第12章 进一步规范化II：高级范式

12.1 引言

在前一章讨论了一直规范到Boyce/Codd范式的规范化思想（这是函数依赖的概念所能达到的最高层次）。现在讨论4NF和5NF来结束这一部分的内容。正如下面将要看到的，4NF的定义利用了一个新型的依赖，称为多值依赖（MVD）。多值依赖是广义的函数依赖。同样，5NF的定义应用了另一种形式的依赖，称为连接依赖（JD），连接依赖也是一种广义的多值依赖。在第12.2节将讨论MVD和4NF，第12.3节讨论JD和5NF（在该节中还解释了5NF在某种特定意义上被称为是最终的规范化形式的理由）。值得说明的是，对MVD和JD的讨论没有第10章对函数依赖的讨论那么正式和完整。这是有意为之，正式的讨论留给研究论文去完成（见“参考文献”）。

第12.4节回顾了全部的规范化过程，并且对此作了一些附加的介绍。第12.5节简单地讨论了范式逆规范化的概念。12.6节描述了另一个设计原则：正交设计。第12.7节是对今后在规范化领域的一些可能研究方向的简单介绍。12.8节是全章的小结。

12.2 多值依赖与第四范式

假定有一个关系变量HCTX（H代表层次），其中包括COURSES、TEACHERS和TEXTS信息。在此变量中与TEACHERS和TEXTS相对应的属性是关系值属性（参考图12-1中的HTCX值的例子）。可以看到，每一个HCTX元组包括一个COURSE（课程名）、一个包括TEACHER名的关系以及一个包括TEXT名的关系（在表中列出了两个这样的元组）。建立这样的元组的意图是指定的课程COURSE可以为任意一个指定的教师TEACHER所教，并可以用任意一本指定的教材TEXT作为参考书。假定对于一个给定的课程，可以存在任意数量的相应教师和任意数量的相应教材。而且，还假定——这种假定可能并不实际——教师与教材之间是彼此相对独立的。这就是说，无论哪位教师教指定的哪一门课程，都可以用同一本教材。最后还认为一个指定的教师和一本指定的教材是可以与任意数量的课程相联系的。

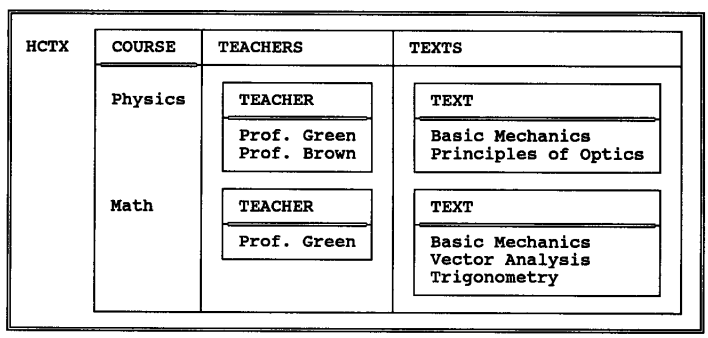

图12-1 关系变量HCTX的值的例子

现在假设（与在前一章的11.6节相同）要去掉关系值属性。一种办法——与在习题11.3中所描述的解答方式（也是在本节结尾将要讨论的观点）不同——是简单地将关系变量HCTX替换为一个有三个标量属性COURSE、TEACHAR和TEXT的关系变量CTX，如图12-2所示。正如在图中可以看到的，每一个HCTX中的元组都在CTX中引发出$m*n$个元组，其中m和n是HCTX中TEACHERS和TEXTS关系的势（cardinalities）。值得注意的是所得到的的关系变量CTX是“全码”（然而在HCTX中只有单一的候选码，即，COURSE）。

CTX

COURSE	TEACHER	TEXT
Physics	Prof. Green	Basic Mechanics
Physics	Prof. Green	Principles of Optics
Physics	Prof. Brown	Basic Mechanics
Physics	Prof. Brown	Principles of Optics
Math	Prof. Green	Basic Mechanics
Math	Prof. Green	Vector Analysis
Math	Prof. Green	Trigonometry

图12-2 与图12-1中HCTX值相对应的关系变量CTX的值

关系变量CTX的含意基本上可以归纳为：一个元组{COURSE：c，TEACHER：t，TEXT，x}在CTX中出现，当且仅当课程c可以为教师t所教授，并且应用教材x为参考书。有鉴于此，对于一门给定的课程，所有的教师和教材的可能组合情况都可能出现。这就是说，CTX满足如下关系变量的约束条件：

如果元组（$c,t1,x1$），（$c,t2,x2$）都出现，那么元组（$c,t1,x2$），（$c,t2,x1$）也都出现（在此又用到了元组的简单表示法）。

很显然，在关系变量CTX中存在大量冗余，通常这会导致一定的更新异常。举例来说，如果添加物理课可以由一个新教师教的信息，就必须要添加两个元组，对每一个教材都要添加一个。能避免这个问题吗？不难看出：

1) 所讨论的这个问题是由教师和教材的彼此完全独立引起的。
2) 如果CTX被分解为两个相互独立的投影——称为CT和CX——{COURSE，TEACHER}和{COURSE，TEXT}，情况就会好得多（参见图12-3）。

CT

COURSE	TEACHER
Physics	Prof. Green
Physics	Prof. Brown
Math	Prof. Green

CX

COURSE	TEXT
Physics	Basic Mechanics
Physics	Principles of Optics
Math	Basic Mechanics
Math	Vector Analysis
Math	Trigonometry

图12-3 与CTX的值相对应的关系变量CT和CX值

如果增加一个物理课新教师的信息，现在我们所要做的就是在关系变量CT上插入一个单独的元组（还要注意关系变量CTX可以由CT和CX重新合并得到，因此这样的分解并没有损失）。因此，这种思想看来确实很合理，即对于像CTX这样的关系变量应该有一种“进一步规范化”的方法。

注意：在这一点上，可能有两种看法，第一种认为在CTX中存在的冗余是必要的，并且因此认为相应的更新异常是没有必要出现的。另一种是，在更特殊的情况下，在CTX中对每

一个给定的课程，并不是都需要包括所有可能的教师和教材的组合。例如：两个元组显然足以显示物理课有两个教师和两本教材。但问题是，究竟是用哪两个元组呢？任何一个具体的选择都会引起关系变量中不明显的解释和奇怪的更新操作（试一试去标明这样一个关系变量中的谓词！——例如，试着找出可以决定在这个关系变量中一些更新操作是否是一个可以接受的操作的标准）。

因此，由于不正规，CTX的设计明显是不好的，而分解为CT和CX就好多了。然而，问题是以上这些情况在正式的条件下并不很明显。特别是在CTX根本不满足任何函数依赖的情况下（除了一些平凡的函数依赖，如COURSE→COURSE），事实上，CTX是属于BCNF范式的，因为已经是全码——所有的含全码的关系变量都包含在BCNF中（注意到CT和CX这两个投影也是全码，因此它们也属于BCNF范式）。因此前一章的问题对我们这一章的讨论没有任何帮助。

像CTX这样的BCNF关系变量“问题”的存在很早就为人们所发现，并且处理它们的方法也很快就为人们所理解，至少是在直观上。然而，直到1977年Fagin的多值依赖概念的提出，这些直观上的观点才因此有了令人信服的理论根基。多值依赖是一种一般化的函数依赖，因而每一个FD都是MVD，但反之则不正确（例如上面的MVD就不是FD）。在关系变量CTX中，有两个MVD如下：

```
COURSE ↠ TEACHER
COURSE ↠ TEXT
```

注意这里用双箭头，多值依赖A↠B称为“B多值依赖于A”，或者说，“A多值决定B”，让我们来看一下第一个MVD，即COURSE↠TEACHER。直观地说，MVD就意味着虽然一门课程并不对应一个相应的教师——例如并不存在函数依赖COURSE→TEACHER——但每一门课对于每一个相应的教师确实有一个定义得很好的集合。这里所说的“定义得很好”的含义，是指对于一门给定的课程c和一本给定的教材x，教师t的集合是否与CTX中的（c,x）对匹配，只依赖于c的值，而与选择哪一个x值无关。第二个MVD，COURSE↠TEXT，也可以有类似的解释。

下面我们给出正式的定义：

多值依赖：R是一个关系变量，A、B和C是R的属性的子集。那么我们说B多值依赖于A——符号如下：$A\twoheadrightarrow B$（读做“A多值决定B”，或简单地称为“A双箭头B”）——当且仅当对于每一个可能的合法R值，B值的集合对于给定的一组（A值，C值）只依赖于A的值，而与C的值无关。

很容易看出——参见[12.13]——对于给定的变量$R\{A,B,C\}$，多值依赖$A\twoheadrightarrow B$存在，当且仅当多值依赖$A\twoheadrightarrow C$也存在。这样MVD总是成对的一起出现。因此通常用一种语句来表示它们：$A\twoheadrightarrow B|C$。例如：COURSE↠TEACHER | TEXT。

在前面我们已经提到，多值依赖是一般化的函数依赖，在这种意义上讲每一个FD都是MVD。更精确地说，一个FD就是一个只有一个依赖值（右边的）与一个给定的决定值相符合的MVD。因此，如果$A\rightarrow B$，那么一定$A\twoheadrightarrow B$。

回到我们原来的CTX问题，现在可以看到像CTX这样的关系变量的问题是：它们包括不是FD的MVD（在这种问题并不明显的情况下，可以指出，正是MVD的存在使得这种问题存在——举例来说——增加一个教师要插入两个元组。为了保证MVD所提供的完整性约束，需要插入两个元组）。CT和CX这两个投影并不包括任何这样的MVD，它们完善了原来的设计。

