第5章 关系数据库模式的规范化设计

在一个关系数据库应用系统中，构成该系统的关系数据库的全局逻辑结构（逻辑模式）的基本表的全体，称为该数据库应用系统的关系数据库模式。

关系数据库模式设计，就是按照不同的范式（标准）对关系数据库模式中的每个关系模式进行分解，用一组等价的关系子模式来代替原有的某个关系模式，即通过将一个关系模式不断地分解成多个关系子模式和建立模式之间的关联，来消除数据依赖中不合理的部分，最终实现使一个关系仅描述一个实体或者实体间的一种联系的目的。关系数据库模式设计是数据库应用系统设计中的核心问题之一。

本章首先论述关系约束与关系模式的表示，然后讨论为什么要对关系模式进行规范化设计的问题，在此基础上引出函数依赖的概念和函数依赖的公理体系，并给出最小函数依赖集求解算法；接着引出关系模式的范式概念和关系模式的分解概念，并给出保持无损性和保持依赖性的分解算法，最后总结并给出关系模式的规范化概念和步骤。

5.1 关系约束与关系模式的表示

假设在大学教学信息管理数据库应用系统的设计中，通过直接整理用户组织日常所用的相关教学管理表格，得到大学学生学籍关系表（表3.2）、课程归属表、学习成绩表、授课统计表和教师信息表。这几个关系表合并在一起，构成的大学教学信息管理数据库应用系统中除学生学籍关系表以外的其他关系表结构如表5.1所示。

表5.1 大学教学信息管理数据库应用系统中部分表的属性

表　　名	课程归属表	学习成绩表	授课统计表	教师信息表
拥有的属性	课程号	学号	课程号	教师工号
	课程名	姓名	课程名	教师姓名
	学时	课程号	学时	教师性别
	（归属）教研室	课程名	（任课）教研室	教师出生年月
		分数	教师工号	职称
		专业名称	教师姓名	（所在）教研室
			职称	

如果直接将表3.2和表5.1中涉及的5个表的表名作为关系模式名，将各表中的属性作为各关系模式的属性，就可得到如下5个关系模式。

（1）学生学籍关系（学号，姓名，性别，出生日期，籍贯，专业代码，班级）。

(2) 课程归属关系(课程号,课程名,学时,教研室)。

(3) 学习成绩关系(学号,姓名,课程号,课程名,分数,专业名称)。

(4) 授课统计关系(课程号,课程名,学时,教研室,教师工号,教师姓名,职称)。

(5) 教师信息关系(教师工号,教师姓名,教师性别,教师出生年月,职称,教研室)。

按照一般的语义概念分析以上的关系模式可知,一个关系模式中的属性的全体,构成了该关系模式的属性集合(设用U表示关系模式的属性集合)。每个属性的值域(设用D表示各属性的值域集合)实质上构成了对该关系的一种取值约束,并反映了属性集合向属性的值域集合的映射或约束(设用DOM表示属性集U到值域集合D的映射)。

同时可知,在每一个关系表中,都存在着由一些属性的值决定另一些属性的值的数据依赖关系。例如,在课程归属表中,给定一个“课程号”的值,就可确定开设该课程的教研室的名称;在教师信息表中,给定一个“教师工号”的值,就可确定该教师所在的教研室的名称。也就是说,这些关系模式中还存在属性之间的决定因素和被决定因素的关系约束,或进一步可把其看作一个关系模式中的属性之间存在的一个或多个依赖关系,由于这种依赖是用属性的值体现的,所以称为数据依赖;当一般地用属性名集合表示其依赖关系时,就可看作是一种函数依赖(设用F表示函数依赖集合)。

综上可知,当把所有这些要素完整地反映到关系模式的描述中时,就可得到如下结论。即,一个关系模式应该完整地用一个五元组{R,U,D,DOM,F}表示,并一般地记为:

$$R(U,D,DOM,F)$$

其中,R是关系名;U是关系R的全部属性组成的属性集合;D是U中各属性的值域的集合;DOM是属性集U到值域集合D的映射;F是关系R的属性集U上的函数依赖集合。

在本章的关系数据库设计理论的相关概念讨论中,关系名R、属性集U和依赖集F三者相互关联,相互依存;而D和DOM在讨论各概念的描述时则关联性不是很大。所以为了简单起见,在本书后续内容的介绍中,把关系模式简化地看作一个三元组:R(U,F)。当且仅当U上的一个关系R满足F时,将R称为关系模式R(U,F)上的一个关系。

5.2 对关系模式进行规范化设计的必要性

下面通过一个由于构建的关系模式不合理而引起操作异常的例子,来说明对关系模式进行规范化设计的必要性。

对于上面的授课统计关系模式:

```
授课统计关系(课程号,课程名,学时,教研室,教师工号,教师姓名,职称)
```

通过调整其中的属性的顺序,可得用符号形式表示的授课统计关系模式如下。

```
TEACHES(T#,TNAME,TITLEOF,TRSECTION,C#,CNAME,CLASSH)
```

则这个关系模式在使用过程中存在以下操作异常问题。

1. 数据冗余

对于教师所讲的每一门课,有关教师的姓名、职称、所属教研室等信息都要重复存放,造成大量的数据冗余。

2. 更新异常

由于有数据冗余，如果教师的职称或所属教研室有变化，就必须对所有具有教师职称或所属单位的元组进行修改。这不仅增加了更新的代价，而且可能出现一部分数据元组修改了，而另一些元组没有被修改的情况，存在着潜在的数据不一致性。

3. 插入异常

由于这个关系模式的主键由教师工号 T＃和课程号 C＃组成，如果某一位老师刚调来，或某一位老师因为某种原因没有上课，就会由于关系的主键属性值不能为空（NULL）而无法将该教师的姓名、职称、所属教研室等基本信息输入到数据库中。学校的数据库中没有该教师的信息，就相当于该学校没有这位教师，显然不符合实际情况。

4. 删除异常

如果某教师不再上课，则删除该教师所担任的课程就连同该教师的姓名、职称、所属教研室等基本信息都删除了。

以上表明，上述的 TEACHES 关系模式的设计是不合适的。之所以存在这种操作异常，是因为在数据之间存在着一种依赖关系。例如，某位教师的职称或所属教研室只由其教师工号就可确定，而与所上课程的编号无关。

如果将上述的关系模式分解成以下的两个关系模式。

```
TEACHER(T＃,TNAME,TITLEOF,TRSECTION)
COURSE(C＃,CNAME,CLASSH)
```

