从函数依赖综合第三范式关系

菲利普·伯恩斯坦

多伦多大学

摘　要：有人提出，关系数据库的描述可以表述为数据库属性之间的一组函数关系。然后可以使用这些函数关系通过算法合成关系模式。本文的目的是提出一种进行此类合成的有效程序。该过程产生的模式被证明是 Codd 的第三范式，并且包含尽可能少的关系。还讨论了早期尝试构建此类程序的问题。

关键词：数据库模式、函数依赖、关系模型、数据语义、第三范式

1　简介

对关系数据库模型的研究表明，当人们考虑如何将属性分组为关系时，函数关系是一个重要的概念[3, 8-12]，有人提出可以制定数据库的基本描述纯粹作为一组这样的函数关系，可以通过算法合成关系模式 [3, 12]。本文的目的是开发一种可证明合理且有效的程序，用于从一组给定的函数关系中合成满足科德第三范式的关系。此外，我们的过程合成的模式将显示包含最少数量的关系。

该方法假设最多存在一个将任何一组属性连接到另一组属性的函数关系。所有早期方法都需要这种唯一性假设，它提出了将详细讨论的困难语义问题。

前三节介绍了从函数依赖性合成关系的问题。本文的第 2 节回顾了关系模型、函数依赖的概念和 Codd 范式。引入了两个新概念：“超级密钥”和“体现”。第 3 节概述了一般综合问题，并提出了一种从函数依赖性综合关系的简单算法。第四节介绍了阿姆斯特朗的函数依赖公理化以及对该理论的唯一性假设和其他语义考虑的评论。主要的综合算法在第5节中描述。第 6 节检查了应用于由两个版本的综合算法综合的关系的 Codd 的第三范式属性。本节最后介绍了一种用于合成可证明的第三范式关系的新算法。在第 7 节中，合成的模式显示为最小尺寸。

2　关系模型

2.1　关系

在 Codd 的关系数据库模型中，一组域上的数学关系用于描述数据项之间的连接 [7]。然而，并非所有关系都同样能很好地描述这些联系 [8]。为了判断各类关系的有效性，我们首先回顾与关系模型相关的术语。

从概念上讲，关系是一个表，其中每一列对应一个不同的属性，每一行对应一个不同的实体（或元组）。对于每个属性，都有一组可能的关联值，称为该属性的域。不同的属性共享一个域是很常见的。例如，属性 QUANTITY__IN__STOCK和 SIZE__OF __CLASS均采用来自称为NONNEGATIVE INTEGERS（非负整数域）的值。关系中的（实体，属性）条目是与从属性域中选择的实体关联的值。形式上，关系是与关系属性关联的域的笛卡尔积的（有限）子集。用于描述数据库关系结构的符号包括关系名称（例如 R）和 R 中的一组属性（例如 $\{A_1, A_2, \cdots, A_n\}$），并写作R$\{A_1, A_2, \cdots, A_n\}$; 例如见图(1a)。属性的顺序并不重要，因为属性名称在关系中是不同的。（这是区分属性和域的原因之一。）从符号上讲，我们将在字母表开头附近使用大写字母表示简单（即单例）属性，在字母表末尾附近使用大写字母表示复合（即组）属性。

组成关系的实体集通常随着实体的插入、删除和修改而变化。这是数据库关系不同于数学关系的重要方式之一。文献中经常使用“关系”一词来描述关系的结构（例如R$\{A_1, A_2, \cdots, A_n\}$），称为其静态的意图，以及关系中的元组集合，称为其扩展。在下文中，除非另有明确说明，“关系”一词将指意图。也就是说，我们通常指的是关系的结构，而不是元组本身。

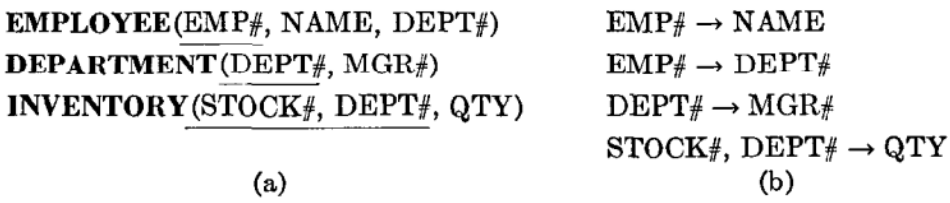

Fig. 1. Relations and functional dependencies: (a) an example of a relational schema (underlined attributes are keys); (b) functional dependencies for the schema of (a).

2.2 函数依赖

正如我们将在后面的部分中看到的，在选择如何将属性分组为关系时，考虑功能关系非常重要。数据库属性之间的函数关系以函数依赖的概念形式化。

令 A 和 B 为属性，令 DOM (A) 为 A 的域，DOM(B) 为 B 的域，令f 为时变函数，使得 f: DOM(A) —> DOM(B ）。f 不是精确数学意义上的函数，因为我们允许 f 的扩展随时间变化，就像我们允许数据库关系的扩展随时间变化一样。也就是说，如果 f 被认为是一组有序对{(a, b) | a ∈ DOM(A) 和b ∈ DOM(B)}，那么在每个时间点，对于给定的 a∈ DOM(A) 值，最多有一个b ∈DOM(B) 值。为了区分 f 和数学函数，我们将 f 称为函数依赖（缩写为 FD）。为了符号方便，我们通常省略“DOM”并写为 f: A —> B。如果存在 FD f: A —> B，则称 B 在功能上依赖于 A。

上述定义以明显的方式概括为复合属性的函数依赖性。如果 X = {$A_1, A_2, \cdots, A_n$ | 且 Y = {Bi,..., B m }是属性集，则 f:X —> Y 表示f: DOM(Ai) × × DOM(A n) —> DOM(Bi) × × DOM(Bm)。我们通常会在 FD 中省略集合符号并写成 f: {Ai, . .., A n j —> {Bi,. .., B m } 简单地表示为 f: {A1,. .., An}× {B1,..., Bm}.例如，图1(a)中属性的函数依赖关系如图 1(b) 所示。

在本文中，我们将假设对于任意两组属性X和Y，至多有一个FD X -> Y。属性可能需要重命名以保证这一假设。这一限制很重要，将在 4.2 节中详细讨论。稍后我们还将证明，非功能关系不需要满足这种唯一性假设。

考虑到这个假设，如果 f: A —> B 我们经常将 A —> B 写为缩写。符号 表示不存在感兴趣的 FDA —> B（尽管在某种关系中的给定时间点，A 的任何值都可能不具有多个对应的值） B）。

设 为关系，X 为的子集。如果Ai中的每个属性. A n)不在 X 中，则在功能上依赖于 X，并且如果 X 的子集不具有此属性。显然，一个关系可以有很多键。R 的超级键是 R 中包含 R 的键的任何属性集。（每个键也是一个超级键。）引入超级键的概念主要是为了简化后面章节中的证明，

