，作者把 FDs 和 MVDs 组合成一个依赖的包封集合(envelop set)，可以作为一稍加修改以考虑 FDs 和 MVDs 不同语义的 NNF 算法的输入。使用文[3]中的算法，当使用 FDs 执行分解算法时很可能出现单条集合。M. A. Roth 和 H. F. Korth[6]提出了一种考虑 FDs 不同语义以获得嵌套范式 NNF 的新方法，由于篇幅所限，关于嵌套范式设计问题的举例及嵌套关系的分解算法，有兴趣的读者可参见文[4,6]。

4. 嵌套关系的最小化及唯一性问题

Koichi Takeda 于1987年在文[5]中对最小化问题和唯一性问题进行了较深入的研究，因此，在下面的讨论中我们将主要介绍并引用文[5]中的思想及结果。

先来看一下建立在 Department (DEPT)、Course (COURSE) 和 Programming Language (LANGUAGE)上的一简单的1NF 关系 r，如图3所示。

DEPT	COURSE	LANGUAGE
CS	AI	Lisp
CS	AI	Smalltalk
CS	AI	Prolog
CS	Programming	Lisp
CS	Programming	Smalltalk
CS	Programming	Prolog
CS	Programming	C
CS	Programming	PL/I
CS	Database	Prolog
CS	Database	C
CS	Database	PL/I

图3

在图3的关系中，若允许其课程(Course)和程序设计语言(Programming language)分别为一集合作为属性 COURSES 和 LANGUAGES 的域值，则可得到如图4所示的嵌套关系。

DEPT	COURSES	LANGUAGES
CS	{AI}	{Lisp,Smalltalk,Prolog}
CS	{Programming}	{Lisp,Smalltalk,Prolog,PL/I}
CS	{Database}	{Prolog,C,PL/I}

DEPT	COURSES	LANGUAGES
CS	{AI,Programming}	{Lisp,Smalltalk}
		{Prolog}
CS	{AI,Programming,Database}	{C,PL/I}
CS	{Programming,Database}	

图4

显然，图4中的两个嵌套关系均为由图3中的1NF 关系 r 所产生的两个可能的嵌套关系形式。但是，使用文[9]中的 NEST 操作，可以验证这两个关系分别是由 $NEST_{COURSE}(NEST_{LANGUAGE}(r))$ $NEST_{LANGUAGE}(NEST_{COURSE}(r))$ 操作所产生。

现在的问题是，图4中的两个嵌套关系能否认为是"相同的"。对于集合的值的解释存在两种不同的假设，第一种假设[2]认为值的每一集合都是有意义

的。这样，图4中的两个关系则具有不同的意义。因为它们表示了值的不同集合。这种假设是基于每一嵌套关系被认为是一数据的唯一表示这样一种思想的，但是，这种假设存在着缺点，即当对其归范到一简单的事实时有可能丢失值的集合的信息。例如，若我们把 LANGUAGEs 的集合认为是一有意义的对象，则：

"AI Course aims at building an intelligent system using a set of Languages: Lisp, Smalltalk, and Prolog"将显然不同于以下三种事实的集合：

"AI Course aims at…using Lisp"，

"AI Course aims at…using Smalltalk"以及

"AI Course aims at…using Prolog"。

这样，我们将很难区别嵌套关系与1NF 关系，除非对1NF 关系增加一人工属性以对1NF 关系的集合进行标识。在这种意义下，虽然第一种假设可以使嵌套关系以一种自然的方法表示复杂对象，但却存在着1NF 与嵌套型关系间的语义隔阂。与其相反，第二种方法则假设集的每一集合仅为其单值的并，而不是一具体的对象，这样就可以对图4中的两个关系进行识别并可与图3中的1NF 关系识别之。事实上，第二种假设在很多研究的文章中均为隐含的假设，如在1NF 与嵌套型关系间的变换[1,9,30]、设计嵌套关系[31]以及数据操纵[32]等文献中。

我们将沿用第二种假设，首先关心的问题是在其表示的变体中定义一规范的关系并讨论其唯一性。某些研究者已经提出了在 NEST 操作的可约性[32]和可置换性[31]意义上的规范嵌套关系的概念。根据对1NF 关系进行简单的、但均匀的变换可使这些规范的嵌套关系的冗余减至最小(或元组数量)。最小化嵌套关系非常重要，因为它常用来实现存贮优化、良好的用户接口以及某些基于规则的系统的最优实现。因此，我们将主要介绍最小化嵌套关系作为一类规范的嵌套关系所具有的一些基本性质。