就不会存在上述的操作异常了。一个教师的基本信息不会因为他没上课而不存在；某门课程也不会因为某学期没有上而被认为是没有开设的课程。

然而，上述关系模式也不一定在任何情况下都是最优的。例如，当要查询某教师所上的有关课程信息时，就要进行两个或两个以上关系的连接运算，而连接运算的代价一般是比较大的。相反，在原来的关系模式中却能直接找到这一查询结果。也就是说，原来的关系模式也有它好的一面。怎样判断一个关系模式是最优的？以什么样的标准判断一个关系模式是最优的即是本章要解决的问题。

本章后续的内容涉及比较严格的定义、定律和公式推导，为了表述方便，如不做特殊声明，总是假设有如下约定。

（1）用大写英文字母 A,B,C,D 等表示关系的单个属性。

（2）用大写字母 U 表示某一关系的属性集（全集），用大写字母 V,W,X,Y,Z 表示属性集 U 的子集。

（3）不再特意地区分关系和关系模式，并用大写字母 R，R_1，R_2，…和 S，S_1，S_2，…表示关系和关系模式。

（4）R(A,B,C)，R(ABC)和 ABC 3 种表示关系模式的方法是等价的。同理，{A_1，…，A_n}和 $A_1 \cdots A_n$ 是等价的，X∪Y 和 XY 是等价的，X∪{A}和 XA 是等价的。

（5）用大写英文字母 F，F_1，…，以及 G 表示函数依赖集。

（6）若 X＝{A,B}，Y＝{C,D}，则 X→Y，{A,B}→{C,D}和 AB→CD 3 种函数依赖表示方法是等价的。

5.3 函数依赖

函数依赖(functional dependency)是关系所表述的信息本身所具有的特性。换言之,函数依赖不是研究关系由什么属性组成或关系的当前值如何确定,而是研究施加于关系的只依赖于值的相等与否的限制。这类限制并不取决于某元组在它的某些分量上取什么值,而仅仅取决于两个元组的某些分量是否相等。正是这种限制,才给数据库模式的设计产生了重大而积极的影响。

5.3.1 函数依赖的定义

定义 5.1 设有关系模式 $R(A_1,A_2,\cdots,A_n)$和属性集 $U=\{A_1,A_2,\cdots,A_n\}$的子集 X、Y。如果对于具体关系 r 的任何两个元组 u 和 v,只要 u[X]=v[X],就有 u[Y]=v[Y],则称 X 函数决定 Y,或 Y 函数依赖 X,记为 X→Y。

【例 5.1】 对于图 1.11 中的学生关系模式:

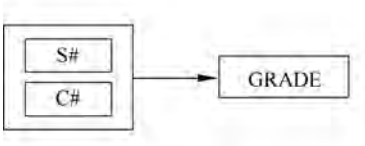
S(S#,SNAME,SSEX,SBIRTHIN,PLACEOFB,SCODE#,CLASS)

有 X={S#},Y={SNAME,SSEX,SBIRTHIN,PLACEOFB,SCODE#,CLASS}。

也就是说,对于如图 1.8 的大学教学信息管理数据库中的学生关系 S 的具体关系 r_s(关系 S 的当前值)来说,不可能同时存在这样的两个元组:它们对于子集 X={S#}中的每个属性有相等的分量,但对于子集 Y={SNAME,SSEX,SBIRTHIN,PLACEOFB,SCODE#,CLASS}中的某个或某些属性具有不相等的分量。例如,对于 X={201401003},只能唯一地找到王丽丽同学的性别为"女",出生年月为 1997-02-02,籍贯为"上海",专业代码为 S0401,班级为 201401。所以有 X→Y。当然,并不是说依据某一具体的关系就可以来验证该关系上的函数依赖是否成立,函数依赖是针对作为关系模式 R 的所有可能的值的。

由函数依赖的定义和例 5.1 可知,若关系模式 R 中属性之间的关系可用一个函数依赖 X→Y 表示时,则决定因素 X 即为该关系的主键。

进一步分析可知,函数依赖是一种语义范畴的概念,所以要从语义的角度来确定各个关系的函数依赖。例如,以学号为唯一主键属性的假设是认为不同的学生一定有不同的、唯一的学号。如果学生不可同名,则{学号,姓名}可作为主键。但事实上由于存在着同名的学生,所以姓名不能作为主键中的属性。否则,对于同一姓名来说,就可能与不同的学生相对应,例如,将{201401001,张华}和{201402001,张华}同时用作为主键值显然是不确切的。图 5.1 用图示方式直观地说明了学习关系 SC 的函数依赖{S#,C#}→{GRADE}。



(a) 函数依赖

学习关系 SC

学　号	课程号	分数
201401001	C401001	90
201402002	C403001	92
201403001	C403002	91

(b) 与图(a)对应的关系

图 5.1　学习关系中的函数依赖

函数依赖描述了每个关系中主属性与非主属性之间的关系。对于关系 $R(A_1,A_2,\cdots,A_n)$和函数依赖 X→Y 来说，属性子集 X 中包括，且仅仅包括了关系 R 的主属性，且对于关系 R 的任何属性子集 Y，一定成立 X→Y。这也就是说，对于 X→Y，可能存在 Y⊆X 和 Y⊈X 两种情况，所以约定：

(1) 若有 X→Y，但 Y⊈X，则称 X→Y 为非平凡函数依赖。若不特别声明，则假定本章所讨论的是非平凡依赖。

(2) 若 X→Y，则称 X 为决定因素。

(3) 若 Y 不依赖于 X，则记作 $X \nrightarrow Y$。

(4) 若同时有 X→Y 和 Y→X，则将其记为 X↔Y。

5.3.2 具有函数依赖约束的关系模式

由函数依赖的概念可知，在一个关系模式中由函数依赖表征的、由一个属性或一组属性组成的被决定因素的值，对由一个属性或一组属性组成的决定因素的值的依赖性，实质上反映了一个关系模式中不同属性集的值之间存在的约束关系。当把所有这种约束反映到对关系模式的描述时，就可以进一步由关系的属性集合和属性间的函数依赖(集合)来表示关系。例如，图 1.11 的大学教学信息管理数据库应用系统的关系模式可进一步表示为如图 5.2 的形式。

```
S ({S#, SNAME, SSEX, SBIRTHIN, PLACEOFB, SCODE#, CLASS},
        {S#}→{SNAME, SSEX, SBIRTHIN, PLACEOFB, SCODE#, CLASS})
SS ({SCODE#, SSNAME}, {SCODE#}→{SSNAME})
C ({C#, CNAME, CLASSH }, {C#}→{CNAME, CLASSH })
CS ({SCODE#, C#}, {SCODE#, C#}→{ SCODE#, C#})
SC ({S#, C#, GRADE }, { S#, C#}→{ GRADE })
T ({T#, TNAME, TSEX, TBIRTHIN, TITLEOF, TRSECTION, TEL},
       {T#}→{TNAME, TSEX, TBIRTHIN, TITLEOF, TRSECTION, TEL})
TEACH ({ T#, C#}, { T#, C#}→{ T#, C#})
```