2.3 关系操作

在他对关系模型的最初描述中，Codd 引入了关系代数作为关系数据库模型的数据操作语言 [7]。我们会对两种基本的关系代数运算感兴趣：投影和连接。

关系在其属性 X 的子集上的扩展的投影是通过删除 X 中不存在的属性而获得的元组集合。如果两个元组现在无法区分，因为它们仅在被删除的属性上有所不同，那么它们是“合并”到一个元组中。也就是说，投影的结果必须是与 X 的属性相关的域的笛卡尔积的子集。

连接操作用于在不同关系中出现的属性之间建立连接。这里我们要考虑的唯一连接操作是自然连接（即相等连接）。域 B 上关系R(A, B) 的外延与关系S(B, C) 的外延的自然连接（表示为 R>S）定义为。也就是说，它将与公共 B 值相关的所有 A 和 C 值链接在一起。

2.4 模式

任何数据模型（无论是关系数据模型还是其他数据模型）的目的都是允许模型的用户描述和操作 他打算存储在数据库中的现实世界中的对象之间的关系。在关系模型中，这样的关系集合以关系模式表示。关系模式由一组数据库关系以及每个关系的一个或多个键的规范组成（例如参见图1(a)）。

我们会说，如果 X 是 R 的键且 A 是 R 的任何其他属性，则函数依赖 X —> A 体现在关系 R 中。模式中体现的 FD 集合是所有关系中体现的 FD 的并集。模式的关系。

请注意，这种模式表述是 Codd 的 [8] 的修改，其中 FD 作为关系及其键的附加信息给出。我们采用这种修改后的模式概念有几个原因。首先，我们所知的所有数据定义语言都只允许关系和键的规范。其次，我们对模式的定义消除了明确谈论 FD 的需要。FD 凭借我们对键和实施例的定义而隐式存在。第三，我们将看到第三范式组织了一个关系

模式，使得作为外部信息给出的每个FD要么体现在某种关系中，要么可以通过连接和投影操作从体现的FD中恢复。将体现的概念作为一个原始概念可以大大简化随后的理论。

2.5 标准化

Codd 观察到，某些关系具有不适合描述数据库的结构属性。这导致他定义了一系列关系的三种范式。
首先，关系值域被排除在关系之外。如果每个域都包含简单值，则关系处于第一范式（缩写为 INF）。INF 有两个主要优点[7]。首先，它允许将数据库视为表的集合——一种非常简单且易于理解的结构。其次，它允许定义一小类原始运算符，这些运算符能够操纵关系以获得属性之间的所有必要的逻辑连接。
引入第二范式和第三范式是为了纠正某些函数依赖引起的问题。要检查这些问题，请考虑重新通过加入DEPT__INV (STOCK#, DEPT#, QTY, MGR#)获得图1(a)属性 DEPT# 上的DEPARTMENT和 INVENTORY关系。将特定 DEPT# 的第一个库存项目插入到DEPT__INV的扩展在 DEPT# 和它的 MGR#。删除特定 DEPT# 的最后一个清单项目会丢失该 DEPT# 与其 MGR# 之间的连接。这些副作用称为插入删除异常，仅在插入或删除 DEPT# 的第一个或最后一个元组时发生。此外，如果允许对各个元组进行任意更新，则对于 DEPT# 中的每个 STOCK#，重复 DEPT# 与其 MGR# 之间的连接可能会导致不一致的关系。出现这些问题的原因是 MGR# 在功能上仅依赖于键 STOCK#、DEPT# 的一部分。为了消除DEPT_INV的这些问题，必须将DEPT_INV放入第二范式。
当某个属性在功能上依赖于一组属性的子集时，就会出现部分依赖性。让和是函数依赖，其中 m < n。属性在 f 中是无关的，因为Ai,..., A m 足以在功能上确定 B。在这种情况下，据说 B部分依赖于。如果对于给定的f，不存在具有上述属性的g ，则B完全依赖于。也就是说，f域中没有无关的属性。
如果属性Aj出现在R的任意键中，则称其在R中为素数。否则，在R中为非素数。如果在 INF 中且其每个非素数属性为，则关系处于第二范式 （缩写为 2NF）完全依赖于每个键.关系DEPT__INV (STOCK#, DEPT#, QTY, MGR#) 不在 2NF 中
因为 MGR# 是非主属性，并且部分依赖于键 STOCK#、DEPT#。图1(a)中的关系DEPT和INVENTORY属于2NF。
现在考虑关系EMP__DEPT (EMP#, NAME, DEPT#, MGR#)通过加入EMPLOYEE和DEPARTMENT关系得到如图1(a)关于 DEPT#。尽管EMP __DEPT处于 2NF，但它显示相同的问题作为DEPT-INV。在特定 DEPT# 中插入或删除第一个 EMP# 会产生异常，因为在此过程中创建或销毁了 DEPT#-MGR# 连接。DEPT# 中每个 EMP# 的 DEPT#-MGR# 连接的重复会产生与DEPT-INV 中相同的一致性问题。在这种情况下，会出现问题，因为 MGR# 在功能上通过属性 DEPT# 依赖于密钥 EMP#。为了消除这个问题，必须将EMP—DEPT关系转化为第三范式。
令R(Ai, ..., An)为关系。如果存在一组属性使得，则属性Ai传递依赖于一组属性 X ，与A；不是 X 或 Y 的元素。
如果一个关系的非素数属性都不传递依赖于任何键，则该关系处于第三范式（缩写为 3NF）[8]。3NF 关系也在 2NF 中，因为如果属性 Ai 部分依赖于键 X，则 Ai 传递依赖于 X，因为 关系 EMP__DEPT 不在 3NF 中，因为 MGR# 是非素数且是过渡的
完全依赖于密钥 EMP#。考虑到图 1(b) 的 FD，图1(a)中的所有关系都属于3NF（因此也属于 2NF） 。范式关系和周围问题的更多例子可以在[7-9]中找到，