因此我们用这两个投影来代替 CTX，由[12.13]Fagin证明的一个重要的定理为我们做这样的替换提供了依据：

定理（Fagin）：假定$R\{A,B,C\}$是一个关系变量，其中A、B和C都是属性集。那么R等同于它在$\{A,B\}$和$\{A,C\}$上的投影的组合，当且仅当R满足多值依赖$A\rightarrow\rightarrow B|C$。

（注意：这是一个在第 11章定义的Heath定理的更高级的版本）根据 Fagin的理论，我们现在就可以定义第四范式（这是因为 BCNF范式在第11章中依然被称为第三范式）：

第四范式：只要存在R的属性的子集A和B，满足非平凡 ⊖ 的多值依赖，并且 R的所有属性也都函数依赖于A，这样的关系变量R满足4NF。

换句话说，在R中的唯一的非平凡的依赖（函数依赖或多值依赖）是 $K\rightarrow\rightarrow X$形式（例如：一个超码$K$对另一个属性$X$的函数依赖）。同样，如果$R$是BCNF，并且$R$中的所有非平凡的多值依赖事实上都是“ 非码函数依赖（ FDs out of key）”，则R是4NF的。因此特别要注意的是，4NF包含了BCNF。

既然关系变量CTX包括了根本不是FD的MVD，它就不是4NF，更不用说非码函数依赖了。然而，它的两个投影 CT和CX都是4NF。因此，4NF是BCNF的一种改进，因为 4NF消除了一种并不受欢迎的依赖。而且， Fagin在文献[12.13]中还提到了4NF通常是可以实现的，这就是说，任何关系变量都可以无损失分解为相应的 4NF关系变量的集合 —— 虽然在11.5节中对SJT示例的讨论表明，在某些情况下分解得如此之细并不必要（甚至深化到 BCNF都不必要）。

注意：在11.6节谈论的Rissanen的投影独立理论虽然是以函数依赖为依据的，但是也适用于多值依赖。前面也谈及一个满足函数依赖 $A\rightarrow B$和$B\rightarrow C$的关系变量$R\{A,B,C\}$，将其分解为在$\{A,B\}$和$\{B,C\}$上的投影要好于分解为 $\{A,B\}$和$\{A,C\}$的投影。如果我们将这里的函数依赖替换为多值依赖，这个定理依然成立。

总结一下这一节，回到消除关系值属性（关系值属性简称为 RVA）的问题，特别是回到在上一章的练习 11.3的答案中所描述的进行这种消除操作的过程。观点如下，实际上要达到4NF所要做的就是：如果开始时一个关系变量中存在两个或多个相互独立的关系 RVA，那么所要做的第一件事就是将这些 RVA分开。这个规则不仅直观上看很有意义，而且在练习 11.3的答案中也正是这么做的。例如，在关系变量 HCTX的例子中，第一件事就是将原来的关系变量替换为它的两个投影HCT{COURSE，TEACHERS}和HCX{COURSE，TEXTS}（TEACHERS和TEXTS仍旧是RVA）。然后在这两个投影中的 RVA可以用通常的方式加以消除（并且投影被还原为BCNF），而且满足BCNF范式的关系变量CTX不会产生这种问题。正是 MVD和4NF为上述的分解提供了一个理论基础，否则这将只是一种单凭经验而作出的结论。

12.3 连接依赖与第五范式

到目前为止，在本章(并且在前一章)中都默认，在进一步规范化过程中存在并且是必要的唯一的操作是用一种无损方法将一个关系变量用它的两个投影来代替。这种方法成功地进行到了第四范式。因此，如果发现存在关系变量不是无损分解为两个投影而是无损分解为三个投影或更多，可能会觉得很奇怪。这样的关系变量可描述为“ 分解 ”（$n>2$） —— 这表示关系变量可以被无损分解为 n个投影，而不是为 m个投影，对于任意 $m<n$。一个关系变量可以被无

⊖ 当A是B的一个子集或A和B的并集是全集时，多值依赖$A\rightarrow\rightarrow B$是平凡的。

损分解为两个投影，则称之为“可以2分解”的。注意：当n>2时的n分解现象是由Aho、Beeri和Ullman提出的[12.1]，而Nicolas也研究了n=3时的情况[12.25]。

考虑从供应商-零件-项目数据库中提取的关系变量 SPJ（由于QTY属性简单，因此忽略QTY属性）；它的一个简单的值在图 12-4的上半部表示出来。可以看到关系变量 SPJ是全码，并且根本不包含非平凡的函数依赖或多值依赖，因此 SPJ是4NF。可以看到图12-4还显示了：

a. 与显示在图顶端的SPJ的相关值相对应的三个二元投影 SP、PJ和JS；

b. 将SP和PJ投影连接起来的结果（通过 P#）；

c. 将上一步的结果和JS投影相结合的结果（通过 J#和S#）。

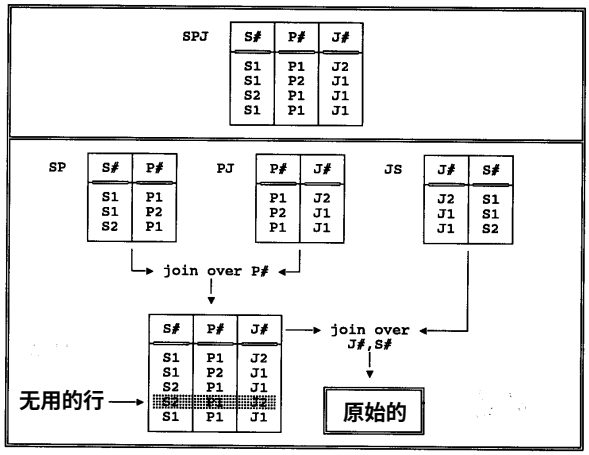

图12-4 关系SPJ是三个二元投影的连接而非其中任何两个的连接

可以观察到第一个连接的结果是产生一个原始 SPJ关系的副本加上一个多余的元组，第二个连接的结果是消除多余的元组，从而复原为原始的 SPJ关系。换句话说，原始的 SPJ关系是可3分解的。注意：无论我们选择哪两个投影作为第一次连接，结果都是一样的，虽然在每种情况下中间结果不同（练习：证明这一命题）。

现在，图12-4的例子中是给出关系，而不是关系变量。然而， SPJ的可3分解性是基本的、与时间无关的特性——即，它是关系变量的所有合法值满足的特性——如果这个关系变量满足一个特定的与时间无关的完整性约束。要理解这个完整性约束的意义，必须首先看到“ SPJ与三个投影SP、PJ和JS的连接等价”的说法与下面的说法完全相同：

如果（$s1$，$p1$）在SP中出现，

并且（$p1$，$j1$）在PJ中出现，

并且（$j1$，$s1$）在JS中出现，

那么三元组（$s1$，$p1$，$j1$）在SPJ中出现。

因为三元组（$s1$，$p1$，$j1$）显然出现在SP，PJ和JS的连接中(反之，如果（$s1$，$p1$，$j1$）出现在SPJ中，那么（$s1$，$p1$）出现在投影SP中，等等。这对于任意的三元关系 SPJ来说都是正确的)。对于一个 J2，（$s1$,$p1$）出现在 SP中，当且仅当（ $s1$,$p1$,$j2$）出现在 SPJ中，并且对于（$p1$,$j1$）和（$j1$,$s1$）的情况也类似，那么我们就可以将上面的综述作为 SPJ的一个约束。重新复述如下：

如果（$s1$,$p1$,$j2$），（$s2$,$p1$,$j1$），（$s1$,$p2$,$j1$）在SPJ中出现，那么（$s1$,$p1$,$j1$）也在SPJ中出现。

如果这种说法在任何情况下都成立——例如，对于关系变量SPJ的所有可能的合法值——那么这个关系变量确实存在一个与时间无关的约束（虽然是一个很奇异的约束）。可以看到这个约束的循环性（“如果*s1*与*p1*连接，*p1*与*j1*连接，而*j1*又与*s1*连接，那么*s1*、*p1*和*j1*一定是在一个元组中同时出现的”）。当*n*>2时，一个关系变量是可*n*分解的，当且仅当它满足这样的n路的循环约束。

于是可以认为，关系变量SPJ确实满足时间无关约束（图12-4中的例子的值是符合这一假设的）。可以将这种约束简化为3D约束（3D代表可3分解的）。在现实条件下，3D约束意味着什么呢？通过一个例子可以将其具体化。这个约束表明，在所假设的关系变量SPJ表示的那部分现实世界当中，如果（例如）

a. 史密斯提供活动扳手，并且

b. 活动扳手在曼哈顿项目中使用，并且

c. 史密斯为曼哈顿项目提供支持，

那么

d. 史密斯提供活动扳手给曼哈顿项目。

注意：（正如在第1章的1.3节所指出的）*a*、*b*和*c*一起出现通常并不表示*d*也一定出现；事实上，这个例子在第1章中是作为“连接陷阱（connection trap）”的一个说明提出的。然而，在现在的例子中并不存在陷阱——因为并不存在一个附加的现实世界约束在起作用，即3D完整性，使在这一个特殊的例子中从*a*、*b*和*c*中推演出*d*有效。

回到所讨论的主题：因为3D完整性被满足当且仅当所考虑的关系变量与它的特定的投影的连接结果等价，所以这种约束被称为连接依赖（JD）。当MVD或FD是某关系变量中的约束时，JD才是其中的约束。下面是其定义：