图 5.2 具有函数依赖约束的关系模式表示示例

就学习关系 SC 来说，有 U={S#,C#,GRADE}，且设 X={S#,C#}，Y={GRADE}和 F={X→Y}；则学习关系可一般地表示为：

SC({S#,C#,GRADE},{X→Y})或 SC(U,F)

5.3.3 函数依赖的逻辑蕴涵

在研究函数依赖时，有时需要根据已知的一组函数依赖，来判断另外一个或一些函数依赖是否成立，或是否能从已知的函数依赖中推导出其他函数依赖来，这就是函数依赖的逻辑蕴涵要讨论的问题。

定义 5.2 设有关系模式 R(U,F)，X、Y 是属性集 $U=\{A_1,A_2,\cdots,A_n\}$的子集，如果从 F 中的函数依赖能够推导出 X→Y，则称 F 逻辑蕴涵 X→Y，或称 X→Y 是 F 的逻辑蕴涵。

所有被 F 逻辑蕴涵的函数依赖组成的依赖集称为 F 的闭包(closure)，记为 F^+。一般来说有 $F \subseteq F^+$。

显然,如果能计算出 F^+,就可以很方便地判断某个函数依赖是否包含在 F^+ 中,即被 F 逻辑蕴涵,但函数依赖集 F 的闭包 F^+ 的计算是一件十分麻烦的事情,即使 F 不大,F^+ 也比较大。例如,有关系模式 R(A,B,C),其函数依赖集为 F={A→B,B→C},则 F 的闭包为:

$$F^+=\begin{Bmatrix}
A\to\Phi, & AB\to\phi, & AC\to\phi, & ABC\to\phi, & B\to\phi, & C\to\phi\\
A\to A, & AB\to A, & AC\to A, & ABC\to A, & B\to B, & C\to C\\
A\to B, & AB\to B, & AC\to B, & ABC\to B, & B\to C, & \\
A\to C, & AB\to C, & AC\to C, & ABC\to C, & B\to BC, & \\
A\to AB, & AB\to AB, & AC\to AB, & ABC\to AB, & BC\to\phi, & \\
A\to AC, & AB\to AC, & AC\to AC, & ABC\to AC, & BC\to B, & \\
A\to BC, & AB\to BC, & AC\to BC, & ABC\to BC, & BC\to C, & \\
A\to ABC, & AB\to ABC, & AC\to ABC, & ABC\to ABC, & BC\to BC, &
\end{Bmatrix}$$

5.4 函数依赖的公理体系

由前述可知,对于关系模式 R(U,F)来说,该关系的主键实质上是由其依赖集 F 中的函数依赖关系确定的。为了从关系模式 R(U,F)的函数依赖集 F 中的函数依赖确定该关系的主键,就需要分析各函数依赖之间的逻辑蕴涵关系,或者至少要根据给定的函数依赖集 F 和函数依赖 X→Y,确定 X→Y 是否属于 F^+,这显然涉及 F^+ 的计算。由于 F^+ 的计算是一件非常复杂的事情,经过一些学者的潜心研究,提出了一组推导函数依赖逻辑蕴涵关系的推理规则,并由 W. W. Armstrong 于 1974 年归纳成公理体系,就形成了著名的 Armstrong(阿姆斯特朗)公理体系。阿姆斯特朗公理体系包括公理和规则两部分。

5.4.1 阿姆斯特朗公理

定义 5.3 设有关系模式 R(U,F)和属性集 U={A_1,A_2,…,A_n}的子集 X、Y、Z、W,阿姆斯特朗公理包括:

- 自反律:若 X⊇Y,则 X→Y。
- 增广律:若 X→Y,则 XZ→YZ。
- 传递律:若 X→Y,Y→Z,则 X→Z。

从理论上讲,公理(定义)是永真而无须证明的,但为了进一步理解函数依赖的定义和加深理解,下面从函数依赖的定义出发,给出公理的正确性证明。

定理 5.1 阿姆斯特朗公理是正确的。

证明:

(1) 证明自反律是正确的。设 r 为关系模式 R 的任意一个关系,u 和 v 为 r 的任意两个元组。

若 u[X]=v[X],则 u 和 v 在 X 的任何子集上必然相等。

由条件 X⊇Y,所以有 u[Y]=v[Y]。

由 r、u、v 的任意性,并根据函数依赖的定义 5.1,可得 X→Y。

(2) 证明增广律是正确的。设 r、u 和 v 的含义同上,并设 u[XZ]=v[XZ],则 u[X]u[Z]=v[X]v[Z]。

由条件 X→Y，若 u[X]＝v[X]，则 u[Y]＝v[Y]，并推知 u[Z]＝v[Z]。

所以 u[Y]u[Z]＝v[Y]v[Z]，则有 u[YZ]＝v[YZ]。

根据函数依赖的定义 5.1，可得 XZ→YZ。

(3) 证明传递律是正确的。设 r、u 和 v 的含义同上，由条件 X→Y，若 u[X]＝v[X]，则 u[Y]＝v[Y]。

又由条件 Y→Z，若 u[Y]＝v[Y]，则 u[Z]＝v[Z]。

所以推知若 u[X]＝v[X]，则 u[Z]＝v[Z]。

根据函数依赖的定义 5.1，可得 X→Z。　　证毕。

5.4.2　阿姆斯特朗公理的推论

从阿姆斯特朗公理可以得出下面的推论。

推论 5.1(合并规则)：若 X→Y 且 X→Z，则 X→YZ。

推论 5.2(分解规则)：若 X→Y，且 Z⊆Y，则 X→Z。

推论 5.3(伪传递规则)：若 X→Y 且 WY→Z，则 XW→Z。

证明：下面利用阿姆斯特朗公理分别证明 3 个推论的正确性。

1. 证明推论 5.1

由条件 X→Y，并增广律可得 X→XY。

由条件 X→Z，并增广律可得 XY→YZ。

利用传递律，由 X→XY 和 XY→YZ，可得 X→YZ。

2. 证明推论 5.2

已知有 X→Y。由条件 Z⊆Y，并自反律可得 Y→Z。

利用传递律，由 X→Y 和 Y→Z，可得 X→Z。

3. 证明推论 5.3

由条件 X→Y，并增广律可得 XW→WY。

利用传递律，和已知条件 WY→Z，可得 XW→Z。证毕。

【例 5.2】 对于关系模式 R(CITY，STREET，ZIP)，依赖集 F＝{ZIP→CITY，{CITY，STREET}→ZIP}，候选键为{CITY，STREET}和{ZIP，STREET}。请证明成立：{CITY，STREET}→{CITY，STREET，ZIP}和{STREET，ZIP}→{CITY，STREET，ZIP}。现证明后者。