3 综合关系模式

3.1 综合问题和非函数关系

Codd 表明，通过对 FD 已知的 INF 关系应用简单的分解步骤，该关系可以分解为 3NF 中包含所有 FD 的一组关系[8]，在 [3] 中提出：由于FD完全决定了一个关系是否属于3NF，因此可以选择FD作为基本概念并从中构建3NF关系。在推进该提案的过程中，提出了一种有效的算法技术来实际从 FD 构建关系。在本文中，我们提出了一种改进的算法，然后讨论了该算法合成的模式的属性。
从 FD 构建关系模式的方法完全依赖于将所有数据关系表示为 FD 的能力。但显然，并不是世界上的每一个逻辑连接都是有效的。尽管如此，我们声称数据库描述中属性之间的所有联系都可以用FD来表示。只要连接有效，当然就没有问题。非功能性连接需要特殊处理。
一组属性Ai，A 2，...，A n之间的非功能性连接f将被表示为以下FD：$f: \mathrm{A}_1, \mathrm{A}_2, \ldots, \mathrm{A}_\mathrm{n} \rightarrow \theta . \theta$ 是 f 独有的属性； 它不会出现在任何其他 FD 中。代表非功能关系的每个FD都有其自己的私有9属性。所有这 9 个属性的基础域是集合 $\{0,1\}$。对于每个元素$\left(a_1, a_2, \ldots, a_\mathrm{n}\right) \in \mathrm{DOM}\left(\mathrm{A}_1\right) \times \mathrm{DOM}\left(\mathrm{A}_2\right) \times \cdots \times \mathrm{DOM}\left(\mathrm{A}_\mathrm{n}\right), f\left(a_1, a_2, \ldots, a_\mathrm{n}\right)=1$当且仅当$\left(a_1, a_2, \ldots, a_\mathrm{n}\right)$在 f 下相关。因此， f 的扩展完全定义了Ai,..., An 之间的非函数关系。例如，DRIVER 和

AUTOMOBILE 之间的非功能关系，其中每个 AUTOMOBILE 可以由多个 DRIVER 驱动，每个 DRIVER 可以驱动多个 AUTOMOBILE，由 FD DRIVER,AUTOMOBILE表示。

请注意，一组属性之间可以存在多个非函数关系，而不会违反 FD 的唯一性假设。例如，我们可以在 DRIVER 和 AUTOMOBILE 之间建立第二个关系来指示所有权： DRIVER,AUTOMOBILE 。通过为每个非函数关系分配唯一的 theta，保留了 FD 的唯一性假设。
这个 9 表示法允许我们将所有非功能关系表示为 FD。综合算法将为这些非函数关系中的每一个生成大约一个关系。在第 5 节中，我们将准确展示每个“非功能性 FD”如何在综合关系模式中体现。

3.2 综合问题的形式化

尽管综合问题的动机来自数据库管理，但我们可以用纯粹的符号术语将问题形式化，如下所示。我们得到一组符号（即属性）和一组符号到符号（即FD）的映射集合F。问题是找到一个集合 C = {C1, .. ., C m } 的
我们得到以下一组 FD：

我们根据共同的左侧对 FD 进行分组，得到四组：

对于每个组，我们构建一个由组中所有属性组成的关系：

其中带下划线的属性是键。

We are given the following set of FDs:

f_1: A → B　　f_4: B → D
f_2: A → C　　f_5: D → B
f_3: B → C　　f_6: ABE → F

We group the FDs according to common left-hand sides, obtaining four groups:

$g_1 = \{f_1, f_2\}$　　$g_3 = \{f_5\}$
$g_2 = \{f_3, f_4\}$　　$g_4 = \{f_6\}$

For each group we construct a relation consisting of all of the attributes in the group:

R_1(A,B,C)　　R_3(D,B)
R_2(B,C,D)　　R_4(A,E,B,F)

where the underscored attributes are keys.

图 2. 从 FD 导出模式

S 的子集（即关系的集合）以及对于每个Ci 的Ci 子集的集合（即每个关系的键的集合）满足三个属性：首先，F “体现”在C中（即关系体现给定的FD）。其次，每个Ci不能有传递依赖（即它在 3NF 中）。第三，C 的基数最小。
这种对问题的处理仍然有些模糊，因为我们还没有讨论组成 FD 的代数规则。为了激发对这些规则的需求，我们提出了一种简单的合成算法。该算法忽略了代数考虑，并且将被证明是不充分的。

3.3 简单的合成过程

从给定的 FD 集合获取关系的一种（过于）简单的方法是将功能上依赖于同一组属性的所有属性组合在一起。这表明了以下过程。首先，将给定的 FD 集合划分为组，使得每组中的所有 FD 具有相同的左侧。然后，为每个组构造一个由该组中出现的所有属性组成的关系。每组中的FD的左侧是对应关系的键。例如，见图 2。
从示例中可以看出该方法的几个不良特性。首先，综合关系不属于 3NF。例如，在图2的关系Ri中，C传递地依赖于密钥A。在R 4中，B部分地依赖于密钥AE。非标准化关系是由于给定 FD 组中的冗余造成的。稍后我们将看到 f2是多余的，并且 B 是f6中的无关属性。
其次，FD 的左侧不一定是关系的键，尽管它们始终是超键。在 R 4中，ABE 是一个超级密钥，但不是密钥，因为 B 是无关的。

第三，这个过程综合了太多的关系。由于f 4和f s 彼此相反，因此关系R 3 是无关的。这是由于在构建R 2 以凭借 f 5 将 D 识别为第二个密钥时的过程失败，而不是将 f 5放入单独的关系中。
为了解决这些问题，我们首先必须形式化冗余FD的概念。然后我们将返回克服上述困难的合成算法的介绍。

4 函数依赖的代数和语义

4.1 阿姆斯特朗函数依赖性公理化

Armstrong [1] 给出的 FD 的完整公理化为后面章节中讨论的 FD 代数的研究提供了理论背景。阿姆斯特朗表明，如果给定的 FD 集合存在于关系（的扩展）中，则可以使用公理从给定集合导出的任何 FD 也必须存在。阿姆斯特朗提出了几个等价的 FD 公理化。我们将使用的基于Delobel和 Casey [10] 证明的 FD 属性。他们是