在讨论最小化嵌套关系问题时相应于图3中给出的1NF 关系的最小化嵌套关系具有两个元组，如图5所示。

DEPT	COURSES	LANGUAGES
CS	{AI, Programming}	{Lisp, Smalltalk, Prolog}
CS	{Database, Programming}	{Prolog, C, PL/I}

图5

4.1 一些基本概念

首先，考虑两类域：简单域和集合值域，后者为某一简单域 D 的幂集2^D。为了避免空值的标识问题，简单域中不允许空值出现，并且从每一集合值域中排除掉空集$\emptyset$。为了简单起见，我们将不讨论由域的笛卡尔积的幂集构成的关系值，但以下讨论的大多数结论对关系值域仍将有效。

一属性 A 称为简单属性，是指它与一简单域相关，否则称为集合值属性。记一含有 m 个简单属性 $A_1, A_2, \cdots, A_m$ 和 n 个集合值属性 $B_1, B_2, \cdots, B_n$ 的关系模式为：$R(A_1, A_2, \cdots, A_m, B_1, B_2, \cdots, B_n)$，若在其属性已经明确的情况下可简记 R。R 上的嵌套关系记为 r。我们用大写字母 A，B 和 C 表示属性，R 和 S 表示关系模式；用小写字母 r 和 s(可带下标)表示关系。在以下的论述中，对任意的 i，域2^D上的集合值属性 B_i 的相应于域 D 上的简单属性为 C_i。

对于前面 R 上关系 r 中的每一元组 t，若 $A_1A_2\cdots A_mC_1C_2\cdots C_n$ 上的一元组 t_s 使得 $t_s(A_1)=t(A_1)$，$\cdots, t_s(A_m)=t(A_m)$，并且 $t_s(C_1)\in t(B_1), \cdots, t_s(C_n)\in t(B_n)$，则称 t_s 为 t 的一简单元组。这里 t(A)表示属性 A 上 t 的值。$A_1A_2\cdots A_mC_1C_2\cdots C_n$ 上的关系 r_s 称为 r 的1NF 关系，记为 NOR(r)，是指对于 r 的每一元组，r_s 为所有简单元组的集合。

称两个嵌套关系 r_1 和 r_2 为1NF 等价的，记 $r_1 =_{1NF} r_2$，是指 $NOR(r_1)=NOR(r_2)$。R 上的一嵌套关系 r 称为最小化的，是指不存在嵌套关系 s 使得 $r =_{1NF} s$ 且 s 比 r 的元组数少。一般来说，对一给定的1NF 关系，可能存在多个最小化嵌套关系。一个最小化嵌套关系 r 是唯一的，如果不存在其它的与1NF 等价的最小化嵌套关系。

显然，如果1NF 关系上的两个元组 t_1 和 t_2 至少在 r 的一个简单属性上不相同，则 t_1 和 t_2 均不能成为一嵌套关系 r 的某些元组 t 的简单元组。因此，不失一般性，假设 A_i 的每一值在最小化关系的上下文中为一常量，则任何最小化嵌套关系可根据对其简单属性的值为常量的那些最小化嵌套关系进行并运算而得到。

4.2 复杂性问题

求最小化嵌套关系的复杂性问题可描述为以下的判定问题：

最小化嵌套关系；

情况：给定 $R(A_1, A_2, \cdots, A_m, C_1, C_2, \cdots C_n)$ 上的一1NF 关系 r 和一正整数 K；

问题：是否存在 $S(A_1A_2\cdots A_mB_1B_2\cdots B_n)$ 上的一嵌套关系 s 使得 NOR(s)=r，并且 s 最多具有 K 个元组？

这是一个 NP 问题，因为在确信了每一子集可映射到一嵌套关系的单元组上之后，仅能推测 K 值或者比 r 更小的子集并在多项式时间内检验每一 r 的元组至少包含在一个子集之中。