证明：

已知 ZIP→CITY，由增广律可得：

{STREET，ZIP}→{CITY，STREET}。

又已知{CITY，STREET}→ZIP，由增广律可得：

{CITY，STREET}→{CITY，STREET，ZIP}。

由上述所得的两个结论，并根据传递律即可得到：

{STREET，ZIP}→{CITY，STREET，ZIP}。

定理 5.2　如果有关系模式 R(A_1，A_2，…，A_n)，则 X→$A_1A_2\cdots A_n$ 成立的充要条件是 X→A_i(i＝1，2，…，n)均成立。

由合并规则和分解规则即可得到这个定理,证明作为练习留给读者自己完成。

定理5.2的结论为数据库模式设计中各个关系的主键的确定奠定了理论基础。

5.4.3 X关于F的闭包及其计算

判断某个函数依赖是否被F逻辑蕴涵最直接的方法需要计算F^+,但由于F^+的计算比较复杂,人们经过研究提出一种利用X关于F的闭包X_F^+来判断函数依赖X→Y是否被F逻辑蕴涵的方法,且由于X_F^+的计算比较简单,在实际中得到了较好的应用。

1. X关于F的闭包X^+

定义5.4 设关系模式R(U,F)和属性集$U=\{A_1,A_2,\cdots,A_n\}$的子集X。则称所有用阿姆斯特朗公理从F推导出的函数依赖X→A_i的属性A_i组成的集合称为X关于F的闭包,记为X_F^+,通常简记X^+。即

$$X^+=\{A_i \mid \text{用公理从F推出的 } X\rightarrow A_i\}$$

显然$X\subseteq X^+$。

【例5.3】 已知在关系模式R(A,B,C)上有函数依赖$F=\{A\rightarrow B,B\rightarrow C\}$,求X分别等于A、B、C时$X^+$的。

解:

(1) 对于X=A,因为A→A,A→B,且由A→B,B→C可推知A→C,所以$X^+=ABC$。

(2) 对于X=B,因为B→B,B→C,所以$X^+=BC$。

(3) 对于X=C,显然有$X^+=C$。

由例5.3可见,比起F的闭包F^+的计算,X关于F的闭包X^+的计算要简单得多。

下面的定理将会告诉我们利用X^+判断某一函数依赖X→Y能否从F导出的方法。

定理5.3 设有关系模式R(U,F)和属性集$U=\{A_1,A_2,\cdots,A_n\}$的子集X、Y。则X→Y能用阿姆斯特朗公理从F导出的充要条件是$Y\subseteq X^+$。

证明:

① 充分性证明:设$Y=A_1,A_2,\cdots,A_n$,$A_i\subseteq U(i=1,2,\cdots,n)$。假设$Y\subseteq X^+$,由X关于F的闭包$X^+$的定义,对于每一个i,$A_i$能由公理从F导出。再利用合并规则(推论5.1),即可得X→Y。

② 必要性证明:设$Y=A_1,A_2,\cdots,A_n$,$A_i\subseteq U(i=1,2,\cdots,n)$。假设X→Y能由公理导出,根据分解规则(推论5.2)和定理5.2有$X\rightarrow A_1,X\rightarrow A_2,\cdots,X\rightarrow A_n$,由$X^+$的定义可知,$A_i\subseteq X^+(i=1,2,\cdots,n)$,所以$Y\subseteq X^+$。 证毕。

定理5.3的意义是,把判定X→Y是否能从F根据阿姆斯特朗公理导出,即该函数依赖是否被F蕴涵的问题,转化成X^+是否包含Y的问题:即当X^+包含Y时,说明X→Y能从F根据阿姆斯特朗公理导出。

2. X关于F的闭包X^+的计算

下面给出计算X关于F的闭包X^+的方法。

算法5.1 求属性集X关于函数依赖集F的闭包X^+。

输入:关系模式R的全部属性集U,U上的函数依赖集F,U的子集X。

输出:X关于F的闭包X^+。

计算方法：

(1) $X^{(0)}=X$。

(2) 依次考查 F 中的每一个函数依赖，如果 $X^{(i)}$ 包含了某个函数依赖左端的属性，就把该函数依赖右端的属性添加到 $X^{(i)}$ 中，并把该函数依赖从 F 中去掉。即对于 F 中的某个 $V\rightarrow W$，若有 $V\subseteq X^{(i)}$，则有 $X^{(i+1)}=X^{(i)}W$；并得新的 $F=F-\{V\rightarrow W\}$。

(3) 重复第(2)步的过程不断扩充 $X^{(i)}$，直到 F 中剩余的函数依赖的左端属性都不被 $X^{(i)}$ 包含或 F 为空集为止。

(4) 此时的 $X^{(i)}$，即为求得的 X 关于 F 的闭包 X^{+}，输出 X^{+}。

【例 5.4】 已知有函数依赖集 $F=\{AB\rightarrow C,BC\rightarrow D,ACD\rightarrow B,D\rightarrow EH,BE\rightarrow C\}$，属性集 $U=\{A,B,C,D,E,H\}$，$X=BD$，求 X^{+}。

解：

(1) $X^{(0)}=BD$。

(2) 依次考查 F 中的函数依赖，由于 $X^{(0)}$ 包含 $D\rightarrow EH$ 的左端属性 D，所以将其右端属性 EH 添加到 $X^{(0)}$ 中，得到 $X^{(1)}=BDEH$，并通过计算 $F=F-\{D\rightarrow EH\}$ 得到新的 $F=\{AB\rightarrow C,BC\rightarrow D,ACD\rightarrow B,BE\rightarrow C\}$。

(3) 重新依次考查 F 中的函数依赖，由于 $X^{(1)}$ 包含 $BE\rightarrow C$ 的左端属性 BE，所以将其右端属性 C 添加到 $X^{(1)}$ 中，得到 $X^{(2)}=BCDEH$，并得新的 $F=\{AB\rightarrow C,BC\rightarrow D,ACD\rightarrow B\}$。

(4) 重新依次考查 F 中的函数依赖，虽然 $X^{(2)}$ 包含 $BC\rightarrow D$ 的左端属性 BC，但其右端属性 D 已经包含在 $X^{(2)}$ 中，所以得新的 $F=\{AB\rightarrow C,ACD\rightarrow B\}$。

(5) 进一步依次考查 F 中的函数依赖，发现 F 中剩余的函数依赖 $AB\rightarrow C$ 和 $ACD\rightarrow B$ 的左端属性都不被 $X^{(2)}$ 包含，计算终止。