A1 （自反性）$X \rightarrow X$.
A2（增广）如果 if $X \rightarrow Z$ then $X + Y \rightarrow Z$.
A3（伪及物性）如果$X \rightarrow Y$ and $Y + Z \rightarrow W$ then $X + Z \rightarrow W$.。其中符号“+”表示“集合并集”（不一定是不相交的集合）。
如果R(A, B)是关系，则公理 A1可以应用 X = {A, B} 来证明 A, B —>A, B 或应用 X = {A} 来证明 A —>A 。
A2 的含义很简单，如果 f: X —> Z，则可以创建另一个 FD g，其中g的域包括 X 以及一些其他无关属性 Y，其值对所选 Z 的值没有影响由g。因此，知道 A —> A，我们就可以得到$A, B \rightarrow A$ (i.e. $X = \{A\}, Z = \{A\}$, and $Y = \{B\}$).
A3 是组成 FD 的替换规则。设 f: X —> Y 和g: Y + Z —> W。该公理声称存在h: X + Z —> W。要了解h从何而来，请考虑将h应用于给定的x ∈ DOM(X) 和z ∈ DOM(Z) 分两步进行。首先，将 f 应用于x，产生唯一的y ∈ DOM(Y)。其次，将g应用于y和z，产生唯一的w ∈DOM(W)，从而完成h的应用。象征性地，我们可以说h(x, z)被定义为g(f(x), z)。另外，请注意，在公理 A3 的陈述中，如果 Z 是空集，则伪传递性变为简单传递性。
令G为 FD 的集合。G的闭包，表示为G' +，被定义为在A1、A2和A3下闭包的G的最小超集。对于给定的G，G +可以被证明是唯一的。根据阿姆斯特朗理论我们知道，如果G是关系 R 的给定 FD 集合，则G +中的每个 FD也存在于R 中。
如果G + = (G -{g}) + ，则FD g ∈ G在 G 中是冗余的。如果G + = H +并且H不包含冗余 FD，则H是给定 FD 集合G的非冗余覆盖。
引理 1 中阐述了 FD 的一个重要属性，稍后将用于证明许多定理。它基于“推导”的概念，我们将非正式地将其视为阿姆斯特朗公理在给定一组 FD。正式的进展出现在附录中。
引理 1.令G为 FD 集合，并令 g : X —> Y 为 G 中的 FD。如果 h : V->W 在 G +中且 g 用于从 G 导出 h，则 V ->X在 G^+ 中 .
证明: 我们在这里使用推导的非正式概念给出一个直观的论证。附录中给出了使用推导的“推导树”模型的正式证明。我们引入符号 U => Z 来表示 FD U —> Z 可以通过在给定的 FD 集合上应用阿姆斯特朗公理之一来导出。符号 U =>*Z 意味着 U —> Z 可以通过使用公理的多个应用来推导。现在引理表明存在使用g的推导 V =>> W。也就是说，对于某些（可能为空）属性集 Z 存在推导 V =>*ZX => ZY =>*W （步骤 ZX => ZY 是使用g的步骤）。但 V =>*ZX 意味着 $V \rightarrow ZX$，又意味着 $Y \rightarrow X$，从而证明了引理。

4.2 函数依赖的语义

本文对FD 的处理是基于阿姆斯特朗公理的严格句法处理。为了使用这种方法，我们必须做出以下唯一性假设：对于给定的一组 FD G和 FD X —> Y，要么 X —> Y 不在G +中，要么在 G + 中存在唯一的 FD X —> Y+。也就是说，如果同一组属性上有两个FD，那么它们就是同一个FD；如果 f : X —> Y 且g : X —> Y，则 f 与g相同。因此，被接受为综合算法输入的 FD 集合被假定为不仅满足阿姆斯特朗公理，而且满足唯一性假设。（这两个假设也是所有以前的 3NF 句法方法所必需的（例如 [10, 11, 2]））。 从几个例子可以看出这个假设是相当有力的。
Let f 1: DEPT # -> MGR# 和MGR#,FLOOR -> NUMBER OF—EM员工们。j) 和f 2的一种解释是f i 确定每个部门的经理，f 2确定在特定楼层为特定经理工作的员工数量。通过对f i 和 f>应用伪及物性，我们得到 f 3 : DEPT#,FLOOR —> NUMBER_OF_EMPLOYEES，它确定特定楼层的特定部门经理的员工数量。如果一名经理可以管理多个部门，则 f 3 与语法上相同的 FD gi不同。DEPT#,FLOOR —> NUMBER—OF— EMPLOYEES，确定特定楼层特定部门 的员工数量。为了使gi 与 f 3不同，必须更改属性名称以使 FD在语法上不同。例如，可以改变
f> 和 pi 使得 f,: MGR#,FLOOR -> NUMBER__OF__EMPLOYEES—OF—

经理和gi： 部门。现在gi与 f i和 f 2的组合不同。
作为第二个示例，令 f 4 : EMP# —> MGR# 和 f 5 : MGR# —> EMP#。这里的情况一定是 f 4是fs的倒数。因为如果我们组合fi和 f 6，我们就得到gi 。EMP# —> EMP#。由于只有一个 FD 将 EMP# 连接到 EMP#（根据我们的假设），并且根据阿姆斯特朗公理，恒等函数必须存在，因此g 2必定是恒等映射。这意味着ft = U 如果我们采取这样的解释： fi 将雇员映射到他的经理，并且f 3将经理的MGR# 映射到他相应的EMP#，那么当然f 4fs'。所以要把这个纳入解释时，必须使fi和 f 5 在语法上不同（例如 f 5 :MGR# —> EMP#OF MGR）。
作为第三个示例，令 STOCK# -> STORE# 和fr - STOCK#,STORE# —> QTY。由于f 6和fi的组合是g 3 : STOCK# —> QTY，所以（根据我们的假设）fi中的属性STORE#一定是不需要的。但是假设fa将 STOCK# 映射到负责订购该商品的商店的 STORE#，并且 f 7将商品的 STOCK# 和出售该商品的商店的 STORE# 映射到现有数量。在这种情况下，g 3并不意味着 STORE# 在 f 7中是无关的。为了防止发生这种语法推断，我们必须更改属性名称（例如 f 6 : STOCK# —> ORDERING__STORE#）。
在每个例子中，句法推断要么是错误的，要么是误导性的。在每种情况下，我们都通过重命名属性以将其与另一个属性区分开来解决问题。这种重命名实质上将我们拥有的关于 FD 的一些语义知识转移到了句法层面，可供 FD 的代数使用。
指定一组不会导致无效句法推理的 FD 显然是一个难题。因为没有任何语法检查仅基于FD的代数可以确定给定的FD集合是否满足唯一性假设。然而，如果我们要使用 FD 的形式代数，我们必须假设所有句法推论都是有效的。如果我们有一个自动语义分析器可以判断每个句法推理的有效性，那么我们就可以用它作为筛子来剔除无效的推理。不幸的是，这样的语义分析器远远超出了现有技术的水平。因此，我们将在句法推论有效性的假设中添加一个条件，即所有句法推论都（或至少可以）检查语义有效性。如果推论无效，则可能会导致某些属性的重命名或简单地被拒绝。
第三范式是受 FD 代数控制的严格句法属性。在本文中，我们给出了从 FD 到 3NF 模式的映射的完整说明，假设阿姆斯特朗公理和唯一性假设被接受。鉴于阿姆斯特朗的完整性证明，我们相信这些假设在关系数据库建模中是相当合理的。我们并不是要解决如何判断句法推论的语义有效性的问题。这类语义问题不太容易理解，似乎比确定 3NT 的句法问题更困难。他们的解决方案仍有待进一步研究。