连接依赖：如果*R*是一个关系变量，并且*A*,*B*...,*Z*都是*R*的属性的子集，那么称*R*满足JD ⋈ {*A*，*B*，...，*Z*}（读做星*A*，*B*，...，*Z*）当且仅当*R*的任何可能出现的合法值都与它在*A*，*B*，...，*Z*上的投影的连接等价。

例如，如果用SP来表示SPJ的属性集的子集{S#，P#}，同样也用PJ和JS来表示SPJ的另两个属性集，那么关系变量SPJ满足JD ⋈{SP，PJ，JS}。

那么，可以看到拥有JD ⋈{SP，PJ，JS}约束的关系变量SPJ，是可3分解的。但问题是，应该让它可3分解吗？答案是“有可能”。关系变量SPJ（存在JD约束）存在一大堆的关于更新操作的问题，当它被3分解之后问题就被解决了。这样一些问题的例子在图12-5中表示。而在3分解之后将会有什么情况发生作为一个练习留给大家思考。

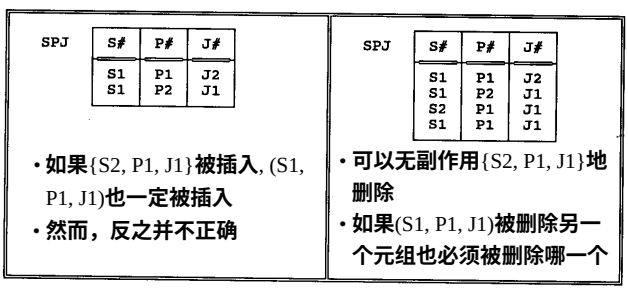

图12-5 在SPJ中的更新问题的例子

Fagin的定理（在12.2节中已经讨论）——*R*{*A*，*B*，*C*}可以被无损分解为它在{*A*,*B*}和

{*A*,*C*}上的投影，当且仅当在*R*中存在多值依赖*A*→→*B*和*A*→→*C* —— 可以被重述如下：

R{*A*，*B*，*C*}满足JD ⋈{*AB*，*AC*}，当且仅当它满足多值依赖*A*→→*B*|*C*。

既然这个定理可以被用做多值依赖的定义，这就表明多值依赖是一种特殊形式的连接依赖，或者说（等价）连接依赖是一种一般化的多值依赖。

形式化地表示，可以有

A→→*B*|*C*≡ ⋈ {*AB*，*AC*}

注意：从定义立即可以看到，连接依赖是可能的依赖中的最一般化的一种形式（在某种特殊的意义上使用了术语“依赖”）。这就是说，只要把对依赖的理解限制在，对关系变量通过投影来分解、通过连接来重组的处理框架内，就不存在一种更高形式的依赖，使得连接依赖仅仅是这种更高形式的依赖的特殊情况（然而，如果允许通过其他的分解和重组操作，那么其它类型的依赖就可能会出现。这种可能性将在第 12.7节加以简单地讨论）。

回到前面的例子中可以看到，关系变量 SPJ的问题是它包括不是多值依赖（当然也不是函数依赖）的连接依赖（练习：为什么这是一个问题？）。可以看到，将这样一个关系变量分解为更小的部分 —— 即分解为由连接依赖指定的投影是可能的，而且也许是希望得到的。这种分解过程可以重复下去，由此得到的关系变量称为第五范式。其定义如下：

第五范式：一个关系变量*R*是第五范式 —— 也称为投影-连接范式（PJ/NF） —— 当且仅当*R*的每一个非平凡的 ⊖ 连接依赖都被*R*的候选码所蕴涵。

注意：下面解释一下对于一个 JD“被候选码所蕴涵”的含义。

关系变量SPJ并不是5NF；它满足一个特定的连接依赖，即 3D约束。这显然没有被其唯一的候选码（这个候选码是其所有的属性值的组合）所蕴涵。可以表示其区别如下：关系变量 SPJ并不是5NF，因为（a）它是可以被3分解的；（b）可3分解性并没有为其{S#，P#，J#}是一个候选码的事实所蕴涵。相反， 3分解后，由于三个投影 SP、PJ和JS根本不包括任何（非平凡的）连接依赖，因此它们都是 5NF。

虽然现在可能还不是很明显 —— 因为还没有解释JD被候选码所蕴涵的含义 —— 事实上，任何属于5NF的关系变量也都是自动属于 4NF的，因为（如我们所见的）多值依赖是连接依赖的一种特殊情况。事实上，在文献[12.14]中，Fagin提出为候选码所蕴涵的多值依赖必须是一个函数依赖，在这个函数依赖中，候选码是决定因素。在[12.14]中，Fagin还提出任何给定的关系变量都可以被无损分解为一个相等的第五范式关系变量集合，这就是说，5NF是可以达到的。

现在解释一下连接依赖为候选码所蕴涵的含义。首先考虑一个简单的问题。假定（如在第11章中11.5节中所做的）供应商关系变量 S有两个候选码，{S#}和{SNAME}，那么关系变量满足几个连接依赖 —— 例如，它满足连接依赖

⋈{{S#，SNAME，STATUS}，{S#，CITY}}

这就是说，关系变量S是和它在{{S#，SNAME，STATUS}和{S#，CITY}上的投影的连接相等的。并且因此能被无损分解为这些投影（这当然并不表示它应该被分解，而只是表示可以如此）。这个连接依赖被 {S#}是一个候选码的情况所蕴涵（事实上它是被 Heath的定理所蕴涵，参考[11.4]）。同样，关系变量S也满足连接依赖

⊖ 连接依赖 ⋈{A, B, ..., Z} 是平凡的，当且仅当投影 A, B, ..., Z是R 的标识(identity)投影(即是R所有属性的投影)。

```
*{{S#, SNAME}, {S#, STATUS}, {SNAME, CITY}}
```

这个连接依赖是由{S#}和{SNAME}都是候选码的事实所蕴涵的。

正如前述的例子所表明的，一个给定的连接依赖 *{A，B，...，Z}是由候选码所蕴涵的，当且仅当每一个A，B，...，Z都是所讨论的关系变量的超码。因此，给定一个关系变量 R，只要知道R中所有的候选码和所有的连接依赖就可以分辨出 R是否是第五范式。然而，所有的连接依赖可能自身是一个非平凡的操作。这就是说，尽管相对来说分辨函数依赖和多值依赖是很简单的（因为它们是相当直来直去的现实世界的解释），但对那些既不是多值依赖也不是函数依赖的连接依赖却并非如此 —— 因为连接依赖的意义可能并很直观。因此确定一个给定的关系变量是4NF而不是5NF，并且决定是否因此可以通过分解逆规范化的方式来规范化此关系变量的过程依然不清楚。经验表明这样的关系变量是病态的，而且在实际中是很少见的。

综上所述，考虑到投影和连接，从定义上看第五范式是最终的范式（这也解释了 5NF的另一个名字，即投影-连接范式）。这就是说，一个 5NF的关系变量是通过投影的方式来保证没有不规则的情况[⊖] 。因此，如果一个关系变量是 5NF，那么所有的连接依赖都是由其候选码所蕴涵的，正确的分解方式就是基于这些候选码的（在这样的分解中，每一个投影都包含一个或多个这样的候选码再加上零个、一个或多个其它的属性）。例如，供应商关系变量 S是属于5NF的。正如前面所述的，它可以通过几种无损的方式被进一步分解，在这种分解中的每一个投影都将包含一个原来的候选码，并且在进一步分解中并没有任何特别的优点。

12.4 规范化过程小结

到现在为止，在本章（以及前一章）中，作为数据库设计的目标，已经讨论了无损分解的技术。基本的观点如下：给定某一 1NF的关系变量和R的一些函数依赖、多值依赖和连接依赖，系统地将R简约为一组“更小”（例如，度更低）的关系变量，这些关系变量在一定的意义上是与R相等的，但在某些方面较之 R更优（原来的关系变量 R应该是已经消除了特定的关系值属性而得到的，如在 12.2节讨论过的）。简约过程的每一步都包括由前一步的结果来投影关系变量。在每一步中用给定的约束来指导下一步投影的选择。整个过程可以为一组规则，如下：

1) 对原始的1NF的关系变量投影，消除任何不可约的函数依赖。这一步将产生一个 2NF关系变量集合。
2) 对2NF的关系变量投影，消除任何可以传递的函数依赖。这一步将产生 3NF关系变量的集合。
3) 对3NF的关系变量投影，消除任何决定方不是候选码的函数依赖。这一步将产生 BCNF关系变量的集合。

   注意：规则1~3可以合并为一个规则，即“对原始的关系变量投影，消除所有的决定因素不是候选码的函数依赖”。
4) 对BCNF的关系变量投影，消除任何不是函数依赖的多值依赖。这一步将产生 4NF的关系变量的集合。注意：在实践中通常是 —— 通过“分离独立的 RVA”的方式来实现，正如在12.2节中对CTX的例子的讨论中所解释的 —— 消除这样的多值依赖之后再应用

⊖ 当然，这句话并不是说5NF对所有可能的异常都有效，而是说它对于可被投影所消除的异常免疫。

以上的规则1~3。

5) 对4NF的关系变量再投影，如果存在任何不被候选码所蕴涵的连接依赖，则加以消除。

从前面的小结中提出以下几点：

1) 首先，在每一步进行的投影过程必须是基于一种无损的方式，并且最好是用保持原有依赖的方法来投影。

2) 观察到（Fagin首先提到，参考[12.14]）在BCNF、4NF和5NF的定义中有一种非常引人注目的对应关系，即：

- 一个关系变量*R*是BCNF，当且仅当*R*中的每一个函数依赖都被*R*的候选码所蕴涵。
- 一个关系变量*R*是4NF，当且仅当*R*中的每一个多值依赖都由其候选码所蕴涵。
- 一个关系变量*R*是5NF，当且仅当*R*中的每一个连接依赖都由其候选码所蕴涵。