(6) 此时得到 $X^{(2)}=BCDEH$ 即为求得的 X^{+}，输出结果 $X^{+}=BCDEH$。

5.4.4 最小函数依赖集

在最小依赖集的求解中，需要用到函数依赖的等价概念。

1. 函数依赖集的等价与覆盖

在关系数据库中，关系的主键总是隐含着最小性，所以需要寻找与关系 R 的属性集 U 的依赖集 F 等价的最小依赖集。这些涉及函数依赖集的等价与覆盖问题。

定义 5.5 设 F 和 G 是两个函数依赖集，如果 $F^{+}=G^{+}$，则称 F 和 G 等价。如果 F 和 G 等价，则称 F 覆盖 G，同时也称 G 覆盖 F。

下面介绍判断依赖集 F 与 G 等价的方法。

定理 5.4 $F^{+}=G^{+}$ 的充要条件是 $F\subseteq G^{+}$ 和 $G\subseteq F^{+}$。

证明：

(1) 如果两个函数依赖集满足 $F_1\subseteq F_2$，那么 F_1 中的任一逻辑蕴涵必是 F_2 的一个逻辑蕴涵，所以 $F_1^{+}\subseteq F_2^{+}$。

(2) 由闭包的定义可知 $(F_1^{+})^{+}=F_1^{+}$。

(3) 证明必要性：若 $F^{+}=G^{+}$，显然有 $F\subseteq G^{+}$ 和 $G\subseteq F^{+}$。

(4) 证明充分性：若 $F\subseteq G^{+}$，由(1)和(2)可得 $F^{+}\subseteq (G^{+})^{+}=G^{+}$，即 $F^{+}\subseteq G^{+}$；又由 $G\subseteq F^{+}$，同理可得 $G^{+}\subseteq F^{+}$，所以 $F^{+}=G^{+}$。 证毕。

由定理5.4可知,为了判断F和G是否等价,只要判断对于F中的每一个函数依赖X→Y是否都属于G^+,若都属于G^+,则$F\subseteq G^+$。只要发现某一个函数依赖X→Y不属于G^+,就说明$F^+\neq G^+$。对于G中的每一个函数依赖也作同样的处理。如果$F\subseteq G^+$且$G\subseteq F^+$,则F和G等价。

由于计算F^+和G^+是NP难问题,所以在实际中,并不需要计算闭包F^+和G^+,而只要根据定理5.3,分别计算X_F^+和X_G^+。

(1) 为了判断$F\subseteq G^+$,只要对F中的每一个函数依赖X→Y计算X关于G的闭包X_G^+,若$Y\subseteq X_G^+$,则说明X→Y属于G^+。

(2) 同理,为了判断$G\subseteq F^+$,只要对G中的每一个函数依赖X→Y计算X关于F的闭包X_F^+,若$Y\subseteq X_F^+$,则说明X→Y属于F^+。

由函数依赖集的等价可以引出一个重要的结论。

推论5.4 每一个函数依赖集F都被其右端只有一个属性的函数依赖组成的依赖集G所覆盖。

证明:设F中的函数依赖包括两种类型:一种是右端只有一个属性的函数依赖,设V→W为其中的任意一个;另一种是右端为一个以上属性的函数依赖。设X→Y为其中的任意一个,$Y=A_1A_2\cdots A_n$,A_i为单个属性。并令G由所有的V→W和$X\rightarrow A_i(i=1,2,\cdots,n)$组成。

对于G中的任意一个$X\rightarrow A_i$,由于F中有X→Y,根据分解规则即F中有$X\rightarrow A_i(i=1,2,\cdots,n)$;而V→W在G和F中都有,所以$G\subseteq F^+$。

对于F中的任意一个X→Y,由于G中有$X\rightarrow A_i(i=1,2,\cdots,n)$,根据合并规则即G中有X→Y,而V→W在F和G中都有,所以$F\subseteq G^+$。从而$F^+=G^+$。 证毕。

推论5.4说明,任何一个函数依赖集都可转化成由右端只有单一属性的依赖组成的集合。这个结论是下面将要讨论的最小函数依赖集的基础。

2. 最小函数依赖集

定义5.6 满足下列条件的函数依赖集F称为最小函数依赖集。

(1) F中的每一个函数依赖的右端都是单个属性。

(2) 对F中的任何函数依赖X→A,$F-\{X\rightarrow A\}$不等价于F。

(3) 对F中的任何函数依赖X→A和X的任何真子集Z,$(F-\{X\rightarrow A\})\cup\{Z\rightarrow A\}$不等价于F。

在这个定义中,条件(1)保证了F中的每一个函数依赖的右端都是单个属性;条件(2)保证了F中不存在多余的函数依赖;条件(3)保证了F中的每一个函数依赖的左端没有多余的属性。所以F理应是最小的函数依赖集。

3. 最小函数依赖集的求解方法

(1) 用分解规则将F中的所有函数依赖分解成右端为单个属性的函数依赖;

(2) 去掉F中冗余的函数依赖,即对于F中任意一个函数依赖X→Y。

① 设$G=F-\{X\rightarrow Y\}$。

② 求X关于G的闭包X_G^+。

③ 判断X_G^+是否包含Y。如果X_G^+包含Y,则说明G逻辑蕴涵X→Y,X→Y是多余的函数依赖,从而可得到较小的函数依赖集F=G;如果X_G^+不包含Y,则说明X→Y不被G

逻辑隐含，就需要保留 X→Y。

④ 按上述方法逐个考查 F 中的每个函数依赖，直到其中的所有函数依赖都考查完。

(3) 去掉 F 中函数依赖左端多余的属性。即对于 F 中左端是非单属性的函数依赖(XY→A)，当要判断 Y 是不是多余的属性时：

① 设 G ＝ (F－{XY→A})∪{X→A}。

② 求 XY 关于 G 的闭包$(XY)_G^+$。

③ 如果$(XY)_G^+$包含 A，则说明 G 逻辑蕴涵 XY→A，即可以用 X→A 代替 XY→A(Y 是多余的属性)，从而可得到较小的函数依赖集 F＝G；如果$(XY)_G^+$不包含 A，则说明 XY→A 不被 G 逻辑隐含，Y 不是多余的属性，则要保留 XY→A。

④ 当 Y 不是多余属性时，要接着用类似的方法判断 X 是不是多余的属性(即设 G＝(F－{XY→A})∪{Y→A}，按步骤②和③进行判断)。

⑤ 接着按上述方法逐个考查 F 中左端是非单属性的其他函数依赖，直到其中的所有左端是非单属性的函数依赖都考查完。

【例 5.5】 求函数依赖集 F 的最小函数依赖集，其中：

$$F=\left\{\begin{matrix}AB\rightarrow C,C\rightarrow A,BC\rightarrow D,\\ACD\rightarrow B,D\rightarrow EG,CG\rightarrow BD\end{matrix}\right\}$$