5 更复杂的合成过程

5.1 算法描述

3.3 节的简单综合过程导致了问题，因为忽略了组成 FD 的规则。主要困难在于，过滤到合成模式中的冗余 FD 会创建额外的属性，并导致属性之间的非标准化连接。通过首先对给定的 FD 集进行非冗余覆盖，可以缓解规范化问题。例如，在图 2 中，fa是冗余的，因此不会出现在给定 FD 的非冗余覆盖中，从而避免了 Ri 的 3NF 违规。
找到非冗余覆盖不足以避免诸如图 2 中的 fe 之类的问题 FD。可以通过从 FD 左侧删除无关属性来消除这个进一步的问题。属性 X,在 FD g ∈ G, g: Xi, .. 中是无关的。, X p —> Y, 如果Xi,. 。., Xi- i , X i+ i , . .. , X p —> Y 在G +中。消除无关属性有助于避免部分依赖关系和非键的超键，如图 2 的 R 4所示 。
如果两个关系的键在功能上相互依赖（即等效），则这两个关系可以合并在一起。如果左侧功能相同，则可以通过将两组 FD 合并在一起来完成合成过程。例如，图 2 中的g 2和g 3可以合并为一个组。
以下过程包括上述改进：
算法1
1.（消除无关属性。）令F为给定的FD集合。从 F 中每个 FD 的左侧消除无关属性，产生集合6。 如果某个属性的消除不改变 FD 集合的闭包，则该属性是无关的。
2.G 的非冗余覆盖H。
3.（划分）将H划分为组，使得每组中的所有FD具有相同的左侧。
4.（合并等效键。）对于每对组，假设Hi和Hi，左侧分别为 X 和 Y，如果H + 中存在双射 X <-> Y，则将Hi和Hi合并在一起。
5.（构造关系。）对于每个组，构造一个由该组中出现的所有属性组成的关系。组中任何 FD 左侧出现的每组属性都是关系的键。（第 1 步保证此类集合不包含任何额外属性。）该算法找到的所有键都将被称为合成键。构造的关系集构成了给定的 FD 集的模式。
接下来，我们将删除步骤 4 的算法 1 称为算法1(a)

[2] 中提出了一种用于测试 FD 集合闭包中的成员资格的线性时间算法。使用此过程，可以在时间界限为0(L 2) 的情况下实现算法 1，其中L是编码给定 FD 集合的字符串的长度。

5.2 综合模式的完整性

如果 S 中体现的FD 的闭包等于F + ，则模式 S完全表征了一组 FD F。为了表明算法 1 合成了一个完整表征给定 FD 的模式，请考虑作为算法 1 的输入给出的一组 FD F。令H为通过消除无关属性和冗余 FD 产生的 FD 集合。显然，H +仍然等于F +。设S是从F合成的模式。由于H正是§中体现的FD的集合，并且H + = F +，所以F中的每个FD都可以从S中体现的FD的子集导出。因此，S完全表征了F 。
我们希望确定给定集合F中每个 FD 的外延都可以使用关系代数从合成模式S的外延中检索出来。我们认为这是根据 S 完全表征F 的事实得出的。
考虑 有f:X ->A ∈ F。我们首先注意到 X 中的无关属性可以被忽略。也就是说，如果扩展名为 FD f: X。—> A 其中 X> c X 可以从S的扩展中检索，那么由于 f 可以简单地通过增广从 f 获得，所以我们可以将 f 视为与 f 相同的 FD。现在，由于S完全表征了F，因此基于S中体现的FD H可以推导 f 。在附录中，我们表明，如果f没有无关属性，则可以仅使用伪及物性从H推导它公理。由于伪及物性的应用 完全对应于关系代数中的连接，因此 H的 f 的推导可以通过S关系的外延上的一系列连接来模拟。这样，每个 f ∈ F的外延可以是使用关系代数从S的扩展中检索。也就是说，我们的“完整表征”概念满足了这样的直觉：给定 FD 集中指定的所有关系实际上都可以从综合模式的扩展中检索。

5.3 非功能关系

我们引入了一种特殊的符号来表示输入 FD 中的非函数关系。我们现在必须确保这些 FD 以预期的方式运行。
如果 X —> 0 位于给定算法 1 的 FD 集合中，则 X —> 0 或 Y —> 0（其中 Y <->X）出现在算法合成的模式中。这是以下引理的结果，在附录中得到了证明。
引理 2.如果 X —> 9在一组 FD G 中，则对于 覆盖 G 的 H 的任何非冗余，X —> 0在 H 中或 Y —> 0在 H 中，其中 X —> Y 和 Y — > X. □ 因此，非功能关系以几乎与在给定 FD 集中指定的形式相同的形式出现在模式中。
根据上述引理，为了允许表示非功能关系而发明的 0 属性总是出现在合成模式中。它们是如何解释的？要了解这一点，请考虑以下示例。假设在给定的 FD 集中指定了两个非函数关系fi: AB —> 0i 和f 2 : AB —> 0 2。（再次注意，FD 的唯一性假设并不强制 A 和 B 之间非函数关系的唯一性。）Alg 的步骤 4。1 将这两个 FD 合并为一个组，产生关系R(A, B, 0i, 0 2)。为了区分 A 和 B 的给定值对是否满足fi、f 2或同时满足fi和 f 2 ，必须保留0i 和 0 2属性。例如 (o, b, 0, 1) 6 R 表示a, b满足 f 2 但不满足fi。请注意，如果一组属性之间只有一个非功能关系，则通常可以删除 0 属性，因为区分关系的问题消失了。例如，如果只有 fi存在，则通常只包含在 f i下相关的(a, b)对；元组(a,b, 0) 通常不会包含在扩展中。因此在这种情况下，0 属性将被完全删除。

6 第三范式模式

6.1 简介

在本节中，我们将展示各种综合关系在什么条件下属于 3NF。我们首先证明算法1(a)（即没有步骤 4 的算法 1）总是产生 3NF 模式。然后我们检查算法 1。引入非素数属性的推导性质，并证明它是算法 1 生成 3NF 模式的充分条件。不幸的是，在某些情况下，FD 不满足此属性，因此可能导致具有传递依赖性的关系。一个这样的例子被提出并被证明是德洛贝尔和凯西给出的定理的反例。

6.2 算法1(a)'模式

为了证明算法1(a)合成的每个关系 都属于 3NF，我们证明传递依赖意味着非冗余覆盖中存在冗余 FD。我们将使用引理 1 来证明产生矛盾的 FD 的存在性。引理 1 将在所有后续的 3NF 证明中以这种方式使用。
定理1.令R(Ai , .. ., A„) 为使用算法1(a) 从FD F 集合合成的关系。那么R的任何非素数属性都不会传递依赖于R的任何键。也就是说，R在3 NF 中。
证明。假设 A ; 是非素数并且传递依赖于R 的密钥 K。 （K 不需要合成。）也就是说，存在一个 XC {Ai,. .., A n } 使得 K -y X、X -+-> K 和 X -> A位于F + 中，且Ai不在 X 中。
我们首先观察到 Ai 传递依赖于R的合成密钥。设 Z 为出现在用于合成R 的 FD 左侧的R密钥。Z —> X 在F +中，因为 Z 是此外，X -h > Z，因为如果 X -> Z，则 X -> Z 和 Z -> K 将意味着 X -> K，与原始传递依赖中的 XK 相矛盾。因