定理4.1 最小化嵌套关系问题为 NP 完全的。

证明见文[5]。

推论4.1 最小化嵌套关系为 NP 完全的，即使 $C_i(i=1,2,\cdots,n)$ 的每一简单域恰好含有两个元素也是如此。

推论4.1根据定理4.1的证明立即可得。

由以上定理及推论我们可以得出这样一个结

论，即对于一给定的1NF关系，不可能找到一种求最小化嵌套关系的有效算法。

4.3 最小化嵌套关系及变换问题

下面讨论由1NF关系构造嵌套关系的三种类型的变换运算及X-可最小化的概念，这里的X为将要介绍的三种运算之一。为了避免简单属性和集合值属性的混淆，假设存在一种映射，能把1NF映射到嵌套型关系，即对任意的$A_1A_2\cdots A_mC_1C_2\cdots C_n$上的1NF关系r均被映射到$A_1A_2\cdots A_mB_1B_2\cdots B_n$上的一嵌套关系s上。这里r的每一元组$t=\langle a_1,a_2,\cdots,a_m,c_1,c_2,\cdots,c_n\rangle$被映射到s的元组$u=\langle a_1,a_2,\cdots,a_m,\{c_1\},\{c_2\},\cdots,\{c_n\}\rangle$上。对s以及所得到的嵌套关系可相继使用以下三种类型的运算：

1. $NEST_{B_i}(r)(i=1,2,\cdots,n)$：把每一元组$(t_1,\cdots,t_k)$的最小集映射到一元组t使得对每一属性X，有：

$$t(X)=\begin{cases}t_1(X) & \text{若 } X\neq B_i\\ \bigcup_{j=1}^{k}t_j(B_i) & \text{其它}\end{cases}$$

2. $TWU(t_1,t_2,r)$：为一元组并运算，它取r的两个元组t_1和$t_2(t_1\neq t_2)$形成一个元组t，使得对每一属性X，有：

$$t(X)=\begin{cases}t_1(X) & \text{若 X 为一简单属性}\\ t_1(X)\cup t_2(X) & \text{若 X 为一集合值属性}\end{cases}$$

注意：两个元组t_1和t_2欲保证$TWU(t_1,t_2,r)$的结果不损失信息，如果除X集合值属性外，对任意属性X，或者$t_1(X)=t_2(X)$；或者$t_1(X)\subseteq t_2(X)$（或对任意X，$t_2(X)\subseteq t_1(X)$），这里的简单属性$A_1,A_2,\cdots,A_m$为常数。

3. $CTWU(t_1,t_2,r)$：为一元组并运算的副本，它与$TWU(t_1,t_2,r)$的区别在于：除了进行两个元组的并运算外还保留了t_1，即它产生t_1的冗余表示。

现在考虑可最小化的概念。一个1NF关系称为NEST-可最小化的，如果对其经过NEST操作序列后至少可得到一个最小化嵌套关系。同样，一个1NF关系称为TWU-可最小化（(CTWU+TWU)-可最小化），如果对其经过一TWU操作（CTWU和TWU操作）序列后至少得到它的一个最小化嵌套关系。

引理4.1 任何NEST运算（操作）均可由一TWU操作序列所替代。

由NEST和TWU运算的定义，显然，对经过NEST操作后映射到s的元组t上的每一元组$t_1,\cdots,t_k$的集合，存在一$k-1$次的TWU操作序列$TWU(t_1,t_2,s),TWU(t',t_3,s'),\cdots,TWU(t^{k-2},t_k,s^{k-2})$，这里$t'$和$s'$分别表示由第j次TWU操作所得到的一元组及关系。特别地，若$k=1$，我们显然可以使用TWU操作的空序列替代。

定理4.2[5] 如果一个1NF关系为NEST-可最小化的，则也是TWU-可最小化的。

要特别指出的是，以上引理4.1及定理4.2的逆定理一般不成立。例如，下面图6所示的嵌套关系s并不是由NEST，而是由TWU运算所得到。因为s为唯一的最小化关系，而NOR(s)为TWU-可最小化的但不是NEST-可最小化的。应当注意的是，某些1NF关系为NEST-可最小化的，但它们却存在不能由NEST-可最小化操作所得到的最小化嵌套关系。例如，若我们从前面图3所示的1NF关系中消除全部COURSE="Database"的元组，则可得到图7所示的最小化嵌套关系，而该关系并不能由NEST操作序列所得到，虽然它是NEST-可最小化的。