解：

(1) 用分解规则将 F 中的所有函数依赖分解成右端为单个属性的函数依赖，可得：

$$F_1=\left\{\begin{matrix}AB\rightarrow C,C\rightarrow A,BC\rightarrow D,ACD\rightarrow B,\\D\rightarrow E,D\rightarrow G,CG\rightarrow B,CG\rightarrow D\end{matrix}\right\}$$

(2) 去掉 F 中冗余的函数依赖。具体判别方法是：从 F 中的第一个函数依赖(例如为 X→Y)开始，求 X 关于 F－{X→Y}的闭包 X^+(即从 F 中去掉 X→Y，然后在剩下的函数依赖中求 X^+)，看 X^+是否包含 Y。如果 X^+包含 Y，则说明 X→Y 是多余的函数依赖，因为在 F－{X→Y}中逻辑蕴涵 X→Y，所以从 F 中去掉 X→Y；如果 X^+不包含 Y，则保留 X→Y。按这样的方法逐个考查 F 中的函数依赖，直到其中所有函数依赖都考察完。

① 从 F_1 中分别依次去掉 AB→C、C→A、BC→D，求得$(AB)^+$不包含 C、$(C)^+$不包含 A、$(BC)^+$不包含 D，所以 AB→C、C→A、BC→D 都不是冗余的函数依赖。

② 从 F_1 中去掉 ACD→B，有：

$$F_2=\left\{\begin{matrix}AB\rightarrow C,C\rightarrow A,BC\rightarrow D,\\D\rightarrow E,D\rightarrow G,CG\rightarrow B,CG\rightarrow D\end{matrix}\right\}$$

因为$(ACD)^+$＝ABCDEG，包含 B，即由 F_2 可推导出 ACD→B，所以 ACD→B 为冗余的函数依赖。

③ 同理，从 F_2 中分别依次去掉 D→E、D→G、CG→B，求得$(D)^+$不包含 E、$(D)^+$不包含 G、$(CG)^+$不包含 B，所以 D→E、D→G、CG→B 都不是冗余的函数依赖。

④ 从 F_2 中去掉 CG→D，有：

$$F_3=\left\{\begin{matrix}AB\rightarrow C,C\rightarrow A,BC\rightarrow D,\\D\rightarrow E,D\rightarrow G,CG\rightarrow B\end{matrix}\right\}$$

因为$(CG)^+$＝ABCDG，包含 D，即由 F_3 可推导出 CG→D，所以 CG→D 为冗余的函数依赖。F_3 即去掉冗余函数依赖后的函数依赖集。

(3) 去掉左端多余的属性。具体判别方法是：逐个考察F中左端是非单属性的函数依赖(例如为XY→A)，假设要判断Y是不是多余的属性，则要以X→A代替XY→A，判断(F−{XY→A})∪{X→A}是否等价于F。为此，只要判断X→A是否在F^+中。方法是：求X关于F的闭包X^+，如果A不属于X^+，则X→A不在F^+中，说明Y不是多余的属性；如果A属于X^+，则X→A在F^+中，说明Y是多余的属性，就要从F中去掉函数依赖XY→A，而用X→A代替。接着判别X是不是多余的属性。按这样的方法逐个考察F中的左端是非单属性的函数依赖，直到其中所有左端是非单属性的函数依赖都考察完。

① 分别从A→C和B→C代替F_3中的AB→C，求得$(A)^+$不包含C、$(B)^+$不包含C，所以AB→C左端的属性已经是最少了。

② 分别从B→D和C→D代替F_3中的BC→D，求得$(B)^+$不包含D、$(C)^+$不包含D，所以BC→D左端的属性已经是最少了。

③ 分别从C→B和G→B代替F_3中的CG→B，求得$(C)^+$不包含B、$(G)^+$不包含B，所以CG→B左端的属性已经是最少了。

由此可得F的最小函数依赖集为：

$$F_{min}=\left\{\begin{matrix}AB\rightarrow C,C\rightarrow A,BC\rightarrow D,\\D\rightarrow E,D\rightarrow G,CG\rightarrow B\end{matrix}\right\}$$

5.5 关系模式的分解

在数据库应用系统设计中，一方面是为了减少冗余，另一方面是为了解决可能存在的插入、删除和修改等的操作异常，常常需要将包含属性较多的关系模式分解成几个包含属性较少的关系模式，这就涉及关系模式的分解问题。

5.5.1 关系模式分解的概念

依据关系模式为R(U,F)的约定，对关系模式的分解必然涉及对依赖集F中的各依赖的划分。这种划分是依据分解成的各子关系模式的属性来进行的，因而涉及F在各子关系模式的属性集U_i上的投影概念。

1. F在U_i上的投影

定义5.7 设U_i是属性集U的一个子集，则函数依赖集合$\{X\rightarrow Y\mid X\rightarrow Y\in F^+\wedge XY\subseteq U_i\}$的一个覆盖$F_i$称为F在$U_i$上的投影。

按照依赖集覆盖(等价)的定义，定义5.7的含义是，对于属性集U的子集U_i，存在一个与其对应的依赖子集F_i，且由F_i中的每个函数依赖X→Y的决定因素X和被决定因素Y组成的属性子集XY均是U_i的子集。

在下面的关系模式分解定义中将会看到，当一个关系模式R(U,F)分解时，除了要将其属性集$U=\{A_1,A_2,\cdots,A_n\}$分解成k个属性子集U_i且$1\leqslant i\leqslant k$外，与其对应地也要将依赖集F分解成k个依赖子集F_i且$1\leqslant i\leqslant k$。而定义5.7的意义就在于给出了组成F_i中的各函数依赖应满足的条件。

在定义5.7的基础上，可进一步给出意义更明确的F在U_i上的投影的定义。

定义5.8 设有关系模式R(U,F)和$U=\{A_1,A_2,\cdots,A_n\}$的子集Z，把F^+中所有满足

$XY \subseteq Z$ 的函数依赖 $X \rightarrow Y$ 组成的集合，称为依赖集 F 在属性集 Z 上的投影，记为 $\pi_Z(F)$。

由定义 5.8 可知，显然 $\pi_Z(F)=\{X \rightarrow Y \mid X \rightarrow Y \in F^+$，且 $XY \subseteq Z\}$。所以定义 5.8 和定义 5.7 本质上是相同的。

2. 关系模式的分解

定义 5.9 设有关系模式 R(U,F)，如果 $U=U_1 \cup U_2 \cup \cdots \cup U_k$，并且对于任意的 i,j($1 \leqslant i, j \leqslant k$)，不成立 $U_i \subseteq U_j$，且 F_i 是 F 在 U_i 上的投影。则称 $\rho=\{R_1(U_1,F_1), R_2(U_2,F_2), \cdots, R_k(U_k,F_k)\}$是关系模式 R(U,F)的一个分解。