此 Z → X、XZ 和 X → Ai 也是传递依赖。
令H为算法 1(a) 中计算的G的非冗余覆盖。现在我们将证明H中出现的 Z —> Ai是多余的。为此，只需证明 Z —> X 和 X —> A 即可；两者都可以从H — |Z —> Ai}导出。
由于合成R时使用的唯一 FD 的形式为Z —> Aj ，因此对于所有Xk ∈ X，Z —> Xk必定在H中。由于 Ai 不在 X 中，因此 Z —> Xk在H — { Z —> Ai}。
假设存在 X —> A 的推导；在H中使用 Z —> A;。然后根据引理 1，我们有 X —>Z。 但这违反了 X -+-> Z的传递依赖关系。所以 X —> A ； 必须可以在不使用 Z —>A的情况下导出； 。
由于 Z —> X 且 X —> A；都可以从II — [Z —> Ai}导出，则 Z —> Ai 在H中必定是冗余的，与H是非冗余的相矛盾。但这反过来必然意味着传递依赖不存在。
上述定理由Wang和Wedekind首先提出[12]；然而他们的证明是不正确的 [4]，在他们的证明中，他们只认为传递依赖可以在H 中导出，而不是H — [Z —> Ai}。根据上述证明，他们声称如果 K —> X 和 X —> A ； 是传递依赖，则 Iv —> A; 可以通过伪传递性导出。他们断言，这违反了 K —> Ai处于非冗余覆盖层中的事实。然而，只有当我们能够证明 K —> X 和 X —> A 时，后者才是正确的；可以从闭包中导出，无需使用 K -> Ai。例如，G = [A —> B, B —> A, A —> C}是一组 FD，其中A —> B 和 B —> C位于闭包中，但A —>C不是冗余的，因为B —> C不能在没有A —> C的情况下从G导出。无论如何，他们的定理如所述是正确的，并且上述论证使用引理 1 和 X -+-> K 中的重要事实纠正了他们的证明。传递依赖。
有趣的是，他们并没有消除算法 1(a) 版本中的超级密钥。 这不是一个错误，因为他们明确假设FD 左侧不存在无关属性。然而，不需要做出这一一般假设，因为可以通过算法消除一些无关的属性。事实上，为了与 FD 代数完全一致，必须消除这些无关的属性。当然，并不是所有这些无关的属性都可以通过这种方式消除；由于第 4.2 节中讨论的原因，许多语义错误必须由用户负责。
人们可能期望定理 1 的证明能够推广到算法 1 合成的模式。不幸的是，情况并非如此。不属于 3NF 的模式可以通过算法 1 来合成，如图3(b) 所示。 在下一节中我们

将添加一个足以保证算法 1 生成的模式的 3NF 的进一步前提条件。

6.3 3NF的充分条件

为了表明没有非素数属性传递依赖于 R 的任何键，我们将使用以下属性：
如果以下命题成立，则称属性 A 满足关系 R 中的属性 P： 令H为算法 1 的步骤 2 生成的非冗余覆盖。如果 K —> A 在H中，并且 K —> A 用于合成 R ，那么对于R的任何素数属性B，可以在不使用K→A的情况下导出FD K→B（即可以在H →{K→ A}中导出）。
属性 P 严格来说是 FD 派生的句法属性，据我们所知，在现实世界关系方面没有语义解释。这是我们所知道的足以保证算法 1 产生 3NF 关系的最弱性质。性质 P 足以保证 3NF 的证明与定理 1 的证明相同。
定理2.设R(Ai, ..., A n) 是通过使用算法1从一组FD F合成的关系之一。如果R的所有非素数属性满足属性P，则R在3NF中。
证明。设A； 是 R 的非素数属性，它传递依赖于 R 的某个键 Y。也就是说，存在 Z c {Ai,..., A n } 使得 Y -> Z, Z -+> Y, 和 Z -“ A ; 与 A ； 不在Z。
令H为算法 1 中计算的非冗余覆盖。令 K 为一个键，使得h: K —> Ai 位于H中。也就是说，h是一个 FD，它带来了 A ； 通过算法 1 转化为 R。
由于 K 是键，所以 K —> Z。此外，Z -t-> K，因为如果 Z —> K，则 Z —> K 且 K —> Y 意味着 Z —> Y，这是一个矛盾。所以我们有一个新的传递依赖： K —> Z、Z -+> K 和 Z —> A;。我们要证明 K → Z 和 Z → Ai 在(H — {K —> A,}) +中，以建立H是冗余的矛盾。

FDs	Schema synthesized by Algorithm 1
f_1: $X_1,X_2 \rightarrow A$	$R_1(\underline{X_1,X_2,C},A)$ (from f_1 and f_2)
f_2: $C \rightarrow X_1,X_2$	
f_3: $A,X_1 \rightarrow B$	$R_2(\underline{A,X_1},B)$
f_4: $B,X_2 \rightarrow C$	$R_3(\underline{B,X_2},C)$

A does not satisfy property P, yet R_1 is in 3NF.

(a)

FDs	Schema synthesized by Algorithm 1
g_1: $X_1,X_2 \rightarrow A,D$	$S_1(\underline{X_1,X_2,C,D},A)$ (from g_1 and g_2)
g_2: $C,D \rightarrow X_1,X_2$	
g_3: $A,X_1 \rightarrow B$	$S_2(\underline{A,X_1},B)$
g_4: $B,X_2 \rightarrow C$	$S_3(\underline{B,X_2},C)$
g_5: $C \rightarrow A$	$S_4(\underline{C},A)$

S_1 is not in 2NF since A is partially dependent upon the key CD.

(b)

图 3. 侵犯财产 P 的示例

令 Z ={$B_1, B_2, \cdots, B_m$}。我们区分两种情况。如果Bj是素数，则属性 P 保证 K —> Bj可以在不使用 K —> Aj的情况下导出。如果Bj不是素数，则存在 FD K' —> Bj将Bj带入 R。因为 K —> K' 可从H — [K —> A;} 和 K 7导出（通过属性 P）—> Bj在H — [K —> A;} 中，我们得到 K —> Bj可以在不使用 K —> A 的情况下导出；。因此 K —> Z 在(H - {K —> A ; J) +中。

现在假设 Z → Aj在其推导中 使用 K → Aj 。然后根据引理 1，Z → K，与传递依赖相矛盾。因此 Z —> A；位于(H — {K —> Aj }) +中。