在第11章以及在本章的前几节中讨论的更新异常的问题正是由于那些不被候选码所蕴涵的FD、MVD和JD所引起的。

3) 规范化过程的总体目的如下：

- 消除某些冗余。
- 避免更新异常。
- 产生一种可以很好代表现实世界的设计——一个很直观并且方便未来扩充的设计。
- 可以简单地满足某些完整性约束。

我们对以上列出的最后一条稍作详细的介绍。通常的观点是（与本书中第8章，第10章中提到的一样）一些完整性约束隐含其它的完整性约束。举一个例子，一个工资必须大于10 000元的约束肯定是蕴涵工资大于0的约束的。现在，如果完整性约束*A*蕴涵*B*，那么满足*A*，则一定自动满足*B*（甚至没必要明显地标明*B*完整性，除非是用注释的形式）。规范到5NF的规范化提供了一种满足某些重要而又经常发生的约束的简单方法；所需要的工作就是满足候选码的唯一性，这样所有的连接依赖就会被自动满足——因为所有的这些连接依赖（和多值依赖以及函数依赖）都会被候选码所蕴涵。

4) 这里再一次强调规范化指南只是指南而已，有时偶尔也可能有理由不完全规范化。这种情况的经典例子是包括姓名和地址的关系变量NADDR（参考第11章练习11.7）——虽然，坦白地讲，这个例子并不具有说服力……凭经验来说，并非所有的规范到底的规范化都是不好的。

5) 再重复一下，第11章中关于依赖和进一步规范化的概念本质上都是语义的概念——换句话说，它们关注数据的意义。相反，基于这些形式的关系代数、关系演算以及像SQL这样的语言，却只关注实际的数据的值；它们除了要求满足1NF外并不要求任何特定级别的规范化：进一步规范化方针应当主要被看作是辅助数据库设计的规则（因此对用户也有帮助）——这种规则帮助设计者用一种简单而又易懂的方式来对现实世界的一部分进行语义描述，虽然这部分是很小的。

6) 承接前面的观点：规范化的思想对数据库的设计是有用的，但它们并不是秘方良药。以下是几点原因（在[12.9]中有详细介绍）：

- 规范化确实可以协助满足某些完整性约束并使其过程变得简单，但是（如在第8章所述）连接依赖、多值依赖和函数依赖并不是唯一可以在实践中应用的约束。
- 分解方法可能并不是唯一的（事实上，通常有许多种将一组给定的关系变量规范为

5NF的方法），并且还有几个用来选择分解方法的有针对性的标准。

- 根据11.5节的解释（“SJT问题”），BCNF范式和保持依赖的目标可能是有冲突的。
- 规范化过程通过投影消除了冗余，但是并不是所有的冗余都是可以用这种方式来消除的（“CTXD问题”——参看[12.13]中的解释）。

无论如何应该指出，好的自上而下的设计方法都趋向于充分规范化（fully normalized）的设计（参见13章）。

12.5 逆规范化

根据本章（以及前一章中）的观点，我们理所当然地应该全面规范到5NF。然而在实践中，为取得好的性能，“逆规范化”又是必须的。这个观点的内容如下：

1) 完全的规范化表示会出现许多逻辑上相分离的关系变量（并且我们这里假定所讨论的关系变量是特定的基本关系变量）；
2) 许多逻辑上分离的关系变量表示许多物理上分离的存储文件；
3) 许多物理上分离的存储文件表示会有许多I/O操作。

严格地说，这种观点当然是不合理的，因为（在本书的其它部分有论述）关系模型并没有保证基本关系变量必须一一映射到存储文件中。如果必要的话，逆规范化应该在存储文件的层次上操作而不是在基本关系变量的层次上操作。但是这种观点对现在的SQL产品来讲是有效的，因为在这些产品中存在着不同级别的不完全分离。因此在这一节，我们仔细地看一下“逆规范化”的概念。

注意：下面的讨论是基于[12.6]的材料的。

1. 什么是逆规范化

简单地回顾一下：规范化一个关系变量 *R*就意味着将*R*用一组投影*R1*，...，*Rn*来代替。因此，对于关系变量*R*的任何可能的值*r*，如果投影*R1*，..., *Rn*的相应的值*r1*，...，*rn*重新连接，那么这个结果保证是与*r*相等的。总体的目标是通过确定每一个投影 *R1*，..., *Rn*都处于最高级的规范化的级别来减少冗余（例如，5NF）。

现在我们可以定义逆规范化概念如下。设 *R1*，...，*Rn*是一组关系变量。逆规范化这些关系变量就意味着将它们替换为它们的连接，因此对于 *R1*，...，*Rn*的每一个可能的值*r1*，...，*rn*，通过*Ri*的属性值投影 *R*的相应值*r*时保证还会产生 *ri*（*i*=1，...，*n*）。总体的目标是增加冗余，通过确认*R*是比关系变量*R1*，...，*Rn*更低的规范化级别来完成。更具体地说，目的是在数据库设计中事先建立一些连接，从而减少执行中的连接代价。

通过一个例子，我们可以考虑逆规范化关系变量零件和发货量来产生一个关系变量 PSQ，如图12-6所示。可以看到关系变量PSQ是1NF而不是2NF。

PSQ

P#	PNAME	COLOR	WEIGHT	CITY	S#	QTY
P1	Nut	Red	12.0	London	S1	300
P1	Nut	Red	12.0	London	S2	300
P2	Bolt	Green	17.0	Paris	S1	200
..	.....	.....	.....	.....	..	...
P6	Cog	Red	19.0	London	S1	100

图12-6 逆规范化零件与货运量

2. 一些问题

逆规范化的概念引起许多很著名的问题。一个很明显的问题是，一旦我们开始逆规范化时，并不清楚在哪里终止……对于规范化来说，有很合情合理的原因使其一直继续到可能的最高的规范化形式；在逆规范化中我们是否要达到最低的的规范化形式呢？当然不是；然而却并没有合理的标准去确切地决定在哪里停止。换句话说，在选择逆规范化时，没有什么固定的科学和合理理论的支持，而是通过纯粹的经验和主观决定的。

第二个明显的问题是存在冗余和更新问题，这正是因为所处理的关系变量是没有完全规范化的关系变量。这些问题已经详细地讨论过了。此外，还可能会有检索问题；这就是说，逆规范化实际上使查询更加难以表示。例如：考虑“对于每一种零件颜色，都给出它们的平均重量”的查询。如果使用通常的规范化的设计，一个适当的公式是：

```
SUMMARIZE  P  PER  {COLOR} ADD  AVG (WEIGHT)  AS  AVWT
```

然而给定如图12-6中的逆规范化设计，这个公式就有些蹊跷（更不必说它是依赖于假定——通常说是不合理的——每一个零件确实有至少一个发货量与其对应）：

```
SUMMARIZE  PSQ  {P#,COLOR,WEIGHT} PER  PSQ {COLOR}
                                ADD  AVG {WEIGHT}  AS  AVWT
```

(注意：这一公式很可能执行起来效率很差)。换句话说，就可用性和性能两方面因素考虑，通常对于逆规范化是“对检索有利而对更新不利”的普遍理解是错误的。

第三点，也是最主要的原因，问题如下（这一点适用于“适当”的逆规范化——即，只在物理层进行的逆规范化——也适用于那些在当今的SQL产品中有时必须作的逆规范化）：当提到逆规范化“对性能有益”时，实际意味着它是对具体的应用的性能有益。任何给定的物理设计都是对某些应用有益而对其它的没好处（根据性能来说）。例如，假定每一个基本关系变量确实映射到一个物理存储文件，并且还假定每一个物理存储文件包括一组物理上相邻的存储记录，每一个记录对应一个相应的关系变量上的元组。那么：

- 假定将供应商、发货量和零件的集合作为一个基本关系变量，并因此给出一个存储文件。那么“取出提供红色零件的供应商的详细信息”的查询可能会在这种物理结构下运行得较好。
- 然而，“取出在伦敦的供应商的详细信息”的查询在这种物理结构下，将比存在三个基本关系变量并且将它们映射到三个物理上分离的存储文件中的结果要差。原因是在后面的设计中，所有的供应商的信息将被在物理上邻接地存储，然而在前面的设计中，它们在物理上分散于一个较大的空间，因此将需要更多的I/O操作。

类似的评论也适用于任何其它的只访问供应商、只访问零件或只访问货运量的查询，而不适用于一些连接的查询。

12.6 正交设计

在这一节我们简要地讨论另外一种数据库设计的原理，一个在本质上不是进一步规范化的设计原理，但它确实与规范化很相像，因为它是很科学的。它称为正交设计准则(The Principle of Orthogonal Design)。图12-7中显示了一种对供应商的设计，它显然不够好，但是很可能出现；在这个设计中，关系变量SA满足那些居住在巴黎的供应商，关系变量SB满足那些或不居住在巴黎，或状态大于30的供应商（大致上说，也就是那些关系变量谓词）。如图中所示，这个设计导致了某些冗余；更具体地说，在每一个关系变量中，供应商S3出现了一次。