DEDT	COURSES	LANGUAGES
CS	{AI,Programming}	{Lisp}
CS	{Database,Programming}	{C,PL/I}
CS	{OS}	{C,Modula-2}

图6

DEPT	COURSES	LANGUAGES
CS	{AI,Programming}	{Lisp,Smalltalk,Prolog}
CS	{Programming}	{Prolog,C,PL/I}

图7 带有冗余的最小化嵌套关系

最小化嵌套关系可分为两类：冗余或非冗余。一个嵌套关系r称为是冗余的，如果存在一由r中两个元组t_1和$t_2(t_1\neq t_2)$所归类的一简单元组t，否则r称为非冗余的。

定理4.3[5] 一个1NF关系具有一非冗余的最小化嵌套关系，当且仅当它是TWU-可最小化的。

证明 因为一TWU操作并不能引入冗余，故定理的当部分是显然的。令一非冗余最小化嵌套关系$r=\{t_1,t_2,\cdots,t_k\}$并且t_i为$\langle a_1,a_2,\cdots,a_m,b_{i,1},b_{i,2},\cdots,b_{i,n}\rangle,(i=1,2,\cdots,k)$，则元组$t_i$可表示成$|b_{i,1}|$元组的并：

$$\begin{aligned}&\langle a_1,a_2,\cdots,a_m,\{c_{i,1}\},b_{i,2},\cdots,b_{i,n}\rangle,\\&\langle a_1,a_2,\cdots,a_m,\{c_{i,2}\},b_{i,2},\cdots,b_{i,n}\rangle,\\&\quad\vdots\\&\langle a_1,a_2,\cdots,a_m,\{c_{i,j}\},b_{i,2},\cdots,b_{i,n}\rangle\end{aligned}$$

其中$j=1,2,\cdots,|b_{i,1}|$且$|b_{i,1}|$表示$b_{i,1}=\{c_{i,1},c_{i,2},\cdots,c_{i,v}\}$的基数。同样，上面的每一$|b_{i,1}|$元组又可依次表示为$|b_{i,2}|$元组的并。重复该过程几次直至每一元组在其全部集合值属性上具有单值为止。一旦得到这种元组的集合，可很容易地构造一TWU操作的序列seq_i来创建t_i，因为关系r是不冗余的，则得到r的整个操作序列为$seq_1,seq_2,\cdots,seq_k$。因此，TWU-可最小化得证。

对以上的证明过程稍加修改，可得到如下结果。

定理4.4 一个1NF总是(CTWU+TWU)-可最小化的。任何最小化嵌套关系均可由一CTWU和TWU操作序列所得到。

4.4 依赖及最小化嵌套关系问题

目前对嵌套关系的各种依赖问题已进行了很多研究。Jaeschke和Schek[9]、Fischer和Van Gucht[34]

都证明了 NEST 运算的置换性可被严格定义成满足1NF 关系的一弱多值依赖的集合。Kambayashi，Tanaka 和 Takeda[30]讨论了在1NF 关系中成立的函数依赖、多值依赖和联结依赖间的联系及在嵌套关系中的特殊形式问题。Ozsoyoglu 和 Yuan[31]考虑了多值依赖，Roth[6]考虑了用以设计嵌套关系模式的函数和多值依赖问题。文[35]中对嵌套关系的运算与依赖的相互联系、文[32]对使用依赖的嵌套关系的性质也进行了讨论。

在这里将介绍关于另一类函数依赖(即样板依赖[36])的研究，这种依赖可以对一类1NF 关系进行描述，使得该类中的每一关系具有一唯一的嵌套关系且为 NEST-可最小化的。下面考虑图8中1NF 关系模式 $R(A_1,A_2,\cdots,A_m,C_1,C_2,\cdots,C_n)$ 上的依赖：

A_1	A_2	…	A_m	C_1	C_2	…	C_n
a_1	a_2	…	a_m	c_1	c_2	…	c_n
a_1	a_2	…	a_m	x_1	c_2	…	c_n
a_1	a_2	…	a_m	c_1	x_2	…	c_n
⋮			⋮	⋮			⋮
a_1	a_2	…	a_m	c_1	c_2	…	x_n
a_1	a_2	…	a_m	x_1	x_2	…	x_n