FD K —> Z 和 Z —> Aj 在(H — {K —> A ; }) +中，证明H是冗余的，这是一个矛盾。因此传递依赖不可能存在。□

对属性 P 的需求源于算法 1 步骤 4 中等效键的合并。假设 Ki 和 K 2在步骤 4 中合并，因为 Ki <-> K 2，并且假设 Ki —> Z, Z -+> Ki， Z —> A 是综合关系中的传递依赖。如果 Ki 和 K 2是两个独立关系的键，则这种传递依赖性将不存在，就像使用算法 1(a) 的情况一样。产生传递依赖的原因之一是存在一个Zj t Z，其中 K 2 —> Z；（和 Ki -> A）在非冗余覆盖中，但 Ki -> Zj （和 K 2 -> A）不在覆盖中。因此，关系必须同时包含 Ki 和 K 2以体现传递依赖。传递依赖中的FD Kj —> Z 是 Ki —> K 2 —> Z的组合。如果 A 不具有属性 P，则可能需要 Ki —> A 才能获得 Ki —> K 2，在这种情况下Ki —> A 不必是冗余的（就像算法1(a) 中那样）。 然而，如果 A 确实具有属性 P，则 Ki —> K 2不需要 Ki —> A，因此 Ki —> A是多余的，我们有定理。

考虑图 3(a) 中的 FD 集合，它通过算法 1 生成关系Ri(Xi,X 2 , C, A)。（读者可以检查 Xi, X 2 —> C 处于闭包中）属性 A 在 R 中是非素数，并且不满足属性 P，因为导出 Xi, X 2 —> C 的唯一方法是使用 Xi, X 2 —>A。然而，尽管违反了性质 P，关系 Ri 仍处于 3NF 状态。因此性质 P 不是 3NF 的必要条件。

图3(b)提供了一个 FD 示例，它表现出与图 3(a) 相同的属性 P 违规，但引起部分（因此，传递）依赖性。根据上述关于传递依赖的讨论，我们有 Xi, X 2 —> C, C -+> Xi, X 2和 C —> A，但是 Xi, X 2 —> A 不是冗余的，因为Xj， X 2 —> C 在其推导中需要 Xi, X 2 —> A。

属性 P 影响另一已发布的从 FD 合成关系的过程。Delobel和 Casey [10] 声称他们的分解过程在某种意义上与我们的算法 1 相当，产生 3NF 关系。然而，他们的主张是不正确的，因为图3(b)中的示例歪曲了他们的定理 [6]。

6.4 将关系放入3NF

对属性 P 的违反可能会导致 3NF 违反。一旦发现特定的违反 3NF 的情况，为了将关系放入 3NF，必须删除违规的依赖关系。足够方便的是，如果非主属性传递依赖于关系的键，则可以简单地从关系中删除该属性，并且生成的模式仍将体现相同的 FD。

定理3.令Rit (Ai ,..., An) 为使用算法1合成的模式S = {Ri ,..., R<}中的关系。令 H 为 FD 的集合，具体体现在S.设 Ai 是 R 的一个属性，它不会出现在R„的任何合成键中 ，并让 Ai 传递依赖于R 的某个键；；。假设从Rk中删除 Ai ，产生一个新关系Rr，从而产生一个新模式 §>' = {Ri,..., Rf,. ..， R M }。 那么§'中体现的FD集合的闭包等于H +。

证明。假设A；从R k中删除。由于 A, 没有出现在算法 1 合成的任何密钥中，因此它的删除只能影响形式为 f: X —> Aj的具体 FD，其中 X 是Rk的合成密钥。令H'为 S f中包含的 FD 集合。如果我们证明所有这样的 f 都在(Ii ') +中，则H + = (H') +。

通过与定理 1 和 2 的证明中使用的相同论证，如果 A; 传递依赖于Rk的任何键，那么它传递依赖于所有键。对于上述形式的eachf，X是一个键。因此，对于每个这样的X，X→V，Y +> X，并且V→A，，Ai不在V中，是传递依赖。

由于 Ai 不在 V 中，因此对于每个 X，即使在 A 之后， FD X —>V 仍然体现在Rk中；已移除。因此每个 X —> V 都在H +中。

为了证明 Y —> Ai 6 (H') +，我们必须证明 V —> A ； 不能使用任何 FD X —> A ； 在其推导过程中。但这是直接遵循的，因为 if X —> A ; 用于导出 V —> A,，然后通过引理 1 V —> X，与传递依赖之一中的 V -+-> X 相矛盾。

由于 X —> V 和 V —> Ai 在(H ') +中，f:X —>A；对于所有这样的f ，都在 (H') +中。因此 (H') + = H +。□

定理 3 为我们提供了一种删除不需要的传递依赖的简单方法：即从关系中删除有问题的属性。该定理保证生成的模式仍然体现给定的 FD 集。

传递依赖如此容易地被删除是相当令人惊讶的，因为形成模式的 FD 是非冗余的。看起来删除一个属性应该会导致 FD 的丢失。然而，在从 FD 的非冗余覆盖合成模式时，我们已经隐式地将新的 FD 添加到算法 1 的步骤 4 中的覆盖中，并且这些 FD 显式地体现在模式中。如果 X 和 Y 是II +中 X —> Y 和 Y —> X 的非冗余覆盖H中两个不同 FD 的左侧，则 X 和 Y 被合并并放入单个关系中。这将 X —> Y 和 Y —> X 添加为两个新的 FD，它们明确体现在模式中，即使它们可能没有出现在覆盖中。例如，在图 3(b) 中，FD Xi, X 2 —> C, D 和 C, D —> X x , X 2 明确体现在 Si 中，即使前一个 FD 不在覆盖范围内。额外的 FD 的添加使我们能够在不影响具体 FD 的闭包的情况下消除传递依赖。

从不同的角度来看定理 3，我们现在可以修改算法 1 以合成保证符合 3NF 的模式。令H为算法 1 步骤 2 产生的非冗余覆盖。令J为所有 FD X —> Y 的集合，使得 X 和 Y 是算法 1 步骤 4 中发现的等效键。令 h : Z —> A， ，h G H，以这样的方式体现在R k中： A; 出现在R k和 A的合成键中，传递依赖于R k的键。那么定理4说明h是冗余的；也就是说，h ∈ (H + J — {f i)) + - 因此，如果我们消除右侧不在任何合成键中且h ∈ (H + J — {A}) +的每个 FD h H ，那么我们就消除了所有传递依赖。如果存在非质数 A ； 如果它传递地依赖于R k 的密钥 ，那么定理 4 将保证我们的额外冗余检查将消除它。这引出了我们的主要结果，算法 2，它综合了一个可证明的 3NF 模式。

算法2

1.（消除无关属性。）令F为给定的FD集合。从 F 中每个 FD 的左侧消除无关属性，生成集合O。 如果属性的消除不会改变 FD 集合的闭包，则该属性是无关的。

2.G 的非冗余覆盖H。

3.（划分）将H划分为组，使得每组中的所有FD具有相同的左侧。

4.（合并等效键。）让 f = 0。对于每对组，说Hi和 ff； ，如果H +中存在平分X <-> Y，则分别用左侧X 和 Y 将Hi和 H 2合并在一起。 对于每个这样的二等分，将 X —> Y 和 Y —> X 添加到 f。对于每个 A ∈ Y，如果 X —> A 在H 中，则将其从H 中删除。对H中的每个Y —> B 与 B 6 X 执行相同操作。