定义 5.9 中的关键点是，对于属性子集集合$\{U_1, U_2, \cdots, U_k\}$中的任意属性子集 U_i 及 $1 \leqslant i \leqslant k$ 和 U_j 及 $1 \leqslant j \leqslant k$，既不成立 $U_i \subseteq U_j$，也不成立 $U_j \subseteq U_i$。即属性子集集合$\{U_1, U_2, \cdots, U_k\}$中的任意一个属性子集不是其他任何一个属性子集的子集。

显然，当一个关系模式分解成多个关系模式时，相应于该关系中的数据也要被分布到分解成的多个关系中去。

【例 5.6】 已知关系模式 R(U,F)，U={S#,SD,SM}，F={S#→SD,SD→SM}，并设关系 R 有如图 5.3 的当前值 r。

S#	SD	SM
S1	D1	M1
S2	D2	M1
S3	D3	M2

图 5.3 例 5.6 关系 R 的当前值 r

解：下面采用 3 种不同的方式对关系 R 进行分解。

1) 方法 1

设 $\rho_1=\{R_1(S\#,\phi), R_2(SD,\phi), R_3(SM,\phi)\}$，即分解后的子关系模式中只有属性子集 U_i，而 $F_i=\phi$(ϕ 表示空集)。

对已知具体关系 r 在 U_i 上投影，即 $R_i=R(U_i)$可得：

$$R_1=R(U_1)=\{S1,S2,S3\}$$
$$R_2=R(U_2)=\{D1,D2,D3\}$$
$$R_3=R(U_3)=\{M1,M2\}$$

对关系模式进行分解的目的是克服可能出现的操作异常等，但分解前后关系中原有的信息应当保持不变。一般采用的方法是用各 R_i 的自然连接恢复 R。所以由：

$$R_1 \bowtie R_2 \bowtie R_3 = R_1 \times R_2 \times R_3 \quad \text{（由于各 } R_i \text{ 间无属性相同的列）}$$

得：

$$\left\{\begin{array}{lll} S1\ D1\ M1, & S2\ D1\ M1, & S3\ D1\ M1 \\ S1\ D1\ M2, & S2\ D1\ M2, & S3\ D1\ M2 \\ S1\ D2\ M1, & S2\ D2\ M1, & S3\ D2\ M1 \\ S1\ D2\ M2, & S2\ D2\ M2, & S3\ D2\ M2 \\ S1\ D3\ M1, & S2\ D3\ M1, & S3\ D3\ M1 \\ S1\ D3\ M2, & S2\ D3\ M2, & S3\ D3\ M2 \end{array}\right\}$$

比较给定关系 R 的当前值 r 和所得元组集合(通过自然连接恢复的结果)可知，恢复后

的关系R的当前值r已经丢失了原来信息的真实性(比原来的元组数多出了许多)。所以方法1的分解是有损原来信息的一种分解,或者说方法1的分解不保持信息无损,不具有无损连接性。

2) 方法2

设 $\rho_2=\{R_1(\{S\#,SD\},\{S\#\rightarrow SD\}),R_2(\{S\#,SM\},\{S\#\rightarrow SM\})\}$。

通过将已知具体关系r在U_i上投影,可以证明这种分解能够恢复原来的具体关系,但分解不保持函数依赖。因为:

$$F_1^+=\begin{Bmatrix}S\#\rightarrow\phi,\quad S\#\ SD\rightarrow\phi,SD\rightarrow\phi\\S\#\rightarrow S\#,\quad S\#\ SD\rightarrow S\#,SD\rightarrow SD\\S\#\rightarrow SD,\quad S\#\ SD\rightarrow SD\\S\#\rightarrow S\#\ SD,S\#\ SD\rightarrow S\#\ SD\end{Bmatrix}$$

$$F_2^+=\begin{Bmatrix}S\#\rightarrow\phi,\quad S\#\ SM\rightarrow\phi,SM\rightarrow\phi\\S\#\rightarrow S\#,\quad S\#\ SM\rightarrow S\#,SM\rightarrow SM\\S\#\rightarrow SM,\quad S\#\ SM\rightarrow SM\\S\#\rightarrow S\#\ SM,S\#\ SM\rightarrow S\#\ SM\end{Bmatrix}$$

根据定义5.7和定义5.9,关系R的依赖集F中的函数依赖SD→SM既不在F_1^+中,也不在F_2^+,即函数依赖SD→SM既不在R_1的F_1中,也不在R_2的F_2中。所以,方法2的分解能够保持信息无损,但不保持函数依赖(详见例5.12)。

3) 方法3

设 $\rho_3=\{R_1(\{S\#,SD\},\{S\#\rightarrow SD\}),R_2(\{SD,SM\},\{SD\rightarrow SM\})\}$。

可以证明,方法3的分解既能保持信息无损(详见例5.10),又能保持函数依赖(详见例5.11)。

如果关系的一个分解既能保持无损连接,又能保持函数依赖,它就既能解决操作异常,又不会丢失原有数据的信息,这正是关系模式分解中所希望的。

5.5.2 保持无损的分解

定义5.10 设有关系模式R(U,F),$\rho=(R_1,R_2\cdots,R_k)$是R的一个分解。如果对于R的任一满足F的关系r,成立:

$$r=\pi_{R_1}(r)\bowtie\pi_{R_2}(r)\bowtie\cdots\bowtie\pi_{R_K}(r)$$

则称该分解ρ是无损连接分解,也称该分解ρ是保持无损的分解。

上述定义说明,r是它在各R_i上的投影的自然连接。按照自然连接的定义,上式中两个相邻的关系在做自然连接时,如果至少有一个列名相同,则为自然连接;如果没有相同的列名,则为笛卡儿乘积运算。

为了表述问题方便,约定:

(1) 把r在ρ上的投影的连接记为$m_\rho(r)$,且$m_\rho(r)=\bowtie_{i=1}^{k}\pi_{R_i}(r)$。于是,关系模式R的满足F的无损分解的条件可以表示成:对所有满足F的关系r,有$r=m_\rho(r)$。

(2) 设t是一个元组,X是一个属性集,用t[X]表示元组t在属性集X上的属性值序列。此时有$\pi_X(r)=\{t[X]\mid t\text{在}r\text{中}\}$。

算法 5.2 判断一个分解是无损连接分解，即该分解是否具有无损连接性。

输入：关系模式 $R(A_1,A_2,\cdots,A_n)$，函数依赖集 F，R 的一个分解 $\rho=(R_1,R_2,\cdots,R_k)$。

输出：ρ 是否为无损连接的判断。

方法：

(1) 构造一个 k 行 n 列表，其中，第 i 行对应于关系模式 R 分解后的模式 R_i，第 j 列对应于关系模式 R 的属性 A_j。表中第 i 行第 j 列位置的元素填入方法为：如果 A_j 在 R_i 中，则在第 i 行第 j 列的位置上填上符号 a_j，否则填上符号 b_{ij}。