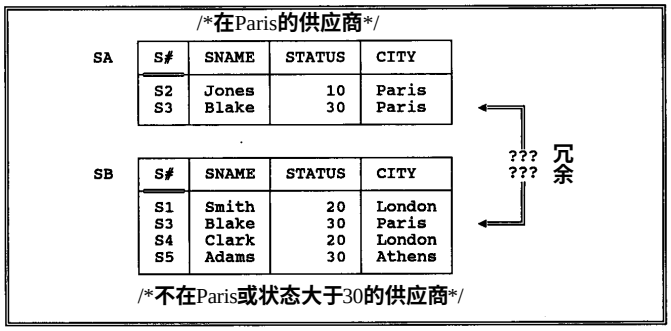

图12-7 供应商：不好但可能存在的一种设计

注意：顺便提及这个元组会在两个关系变量中都出现。相反，假设这个元组要在 SB中出现，而不在SA中出现。在SA中应用封闭世界的假设， SA就会告诉我们供应商S3并不在巴黎，然而， SB告诉我们供应商S3在巴黎。换一句话说，这将产生一个冲突，并且数据库会变得不一致。

当然，在图12-7中的设计问题是很明显的，正是由于允许同样的元组在两个截然不同的关系变量中出现的这个问题。换句话说，这两个关系变量有互相重复的意义，即可能出现对同一个元组满足两个关系变量谓词。因此，有如下的显而易见的规则：

- 正交设计原理（最初版本）：对于一个给定的数据库，任何两个基本关系变量不应该有重叠的含义。

要点如下：

1) 在第9章中，从使用者的角度看，所有的关系变量都是基本关系变量（除了只是作为简便方式才定义的关系变量之外）。换句话说，这个原理适用于所有“可表达”的数据库的设计，并不只限于“真实”的数据库——这就是所谓的数据库相对性原则（当然，类似的观点也适合于规范化理论）。
2) 注意，两个关系变量可能并没有互相重叠的含义，除非它们是同一种类型（也就是说，除非它们有相同的标题）。
3) 正交设计准则暗示着（举例来说），当插入一个元组时，可以认为这个操作是将一个元组插入一个数据库而不是插入一个具体的关系变量——因为最多有一个关系变量是满足新元组的插入谓词的。

现在，当插入一个元组时通常要指定元组插入的关系变量 *R*的名字。但是，这一事实并不与前面的观点矛盾。事实上，名字 *R*只是相应的谓词，比如 *PR*的简写；真实的意思是说“插入元组*t*——并且*t*满足谓词*PR*”。更进一步讲，*R*有可能是一个视图，可能是由 *A*与*B*的并集表达式来定义的视图——并且，正如在第9章所见的，系统如果能知道新的元组是插入 *A*、*B*还是都插入，那么将是最好不过的。

事实上，与上面的论述类似的观点也适用于其他所有的操作；在所有的情况下，关系变量名字实际上都只是关系变量谓词的简写。当表示数据语义这样的观点时，不应该太强调这是谓词，而不仅是名字。

到现在为止还没有讨论完正交设计准则——还有一个很重要的细化工作要作。考虑图12-8，它显示了有关供应商的另一个不好的但却可能存在的设计。其中存在两个关系变量，它们本

身并不包含重叠含义，但是它们在{S#，SNAME}上的投影确实出现了重叠（事实上，这两个投影的含义是类似的）。因此，如果想插入一个元组（S6,Lopez）到一个由这两个投影的并集定义的视图中就会引起元组（S6,Lopez,*t*）被插入到SX中、元组（S6,Lopez,*c*）被插入到SY中（其中*t*和*c*都是可用的缺省值）。很明显，正交设计准则需要扩展以便将图12-8中的问题考虑进去：

SX

S#	SNAME	STATUS
S1	Smith	20
S2	Jones	10
S3	Blake	30
S4	Clark	20
S5	Adams	30

SY

S#	SNAME	CITY
S1	Smith	London
S2	Jones	Paris
S3	Blake	Paris
S4	Clark	London
S5	Adams	Athens

图12-8 有关供应商的另一个不好的但却可能存在的设计

- 正交设计准则（最终版本）：*A*和*B*是数据库中任意的两个基关系变量。那么并不存在无损分解将*A*和*B*分为*A1*，...，*Am*和*B1*，...，*Bn*（相互独立）使在*A1*，...，*Am*中的某些投影*Ai*和在*B1*，...，*Bn*中的某些投影*Bj*有重叠的含义。

要点如下：

1) 术语“无损分解”在这里是其通常的含义——即分解为一组投影：
   - 给定的关系变量可以通过将投影连接回去的方法重新得到；
   - 在重新连接的过程中这些投影中没有冗余的项。
2) 这个版本蕴涵了最初的版本，因为关系变量*R*就是其本身的一个投影，因此无损分解是肯定存在的。

3. 评论

1) 假定以一个通常的供应商关系变量S开始，但是为了设计的原因将这个关系变量分解为一组投影。那么正交设计准则会告诉我们这些投影应该都是互不相交的，在这种意义上讲，没有一个供应商的元组会出现在多于一个投影中。这种分解称为正交分解。注意：术语正交，就是取自这一设计准则的实际含义，即基本关系变量是互相独立的（没有重叠含义）。当然，这个准则是常识性的，但它是形式化意义上的常识（就如同规范化理论一样）。
2) 正交设计的总体目标是减少冗余，并因此避免更新异常（也与规范化类似）。事实上，它补充了规范化理论，不严格地说，规范化在关系变量中减少了冗余，而正交性在关系变量之间减少了冗余。
3) 正交性可能是一个常识，但是它在实践中经常不受重视（事实上，这种不重视的看法甚至有时是受推崇的）。像下面的这个设计是很常见的，它取自金融数据库：

```
ACTIVITIES_1997 { ENTRY#, DESCRIPTION, AMOUNT, NEW_BAL }
ACTIVITIES_1998 { ENTRY#, DESCRIPTION, AMOUNT, NEW_BAL }
ACTIVITIES_1999 { ENTRY#, DESCRIPTION, AMOUNT, NEW_BAL }
ACTIVITIES_2000 { ENTRY#, DESCRIPTION, AMOUNT, NEW_BAL }
ACTIVITIES_2001 { ENTRY#, DESCRIPTION, AMOUNT, NEW_BAL }
```

事实上，将编码编入名字——关系变量或其它的名字——违反了信息准则（The Information Principle），这个准则提出所有的数据库信息必须显式地根据值的方法而不能用其它方法来表示。

4) 如果*A*和*B*是同一类型的基本关系变量，根据正交设计准则蕴涵着：

```
A UNION B       总是一个不相交的并集
A INTERSECT B   总是空的
A MINUS B       总是等于A
```

12.7 其他的规范化形式

再回到对规范化的讨论。在第 11章的引言中提到过，除了这几章讨论的范式，确实存在其他的规范化形式。实际情况是，规范化理论和有关问题 —— 现在通常被认为是依赖理论 —— 已经发展为一个拥有大量文献著作的相当可观的领域。这个领域的研究在不断继续，不断深入。有关这方面的更深入的的讨论已超出本章的范围；这个领域的研究（ 20世纪80年代中期的研究）的综述参见[12.17]。在这里只提两个具体的问题。

1) 域码（Domain-key）范式：域码范式（DK/NF）在[12.15]中由Fagin提出。DK/NF —— 与所讨论过的规范化形式不同 —— 根本不是根据函数依赖、多值依赖或连接依赖而定义的。相反，当且仅当一个关系变量 *R*中的每一个约束都是 *R*中所应用的域约束和码约束的合理结果时，称*R*是DK/NF。

   • 域约束 —— 在这里用到的一个术语 —— 是指有关指定属性值从一些规定的域中取值的约束（在第8章的术语中，这样的约束是一个属性约束而不是一个域约束）。

   • 码约束是有关某一个特定的属性或属性组构成的候选码的约束。

   满足一个DK/NF关系变量上的约束因此在概念上显得很简单，既然足以满足“域”和码约束，那么所有的其它约束就会自动地被满足。还要注意在这里“所有其它的约束”意味着不止是函数依赖、多值依赖和连接依赖 —— 事实上，它代表整个关系变量谓词。Fagin在文献[12.15]中提出，任何DK/NF关系变量都必须是 5NF（并且因此也是 4NF，等等）。事实上在（3,3）范式中也是如此（参见下面）。然而，DK/NF并不是经常可以达到的，而“何时能够达到”的问题也没有解决。

2) “选择-并”（Restriction-union）范式：再次考虑供应商关系变量 S。规范化理论告诉我们，关系变量S是一种“好”的范式；事实上，它是 5NF，并且可以通过投影的方式消除异常。但是为什么将所有的供应商放在一个单独的关系变量中呢？如果设计将伦敦的供应商放入一个关系变量中（如 LS），而将巴黎的供应商放入另一个（如 PS），等等，这样的关系变量可不可以呢？换句话说，可不可以将原来的供应商的关系变量按选择来分解而不是通过投影来分解呢？这样的设计结果将是一个好的设计还是不好的呢？（事实上它几乎肯定是不好的 —— 参见第7章中练习7.8 —— 但是经典的规范化标准化理论对此类问题没有触及）。

   因此，对于规范化研究的另一个方向就是研究通过非投影方式对关系变量进行分解的问题。在这个例子中，已经提到分解运算符是选择；而相应的合成运算符是并。因此，构建一个类似投影-连接规范化理论的“选择-并”规范化理论是可行的 ㊀。就本书作者的了解，还没有任何这样的理论被详细地提出来，但是一些最初的观点可以在Smith(参见[12.31])的论文中找到，在其论文中定义了一个称为“（3,3）范式”的新型

㊀ 事实上，Fagin最早称5NF为投影-连接范式，就是因为这一范式是基于投影的连接操作的。

的范式。（3,3）范式隐含了BCNF范式；然而，一个（3,3）范式关系变量不必是4NF，一个4NF也不必是（3,3）范式，所以（如上面表明的）推演到（3,3）范式是和推演到4NF（和5NF）相互正交的。对此观点的深层研究出现在文献[12.14]和[12.22]中。