5.（消除传递依赖性。）找到一个H' C H使得 (ff' + J) + = (H + J) +并且H'的真子集不具有此属性。将J的每个FD添加到其对应的H'组中。

6.（构造关系。）对于每个组，构造一个由该组中出现的所有属性组成的关系。组中任何 FD 左侧出现的每组属性都是关系的键。（第 1 步保证此类集合不包含任何额外属性。）该算法找到的所有键都将被称为合成键。构造的关系集构成了给定的 FD 集的模式。

步骤 1-4 的有效实现与算法 1 相同。步骤 5 可以通过使用[5]的隶属度算法有效地实现，然后算法 2 可以在与算法 1相同的0(L 2)时间范围内实现。

7 极小性证明

本节的目的是检查算法 2（或 1）针对给定的 FD 集合成的关系数量，并与体现这些 FD 的任何其他关系模式进行比较。我们将通过证明合成键的等价类数量在给定 FD 集的所有非冗余覆盖中是相同的来证明所有非冗余覆盖生成相同数量的关系。这将意味着算法 2 合成的模式在合成关系的数量上是最少的。

引理3.设Gi 和G 2是两个非冗余FD 集合，其中Gi = G 2。如果 g:X —> A 在 Gi 中，则存在h\Y—>B ，其中 h 6 Gi 且 Y —> X 和 X-^ YinGi。

普库夫。如果gr ^ Gi 且Gi = G 2，则gf G-> ，因此G 2中存在g的推导。用于导出g 的每个 FD h都在 G 2中，因此每个这样的h都在Gi 中。

如果h在 Gi 中的推导中不需要g ，那么我们可以构造g在 Gi 中的推导。该推导是通过模仿G 2中g的推导来构造的，并将该推导中的每个h替换为G,中推导的h。由于这个推导没有使用g，所以g在Gi中必然是多余的，这是一个矛盾。因此，至少有一个h必须在 Gi 中的推导中需要g 。

假设h : Y —> B在 Gi 中的推导中需要g: X —>A。然后由引理1, X —> Y。此外，由于h出现在G- i中g的推导中，根据引理 1 Y —> X。因此h: Y —> B 在Gz中 ，其中 X —> Y 和 Y —> X ，完成证明。□

引理 2 不能被强化，使得 X = Y。也就是说，可以有两个具有等效闭包的非冗余覆盖，使得这两个覆盖具有代表关键等价类的不同左侧。例如，在图 4 中，g 3和h)具有功能相同的左侧，因为 CE <-> DE，但 CE ^ DE。

使用引理 3 并认识到算法 2 从具有功能等效左侧的每个最大 FD 组合成一个关系，我们现在可以看到给定 FD 集的所有非冗余覆盖通过算法 2 产生相同数量的关系。

定理 4.令 F 为 FD 的集合。F 的任何两个非冗余覆盖将通过算法 2 产生相同数量的关系。

证明。设Gi和 G' 2是F的两个非冗余覆盖。根据引理 3，如果 FD g: X —>A 在Gi中，则在Gz中存在 f i : Y —> B，其中 X —> Y 且是——>X。因此，对于Gi中具有功能等效左侧的任何 FD 组，Gz中一定存在一个这样的组，即具有相同功能等效左侧的组。由于每个这样的组生成一个关系，因此G'i 和Gz必须生成相同数量的关系。□

定理 4 指出，就合成关系的数量而言，所有非冗余覆盖的选择都同样好。这有点令人惊讶，因为它与直觉相矛盾，即最小尺寸的非冗余覆盖可能会比其他较大的非冗余覆盖产生更少的关系。

该定理还表明，在逻辑层面上，对于如何选择覆盖给定 FD 集合的关系，没有太多选择。一些分解方法（例如 [10,11,12]）声称允许系统在一类可能的模式中进行选择，通过效率考虑来指导选择。由于所有覆盖都有相同的键等价类集，因此可能的模式类实际上非常小。因此，如果一个人在逻辑层面上受到规范化考虑而不是效率考虑的指导，那么就会得到一组几乎相同的可能模式。

从定理 4 可以看出，算法 2 生成的关系数量在所有包含相同给定 FD 集合的关系中是最少的。这给出了[8]中讨论的最优 3NF 模式的完整特征。

推论1.令S为通过使用算法 2 从一组 FD F 合成的模式。令$' 为包含覆盖 F 的一组 FD G 的任何模式。然后 |s' |> |§|。

证明。令H c G为F的非冗余覆盖。当然 ，通过算法 2， H不会生成比s' 中更多的关系。此外，根据定理 4，算法 2 将从G生成与从F生成的关系数量相同的关系。S'| > | S|。

$$G = \{g_1: \mathrm{C} \rightarrow \mathrm{D},\ g_2: \mathrm{D} \rightarrow \mathrm{C},\ g_3: \mathrm{CE} \rightarrow \mathrm{F}\} \qquad H = \{h_1: \mathrm{C} \rightarrow \mathrm{D},\ h_2: \mathrm{D} \rightarrow \mathrm{C},\ h_3: \mathrm{DE} \rightarrow \mathrm{F}\}$$

G and H are nonredundant and $G^+ = H^+$. However, g_3 and h_3 generate different relations. This is an example of Lemma 3 where $\mathrm{X} \neq \mathrm{Y}$.

Fig. 4. Two equivalent coverings with different keys

8 结论

本文的目的是开发一种算法，用于从给定的 FD 集合合成 3NF 模式，并检查此类模式的一些属性。主要结果是:

(1)某些用于合成模式的简单算法要么产生太多关系，要么违反 3NF。

(2)提出了一种合成可证明的3NF模式的算法。该算法的本质在于它从给定的 FD 集中消除了尽可能多的冗余。

(3)当使用后一种方法时，所有非冗余覆盖都会产生相同数量的关系。因此，综合模式包含最少数量的关系。

据我们所知，这是首次成功尝试可证明且有效地实施 Codd 标准化程序 [8]。（早期两次类似尝试中的错误已被隔离。）此外，根据推论 1，综合关系满足 Codd 的最优性标准——覆盖相同 FD 的其他模式没有更少的关系。

9 致谢

我非常感谢 C. Beeri 帮助我重新思考和扩展本文 [3] 的结果。特别是算法 1 的模式和实施例以及步骤 1 的定义均归功于他。
我要感谢 JM Smith 发现了我最初对非功能性关系的处理中的错误，并指出了早期草案中许多不透明的部分。我还要感谢 HA Schmid、K. Sevcik、JR Swenson 和 D. Tsichritzis对于本研究的许多结果的方法和意义进行了许多有益的讨论。最后，我要感谢审稿人为提高终稿的可理解性提出的许多有益的建议。