图8

这里的第一行为属性，两条横线中间的 n－1行称为假设，最后一行称为结论，带下角标的 a，c 和 x 均为变量。称 r 满足样板依赖，是指对于每一个使得 n＋1个假设行被映射到 R 中关系 r 的元组上的、从以上变量到属性值的映射，r 必定存在从结论行映射来的另一元组。图9所示的关系在三个属性上，即 A_1＝PROFESSOR、C_1＝STUDENT 及 C_2＝ROOM，满足 T(T 表示样板依赖)。

引理4.2 令 U 为属性 $A_1\cdots A_mC_1\cdots C_n$ 的一集合，则上面讨论的样板依赖 T 等价于弱多值依赖 $\{(U-C_iC_j)-w\rightarrow C_i\}$ 的集合 W，这里 $U-C_iC_j$ 表示集合的差，且 C_iC_j 为 $C_i\cup C_j$ 的简单记法，$1\leqslant i,j\leqslant n(i\neq j)$。

PROFESSOR	STUDENT	ROOM
P. Y. NIE	L. L. ZHOU	201
P. Y. NIE	J. M. GAO	101
P. Y. NIE	J. M. GAO	102
P. Y. NIE	L. K. WU	101
P. Y. NIE	L. K. WU	102

PROFESSOR	STUDENTS	ROOMS
P. Y. NIE	{L. L. ZHOU}	{201}
P. Y. NIE	{J. M. GAO, L. K. WU}	{101, 102}

图9 满足样板依赖 T 的1NF 关系及相应的嵌套关系

证明 每一弱多值依赖 $(U-C_iC_j)-w\rightarrow C_i$ 可表示为一具有三个假设行和一个结论行的样板依赖。例如，一弱多值依赖 $(U-C_1C_2)-w\rightarrow C_1$ 为以下样板依赖：

A_1	…	A_m	C_1	C_2	C_3	…	C_n
a_1	…	a_m	c_1	c_2	c_3	…	c_n
a_1	…	a_m	x_1	c_2	c_3	…	c_n
a_1	…	a_m	c_1	x_2	c_3	…	c_n
a_1	…	a_m	x_1	x_2	c_3	…	c_n

根据样板依赖的定义，很容易验证 T 蕴涵这一样板依赖。同样，W 中的每一成员均被 T 所蕴涵。要想证明 W 蕴涵 T，可以使用文[36]中的 Chase 过程，由于篇幅所限，这里就不详述了。

下面关于 NEST-可置换性的引理由 Fischer 和 Van Gucht[34]及上面的引理4.2直接可得。

引理4.3 一个 $R(A_1,A_2,\cdots,A_m,C_1,C_2,\cdots,C_n)$ 上的1NF 关系根据对 $B_1,B_2,\cdots,B_n$ 使用 n 次 NEST 置换操作可变换到其嵌套关系，当且仅当 r 满足样板依赖 T。

定理4.5 一个 $R(A_1,A_2,\cdots,A_m,C_1,C_2,\cdots,C_n)$ 上的1NF 关系 r，如果满足样板依赖 T，则经过对 $B_1,B_2,\cdots,B_n$ 的 n 次 NEST 操作序列可得到一个非冗余的唯一最小化嵌套关系。

定理4.5的证明见文[5]。

推论4.2 一个1NF 关系 r 是 NEST-可最小化的，如果 r 满足 T。

因为 NEST-可置换性可在多项式时间内验证[33]，则一个1NF 关系 r 是满足 T 的，并且也可以在多项式时间内验证。

但遗憾的是，样板依赖 T 对具有唯一最小化嵌套关系来说不是必要条件。图10给出了在 C_1＝STUDENT 和 C_2＝ROOM 上违反 T 的、但具有唯一最小化嵌套关系的例子。进一步，因为 T 是两个属性上的弱样板依赖[37]，因而1NF 关系不满足非平凡样板依赖，这样我们可得到以下定理。

STUDENT	ROOM
L. L. ZHOU	201
J. M. GAO	201
J. M. GAO	101
L. K. WU	101

STUDENTS	ROOMS
{L. L. ZHOU, J. M. GAO}	{201}
{J. M. GAO, L. K. WU}	{101}

图10

理4.6[5] 具有非冗余唯一最小嵌套关系的一类1NF 关系 G 不能由样板依赖确定；NEST-可置换的1NF 关系类恰好由类 G 所包含。(参考文献共38篇略)