12.8 小结

本章完成了关于进一步规范化的理论的讨论（从第11章开始）。本章讨论了多值依赖，多值依赖是一种一般化的函数依赖；还讨论了连接依赖（JD），它是多值依赖的一般化形式。大致地说：

- 一个关系变量$R\{A,B,C\}$满足多值依赖$A\twoheadrightarrow B|C$，当且仅当，一组B值与一给定的$\{A,C\}$对匹配与否，只决定于A的值；对C与$\{B,C\}$的情况也是如此。这样的一个关系变量可以被无损分解为它在$\{A,B\}$和$\{A,C\}$上的投影；事实上，多值依赖是这种分解的充要条件（Fagin定理）。
- 一个关系变量$R\{A,B,...,Z\}$满足连接依赖 ⋈ $\{A,B,...,Z\}$，当且仅当它与它在$A,B,...,Z$上的投影的连接的结果相等。这样的一个关系变量可以（显然）被无损分解为这些投影。

一个关系变量是4NF，仅当它满足的多值依赖是非超码的函数依赖。一个关系变量是5NF——也称为投影-连接范式，即PJ/NF——仅当它满足的连接依赖是非超码的函数依赖（意思是如果连接依赖是 ⋈ $\{A,B,...,Z\}$，那么$A,B,...,Z$中每一个都是超码）。5NF（总可以达到的）是关于投影和连接运算的最终范式。

我们还讨论了规范化过程，给出了不很正规的步骤和一些相关的解释。然后描述了正交设计准则：即，大致上说，不应该有两个关系变量有互相重叠的投影。最后，简单地提到了一些其它的范式。

综上所述，应该指出，对上面所讨论的这些问题的研究是很有意义的。原因是“进一步规范化”，或习惯地称为依赖理论的研究，确实描绘出了数据库设计这一领域的一些科学规律。因为数据库设计以往太过工艺化（即太过主观，缺少坚实的理论和方针）。因此，应该欢迎在依赖理论的研究上取得进一步的成功。

练习

12.1 本章讨论的关系变量CTX和SPJ——参见图12-2和图12-4——各自满足一个特定的MVD和一个特定的JD，它们并没有被各自的关系变量中的候选码所蕴涵。用第8章中的语法来表示这个MVD和JD。

12.2 假定C是一个俱乐部，关系变量$R\{A,B\}$满足，当且仅当a和b都是C的一员时，元组(a,b)才出现在R中。那么R满足什么样的FD、MVD和JD？R属于第几范式？

12.3 有一个包含推销员、推销地和产品的数据库。每一个推销员有责任在一个或多个地区推销；每一个地区都有一个或多个推销员。同样，每一个推销员有责任推销一个或多个产品，每一个产品可以有一个或多个推销员。每一个产品都在每一个地区售卖；然而，没有两个推销员在同一地区销售同一产品。每一个推销员在每一个地区卖它有责任卖的同一组产品。对这些数据设计一组合适的关系变量。

12.4 在第11章的练习11.3的答案中给出了一个将任意的关系变量R无损分解为一组BCNF关系变量的算法。修改这个算法，使其能产生4NF的关系变量。

12.5 （对练习12.3的修改）有一个包含推销员、推销地和产品的数据库。每一个推销员有责任在一个或多个地区推销；每一个地区都有一个或多个推销员。同样，每一个推销员有责任推销一个或多个产品，每一个产品可以有一个或多个推销员。最后，每一个产品都在每一个或多个地区售卖。而且，如果推销员 *R*对*A*地区负有责任，产品*P*在*A*地区售卖，并且推销员*R*有责任卖产品*P*，那么*R*在*A*地区卖*P*产品。对这些数据设计一组合适的关系变量。

参考文献和简介

12.1 A. V. Aho，C.Beeri, and J. D. Ullman: “ The Theory of Joins in Relational Databases” ，*ACM TODS 4* , No.3 (September 1979). First published in Proc. 19th IEEE Symp. On Foundations of Computer Science（October 1997)。

这篇论文指出，关系变量在与任何两个它的投影的连接不相同的情况下也可以存在，但要与它的三个或多个投影的连接相等。这篇论文的主要目的是提供一种算法，现在一般称chase，用来决定一个给定的连接依赖是否是一组给定的函数依赖的合理结果（参见[12.17]中的一个相关的例子)。这个问题是与如下的问题等价的，即在给定的一组函数依赖下，判定一给定分解是否是无损的。这篇论文还讨论了将这个算法扩展，来处理给定的依赖是多值依赖而不是函数依赖的问题。

12.2 Catriel Beeri，Ronald Fagin, and John H.Howard: “ A Complete Axiomatization for Functional and Multi-Valued Dependencies,” Proc.1977 ACM SIGMOD Int. Conf. on Management of Data, Toronto, Canada (August 1977)

Armstrong[10.1]的扩展是在其中包括了多值依赖和函数依赖。特别地，它给出了下面一组正确而完整的多值依赖的参考规则：

1) 互补律 （Complementation)：如果*A*、*B*和*C*中包含了关系变量的所有属性，并且 *A*是*B*∩*C*的父集，那么*A*→→*B*，当且仅当*A*→→*C*。
2) 自反律 （Reflexivity)：如果*B*是*A*的子集，则*A*→→*B*。
3) 增广律 （Augmentation)：如果*A*→→*B*，并且*C*是*D*的一个子集，那么*AD*→→*BC*。
4) 传递律：如果*A*→→*B*，并且*B*→→*C*，那么，*A*→→*C*-*B*。

下面的规则是从上面的规则中推导出来的：

5) 伪传递性(Pseudotransitivity)：如果*A*→→*B*，并且*BC*→→*D*，那么*AC*→→*D*-*BC*。
6) 并(Union)：如果*A*→→*B*，并且*A*→→*C*，那么*A*→→*BC*。
7) 分解(Decomposition)：如果*A*→→*BC*，那么*A*→→*B*∩*C*、*A*→→*B*-*C*，并且*A*→→*C*-*B*。
8) 复制(Replication)：如果*A*→*B*，那么*A*→→*B*。
9) 合并 （Coalescence)：如果*A*→→*B*，并且 *C*→*D*，并且 *D*是*B*的一个子集， *B*∩*C*为空，那么*A*→*D*。

Armstrong的规则（参见第10章）再加上上面的规则1~4和8~9就形成了一个正确而又完整的函数依赖和多值依赖的推理规则。

这篇论文还推导了许多有用的相关函数依赖和多值依赖规则。

12.3 Volker Brosda and Gottfried Vossen：“ Update and Retrieval Through a Universal Schema Interface, ” *ACM TODS 13*, No.4(December 1988)

早期提出“泛关系”接口的尝试（参见 [12.19]），只处理了检索操作符。这篇论文还提出处理更新操作的方法。

12.4 C. Robert Carlson and Robert S. Kaplan：“A Generalized Access Path Model and Its Application to a Relational Data Base System,” Proc. 1976 ACM SIGMOD Int. Conf. on Management of Data, Washington DC(June 1976).

参见[12.19]的注释。

12.5 C. J. Date：“Will the Real Fourth Normal Form Please Stand Up?”. in C.J.Date and Hugh Darwen, *Relational Database Writings 1989-1991*. Reading, Mass.: Addison-Wesley (1992).

引自摘要：“在数据库设计中，存在好几种对第四范式（4NF）截然不同的解释。本文的目的就是试图澄清这些问题”。也许应该附加说明在本章中称为 4NF的概念是仅指真正的4NF……不允许任何替代品！

12.6 C. J. Date :“ The Normal Is So...Interesting” (in tow parts). *DBP&D 10* , Nos. 11-12(November-December 1997).

在第12.5节中讨论的逆规范化问题取自这篇论文。下面是另外几个观点：

- 甚至在只读数据库中，标明一个完整性约束也是必要的，因为它们定义了数据的含义，并且（正如在第 12.4节中所注明的）它们的不可逆规范性又提供了一个标明特定重要约束的简单方法。如果数据库不是只读的，那么不可逆规范性同样提供了一种满足这些约束的简单的方法。
- 逆规范化蕴含了冗余的增加 —— 但是（与普遍的观点相反），增加冗余并不一定蕴含逆规范化！许多作者都陷入了这个圈套，有一些作者还在重蹈覆辙。
- 作为一个普遍的规则，逆规范化（即在逻辑层的逆规范化）应该作为“仅当所有其它的方法失败”时的一种性能策略（performance tactic）来使用[4.16]。

12.7 C. J. Date: “The Final Normal Form!”(in two parts), DBP&D11, Nos. 1-2(January-February 1998).

一篇关于连接依赖和5NF的文章。

12.8 C. J. Date: “What’s Normal,Anyway? ”, *DBP&D11*, No.3 (March 1998).

对某些“病态”的规范化例子的综述，如在第 11章练习11.2中的USA例子。

12.9 C. J. Date: “Normalization Is No Panacea,” *DBP&D11*, No.4 (April 1998).

有关规范化理论不能解决的一些数据库设计问题的综述。这篇论文并不是一篇攻击性的文章。

12.10 C. J. Date and Ronald Fagin ：“Simple Conditions for Guaranteeing Higher Normal Forms in Relational Databases,” in C.J.Date and Hugh Darwen, *Relational Database Writing 1989-1991*. Reading, Mass: Addison-Wesley(1992). Also pubished in *ACM TODS 17* , No.3 (September 1992).

如果（a）一个关系变量*R*是3NF；（b）*R*的所有的候选码都是简单的，那么 *R*自动地就是5NF。换句话说，在这个关系变量中没有必要担心在本章中讨论的相对复杂的问题 —— 多值依赖、连接依赖、 4NF和5NF。注意：这篇论文还证明了另一个结论，即如果(a)*R*是属于BCNF范式，并且（b）它至少有一个候选码是简单的，那么 *R*自动属于4NF，但并不一定是5NF。

12.11 C. J. Date and David McGovera：“A New Database Design Principle,”in C. J. Date, *Relational Database Writings 1991-1994*. Reading, Mass: Addison-Wesley(1995)

12.12 C. Delobel and D. S. Parker：“Functional and Multi-Valued Dependencies in a Relational Database and the Theory of Boolean Switching Functions,” Tech. Report No. 142, Dept. Maths. Appl. et Informatique, Univ. de Grenoble, France (Novermber 1978).

将[10.3]的结果扩展到多值依赖和函数依赖。

12.13 Ronald Fagin:“Multi-Valued Dependencies and a New Normal Form for Relational Databases,” *ACM TODS 2*, No.3 (November 1977).

这个新的规范化形式就是4NF。

在这里增加一个嵌入的多值依赖的说明。假定将12.2节的关系变量CTX扩展到包含附加的属性DAYS，这个属性代表在指定的课程上一个指定的教师在一本教材上所花的天数。将这个关系变量称为CTXD，下面是一个样本值：

CTXD

COURSE	TEACHER	TEXT	DAYS
Physics	Prof. Green	Basic Mechanics	5
Physics	Prof. Green	Principles of Optics	5
Physics	Prof. Brown	Basic Mechanics	6
Physics	Prof. Brown	Principles of Optics	4
Math	Prof. Green	Basic Mechanics	3
Math	Prof. Green	Vector Analysis	3
Math	Prof. Green	Trigonometry	4

{COURSE,TEACHER,TEXT}的组合是一个候选码,并且存在函数依赖：

{COURSE，TEACHER，TEXT} → DAYS

可以观察到这个关系变量是属于第四范式的；它并不包括任何不属于函数依赖的多值依赖。然而，它确实包括两个嵌入的多值依赖（TEACHER对于COURSE和TEXT对于COURSE）。当“常规”的多值依赖$A \rightarrow\rightarrow B$在$R$的一些投影中存在时，在关系变量$R$中存在嵌入的$B$对$A$的多值依赖。一个常规的多值依赖是一个嵌入多值依赖的特殊例子，但是并非所有的嵌入的多值依赖都是常规的多值依赖。

正如这个例子所示，嵌入的多值依赖与常规的多值依赖一样蕴含着冗余；然而，这个冗余（通常）不能通过投影消除。上面所示的关系变量根本不能被无损分解为它的投影（事实上，它既是第五范式也是第四范式），因为DAYS依赖于所有的三个属性COURSE、TEACHER和TEXT，并且，如果不是其它三个都出现，它也不能在关系变量中出现。因此这两个嵌入的多值依赖可以标明为关系变量的附加的显式的约束。详细的情况作为练习。

12.14 Ronald Fagin：“Normal Forms and Relational Database Operators.” Proc. 1979 ACM SIGMOD Int. Conf. on Management of Data, Boston, Mass.(May/June 1979).

这篇论文介绍了投影-连接范式的概念（PJ/NF，或5NF），而且，其中还不止这些。它可以看作是“经典的”规范化理论的定义性说明——即基于将投影作为分解的操作符而将自然连接作为相应的合并的操作符的无损分解理论。

12.15 Ronald Fagin：“A Normal Form for Relational Databases that Is Based on Domains and Keys.” *ACM TODS 6*, No.3 (September 1981).

12.16 Ronald Fagin：“Acyclic Databse Schemes(of Various Degrees): A Painless Introduction,” IBM Research Report RJ 3800(April 1983). Republished in G. Ausiello

and M.Protasi (eds.), *Proc. CAAP83 8th Colloquium on Trees in Algebra and Programming* (Springer Verlag Lecture Notes in *Computer Science* 159). New York N. Y.: Springer Verlag (1983).

本章的第12.3节中介绍了满足一个特定的有环约束（cyclic constraint）的特定的三重关系变量SPJ如何被无损分解为它的三个二元投影。所得出的结果数据库的结构被称为是有环的，因为这三个关系变量中的每一个都和其他两个有一个相同的属性（如果结果被描述为一个超图，在这个超图中边代表单独的关系变量，相应的两个边相交汇的节点是这两个边的共同的属性，那么应用术语“环”的原因就是很清楚了）。与之相反，实践中的大多数的结构都是无环的。无环(acyclic)结构有很多有环结构所不具有的属性。在这篇论文中，Fagin提供并解释了一系列这样的属性。

无环性的一种很有用的方法如下：正如规范化理论可以帮助决定何时一个单独的关系变量可以用某种方法重构造一样，无环的理论可以帮助决定何时一组关系变量可以用某种方法来重构造。

12.17 R.Fagin and M. Y. Vardi: “The Theory of Data Dependencies – A Survey.” IBM Research Report RJ4321(June 1984).Republished in *Mathematics of Information Processing: Proc. Symposiain Applied Mathematics 34*, American Mathematical Society (1986).

它介绍了依赖理论在20世纪80年代中期的简单的发展过程（注意：“依赖”在这里并不只代表函数依赖）。特别是，这篇论文总结了在这一领域的三个特定方面的主要成就，并且提供了一系列经过仔细选择的相关参考资料。这三个方面是（ 1）蕴涵（implication）问题；（2）“泛关系”模型；（3）无环模式。蕴涵问题决定的是，给定一组依赖*D*和一些特定的依赖*d*，*d*是否是*D*的一个逻辑结果（参见10.7节）。泛关系模型和无环模式在文献[12.19]和[12.16]的介绍中分别简单地进行了讨论。

12.18 Ronald Fagin, Alberto O. Mendelzon, and Jeffrey D. Ullman: “A Simplified Universal Relation Assumption and Its Properties,” *ACM TODS* 7, No.3 (September 1982).

论文中假设现实世界经常可以通过“泛关系”——或泛关系变量来表示[12.19]-满足一个连接依赖和一组函数依赖，并且探索了这种假设的一些结果。

12.19 W. Kent: “Consequences of Assuming a Universal Relation.” *ACM TODS 6*, No 4 (December 1981).

泛关系的概念用几种不同的方法证明了它自身。首先，在前面两章中描述的规范化理论指出了，定义一初始的泛关系——或者更正确地说，是一个泛关系变量——是可能的，这个关系中包括了与数据库有关的所有属性，并且还描述了关系变量如何被“小”（较低级）的投影连续代替，直到达到一些“好的”结构。但是这种初始的假定现实有效吗？文献[12.19]中表明，无论是在理论上还是实际上都并非如此。文献[12.32]是对文献[12.19]的一个回答，而文献[12.20]是对这个回答的回答。

第二个也是更重要的泛关系变量概念是作为一个用户界面表示的。它的基本观点十分易懂，并且事实上（从直观的角度）也是十分受欢迎的。用户可以只根据属性，而不是根据关系变量和这些关系变量中的连接设计它们对数据库的要求。例如：

```
STATUS  WHERE  COLOR = COLOR('RED')
```

(“取出提供红色零件的供应商的信息”)。在这一点上会引来两种不同的解释：

1) 一种可能情况是为了回答这个查询，系统应该自己决定选择哪一个逻辑存取路径（特别是选择哪些连接）。这个方法在[12.4]中有讨论（文献[12.4]是第一个讨论“泛关系”接口的可能性的文章，但它并没有用这个术语）。这种方法严格地依赖于属性的适当命名。例如，两个关于供应商数量的属性（分别在关系变量S和SP中）必须给定同一个名字；相反，供应商的城市和零件的城市的属性（分别在关系变量S和P中）一定不能是同样的名称。如果违反了这两个规则中的任一个，系统就会无法正确处理某些查询。

2) 另一种可能的方法是简单地将所有的的查询看成是根据事先定义的一组连接——由数据库中所有的关系变量的相应连接形成的事先定义的视图——形成的。

毫无疑问，无论哪一种方法都简化了实际中许多查询的表达——而且这样的方法对前端自然语言查询的支持是十分重要的。很明显，该方法一般情况下必须要求系统对存取路径有自动的指定功能。考虑查询：

```
STATUS  WHERE  COLOR≠COLOR('RED')
```

这个查询的意思是“取出提供非红色的零件的供应商的状态”还是“取出不提供一个红零件的供应商”？不论是哪个，都必须有表示另外的一个查询的方法（关于这个问题，上面的第一个例子也是可以产生歧义的：“给出只提供红色零件的供应商的状态”）。这里还有第三个例子：“取出在同一个城市的一对供应商”。显然一个明显的连接是必要的（因为大致说来，这个问题包括一个关系变量S和它本身的连接）。

12.20 William Kent: “The Universal Relation Revisited,” *ACM TODS 8*. NO.4(December 1983).

12.21 Henry F. Korth *et al.*: “System/U: A Database System Based on the Universal Relation Assumption,” *ACM TODS 9*, No. 3 (September 1984).

这篇文章描述了理论、DDL、DML和一个在斯坦福大学开发的实验性的“泛关系”系统的实现。

12.22 David Maler and Jeffrey D. Ullman: “Fragments of Relations,” Proc. 1983 SIGMOD Int. Conf. on Management of Data, San Jose, Calif. (May 1983).

12.23 David Maler, Jeffrey D. Ullman, and Moshe Y. Vardi: “On the Foundations of the Universal Relation Model,” *ACM TODS 9*, No. 2 (June 1984). An earlier version of this paper, under the title “The Revenge of the JD,” appeared in Proc. 2nd ACM SIGACT-SIGMOD Symposium on Principles of Database Sy stems, Atlanta, Ga. (March 1983).

12.24 David Maier and Jeffrey D. Ullman: “Maximal Objects and the Semantics of Universal Relation Databases,” *ACM TODS 8*, No. I (March 1983).

最大对象表示了一个在“泛关系”系统中当根本的结构不是无环的情况下出现的歧义问题的解决方法（参见[12.16]）。一个最大对象对应一个无环结构的全属性的一个子集。这样的对象用来指导查询的翻译，否则查询的含义就会出现歧义。

12.25 J.-M. Nicolas: “Mutual Dependencies and Some Results on Undecomposable Relations,” Proc. 4th Int. Conf. on Very Large Data Bases, Berlin, Federal German Republic (September 1978).

介绍了“相互依赖”（mutual dependency）的概念。一个相互依赖实际上是普通连接依赖的一个特例——即它是一个连接依赖，但并不是一个多值依赖或函数依赖——它碰

巧包含三个投影（与在 12.3节中给出的连接依赖的例子相似）。这与第11章中讨论的共有依赖的概念毫无关系。

12.26 Sylvia L. Osborn: “ Towards a Universal Relation Interface,” Proc. 5th Int. Conf. on Very Large Data Bases, Rio de Janeiro, Brazil (October 1979).

这篇论文提出了一个假定，如果对一个给定的查询可以提供候选答案的“ 泛关系 ”系统中有两个或多个连接的结果，那么好的回答是所有的这些候选的答案的并集。并给出了产生这样的连接结果的算法。

12.27 D. Stott Parker and Claude Delobel: “ Algorithmic Applications for a New Result on Multi-Valued Dependencies,” Proc. 5th Int. Conf. on Very Large Data Bases, Rio de Janeiro, Brazil (October 1979).

将[12.12]的结果应用于不同的问题中。例如测试一个无损分解的过程。

12.28 Y. Sagiv, C. Delobel, D. S. Parker, and R. Fagin: “ An Equivalence between Relational Database Dependencies and a Subclass of Propositional Logic,” *JACM 28*, No. 3 (June 198 1).

组合了[10.8]和[12.9]内容。

12.29 Y. Sagiv and R. Fagin: “ An Equivalence between Relational Database Dependencies and a Subclass of Propositional Logic,” IBM Research Report RJ2500 (March 1979).

将[10.8]的结果扩展到既包括多值依赖也包括函数依赖。

12.30 E. Sclore: “A Complete Axiomatization of Full Join Dependencies,” *JACM 29*, No. 2 (April 1982).

将[12.2]的工作扩展到既包括连接依赖也包括多值依赖和函数依赖。

12.31 J. M. Smith: “ A Normal Form for Abstract Syntax,” Proc. 4th Int. Conf. on Very Large Data Bases, Berlin, Federal German Republic (September 1978).

这篇论文介绍了（3,3）范式。

12.32 Jeffrey D. Ullman: “ On Kent’s ‘Consequences of Assuming a Universal Relation,”’ *ACM TODS 8*, No. 4 (December 1983).

12.33 Jeffrey D. Ullman: “ The U.R. Strikes Back,” Proc. 1st ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, Los Angeles, Calif. (March 1982).

部分练习答案

12.1 首先是关系变量CTX的多值依赖：

```
CONSTRAINT CTX_MVD
   WITH
     ( CTX RENAME COURSE AS C, TEACHER AS T, TEXT AS X )
                                          AS T1,
     ( EXTEND T1
       ADD ( CTX WHERE COURSE = C ) { T } AS A )
                                          AS T2,
     ( EXTEND T2
       ADD ( CTX WHERE COURSE = C AND TEXT = X ) { T } AS B )
                                          AS T3,
     ( T3 WHERE A ≠ B ) AS T4 :
     IS_EMPTY ( T4 ) ;
```

当然，还存在一个更简单的公式：

```
CONSTRAINT CTX_MVD CTX = CTX { COURSE, TEACHER } JOIN
                         CTX { COURSE, TEXT } ;
```

关系变量SPJ的连接依赖为：

```
CONSTRAINT SPJ_JD SPJ = SPJ { S#, P# } JOIN
                        SPJ { P#, J# } JOIN
                        SPJ { J#, S# } ;
```

12.2 注意，对于每一个可能的 B 值，R 中都有一个可能的 A 值与其对应，并且所有的 A 值的集合（称为 S）与所有的 B 值的集合是相同的。R 的主体因此是 S 集与其自身的笛卡尔积的结果；同样，R 与它的投影 $R\{A\}$ 和 $R\{B\}$ 的笛卡尔积也是相同的。因此 R 满足下面的多值依赖（注意，这个多值依赖是非平凡的，因此它们当然并不被所有的二元关系变量满足）

$$\{\ \} \twoheadrightarrow A \mid B$$

同样，R 也满足连接依赖 $\bowtie\{A, B\}$（当没有相同属性时，连接依赖退化为笛卡尔积）。这说明 R 不是4NF，它可以在 A 和 B 上无损分解为它的投影（当然，这些投影有可辨别的主体），R 是一个BCNF范式（是全码），它满足不是非平凡的函数依赖。

注意：R 也满足多值依赖：

$$A \twoheadrightarrow B \mid \{\ \}$$

和

$$B \twoheadrightarrow A \mid \{\ \}$$

然而，这些多值依赖是平凡的，因为包含属性 A 和 B 的每一个二元关系变量都满足这个依赖。

12.3 首先通过明显的解释来介绍三个关系变量：

```
REP      { REP#, ... }
         KEY { REP# }

AREA     { AREA#, ... }
         KEY { AREA# }

PRODUCT  { PROD#, ... }
         KEY { PROD# }
```

第二步，通过一个关系变量

```
RA { REP#, AREA# }
   KEY { REP#, AREA# }
```

来表示推销员和推销地区的联系。通过一个关系变量

```
RP { REP#, PROD# }
   KEY { REP#, PROD# }
```

来表示推销员和产品的联系（这两个联系都是多对多的）。

由于每一个产品在每一个地区都有，因此，如果引入一个关系变量

```
AP { AREA#, PROD# }
   KEY { AREA#, PROD# }
```

来表示地区和产品之间的联系，那么可以有约束如下（称之为 C）：

```
AP = AREA { AREA# } TIMES PRODUCT { PROD# }
```

注意到约束C暗示关系变量AP不是第四范式（参见练习 12.2）。事实上，关系变量AP并不能提供任何从其它的关系变量中得不到的信息；精确地说，有

```
AP { AREA# } = AREA { AREA# }
```

和

```
AP { PROD# } = PRODUCT { PROD# }
```

然而，还是暂时假定关系变量AP是包括在设计中的。

没有在同一地区卖同一种产品的两个推销员。换句话说，给定一个{AREA#, PROD#}的组合，确实有一个相应的推销员（REP#）存在，所以引入一个关系变量

```
APR { AREA#, PROD#, REP# }
    KEY { AREA#, PROD# }
```

在这个关系变量中（使函数依赖更明显）

```
{ AREA#, PROD# } → REP#
```

（当然，作为候选码的组合{ AREA#, PROD#}的特定值是足以表示这个函数依赖的）。然而，现在关系变量RA、RP和AP都是冗余的，因为它们都是APR的投影；因此它们都可以被删掉。然后用约束C1来代替约束C：

```
APR { AREA#, PROD# } = AREA { AREA# } TIMES PRODUCT { PROD# }
```

这个约束必须单独并且显式地标注。

既然每一个推销员在所有的推销地区销售所有的产品，因此对关系变量APR还有附加的约束C2：

```
REP#→→AREA# | PROD#
```

（一个非平凡的多值依赖；关系变量APR不是4NF）。这个约束也必须被单独和显式地标注。因此最后的设计中包括关系变量REP、AREA、PRODUCT和APR，并且包括显式的约束C1和C2。

这个练习非常清楚地表示了一个观点，即在通常情况下，规范化规则是足够表示一些语义方面的特定问题的（基本上说，就是由候选码所蕴含的依赖，这里的依赖指函数依赖、多值依赖和连接依赖），但是对于一些其它方面的问题，也需要给出附加依赖的明确说明，而且有一些方面无法用这样的依赖来表示。它还表明了一种观点，即有时并不需要“一直”规范到底（关系变量APR是BCNF，但并不是4NF）。

12.4 这个修订是很容易理解的——所有必要的操作都是将对函数依赖和BCNF的参照替换为类似的对多值依赖和4NF的参照，因此：

0. 初始化D使其只包括R。
1. 对于D中每一个不是4NF的关系变量T，执行第2步和第3步。
2. 使$X\rightarrow\rightarrow Y$称为$T$中的一个多值依赖，这个多值依赖不满足4NF的要求。
3. 将D中的T用其两个投影代替，即通过X和Y以及通过除去Y中的属性的所有属性来代替。

12.5 这是一个“有环约束”的例子，下面的设计是适当的：

```
REP      { REP#, ... }
         KEY { REP# }

AREA     { AREA#, ... }
         KEY { AREA# }

PRODUCT  { PROD#, ... }
         KEY { PROD# }

RA { REP#, AREA# }
   KEY { REP#, AREA# }

AP { AREA#, PROD# }
   KEY { AREA#, PROD# }
```

```
PR { PROD#, REP# }
   KEY { PROD#, REP# }
```

用户还需要知道RA、AP和PR的连接并不包括任何“连接陷阱”：

```
( RA JOIN AP JOIN PR ) { REP#, AREA# }  = RA AND
( RA JOIN AP JOIN PR ) { AREA#, PROD# } = AP AND
( RA JOIN AP JOIN PR ) { PROD#, REP# }  = PR
```