(2) 对于 F 中的所有函数依赖 X→Y，在表中寻找在 X 的各个属性上都分别相同的行，若存在两个或多个这样的行，则将这些行上对应于 Y 属性上的元素值，修改成具有相同符号的元素值，修改方法如下。

① 当这些行的 Y 属性上的某一行上都为 a_j（j 为 Y 属性对应的那些列的列序号，且 j=1,2,…,n）时，则把这些行的 Y 属性列上的元素值都修改成 a_j；

② 当这些行的 Y 属性列中没有这样的 a_j 时，则以 Y 属性列中下标较小的 b_{ij} 为基准，把这些行的 Y 属性列上的其他元素值都修改成 b_{ij}。

(3) 按(2)逐个考查 F 中的每一个函数依赖，如果发现某一行变成了 $a_1,a_2,\cdots,a_n$，则分解 ρ 具有无损连接性；如果直到检验完 F 中的所有函数依赖也没有发现有这样的行，则分解 ρ 不具有无损连接性。

【例 5.7】 设有关系模式 R(A,B,C,D,E)，函数依赖集 F={A→C,B→C,C→D,DE→C,CE→A}，分解 $\rho=\{R_1,R_2,R_3,R_4,R_5\}$，其中 $R_1=AD$，$R_2=AB$，$R_3=BE$，$R_4=CDE$，$R_5=AE$。检验分解 ρ 是否具有无损连接性。

解：

(1) 构造一个 5 行 5 列表，并按算法 5.2 的(1)填写表中的元素，如图 5.4 所示。

R_i	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{23}	b_{24}	b_{25}
R_3	b_{31}	a_2	b_{33}	b_{34}	a_5
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	b_{53}	b_{54}	a_5

图 5.4 例 5.7 第(1)步结果

(2) 对于函数依赖 A→C，在表中 A 属性列的 1、2、5 行都为 a_1，在 C 属性列的 1、2、5 行不存在 a_3，所以以该列第 1 行的 b_{13} 为基准，将该列 2、5 行位置的元素修改成 b_{13}，其结果如图 5.5 所示。

R_i	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{13}	b_{24}	b_{25}
R_3	b_{31}	a_2	b_{33}	b_{34}	a_5
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	b_{13}	b_{54}	a_5

图 5.5 例 5.7 第(2)步结果

(3) 对于函数依赖 B→C,在表中 B 属性列的 2、3 行都为 a_2,在 C 属性列的 2、3 行不存在 a_3,所以以该列第 2 行的 b_{13} 为基准,将该列 3 行位置的元素修改成 b_{13},其结果如图 5.6 所示。

R_i	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{13}	b_{24}	b_{25}
R_3	b_{31}	a_2	b_{13}	b_{34}	a_5
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	b_{13}	b_{34}	a_5

图 5.6　例 5.7 第(3)步结果

(4) 对于函数依赖 C→D,在表中 C 属性列的 1、2、3、5 行都为 b_{13},在 D 属性列的 1 行存在 a_4,所以将该列 2、3、5 行位置的元素修改成 a_4,其结果如图 5.7 所示。

R_i	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{13}	a_4	b_{25}
R_3	b_{31}	a_2	b_{13}	a_4	a_5
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	b_{13}	a_4	a_5

图 5.7　例 5.7 第(4)步结果

(5) 对于函数依赖 DE→C,在表中 DE 属性列的 3、4、5 行都相同,在 C 属性列的 4 行存在 a_3,所以将该列 3、5 行位置的元素修改成 a_3,其结果如图 5.8 所示。

R_i	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{13}	a_4	b_{25}
R_3	b_{31}	a_2	a_3	a_4	a_5
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	a_3	a_4	a_5

图 5.8　例 5.7 第(5)步结果

(6) 对于函数依赖 CE→A,在表中 CE 属性列的 3、4、5 行都相同,在 A 属性列的 5 行存在 a_1,所以将该列 3、4 行位置的元素修改成 a_1,其结果如图 5.9 所示。

R_i	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{13}	a_4	b_{25}
R_3	a_1	a_2	a_3	a_4	a_5
R_4	a_1	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	a_3	a_4	a_5

图 5.9　例 5.7 第(6)步结果

这时表中的第 3 行变成了 $a_1, a_2, \cdots, a_5$,所以分解 ρ 具有无损连接性。

【例 5.8】　设有关系模式 R(A,B,C),函数依赖集 F={A→B,C→B},分解 ρ={R_1, R_2},其中 R_1=AB,R_2=BC。检验分解 ρ 是否具有无损连接性。

解：

(1) 构造 2 行 3 列表，并按算法 5.2 的(1)填写表中的元素，如图 5.10 所示。

R_1	A	B	C
R_1	a_1	a_2	b_{13}
R_2	b_{21}	a_2	a_3

图 5.10　例 5.8 第(1)步结果

(2) 对于函数依赖 A→B，在表中 A 属性列中无相同的行，继续考查下一个依赖。

(3) 对于函数依赖 C→B，在表中 C 属性列中无相同的行。

至此，考查完依赖集中所有的函数依赖，表中不存在元素为 a_1、a_2、a_3 的行，所以分解不具有无损连接性，即分解 ρ 不是无损连接分解。

算法 5.2 适用于关系模式 R 向任意多个关系模式的分解。如果只限于将 R 分解为两个关系模式时，下面的定理给出了更简单的检验方法。

定理 5.5　设有关系模式 R(U,F)，$\rho=(R_1,R_2)$ 是 R 的一个分解，当且仅当 $(R_1 \cup R_2)\rightarrow(R_1-R_2)\in F^+$ 或 $(R_1 \cup R_2)\rightarrow(R_2-R_1)\in F^+$ 时，ρ 具有无损连接性。

证明：分别将 $R_1 \cap R_2$、R_1-R_2、R_2-R_1 看成不同的属性集，并利用算法 5.2 构造如图 5.11 的 2 行 k 列表。

R_i	$R_1 \cap R_2$	R_1-R_2	R_2-R_1
R_1	$a_1 \cdots a_i$	$a_{i+1}\cdots a_j$	$b_{1,j+1}\cdots b_{1,k}$
R_2	$a_1 \cdots a_i$	$b_{2,i+1}\cdots b_{2,j}$	$a_{j+1}\cdots a_k$

图 5.11　利用算法 5.2 构造 2 行 k 列表

(1) 充分性：假设 $(R_1 \cup R_2)\rightarrow(R_1-R_2)$ 在 F 中，由算法 5.2 可将表中第 2 行的 $b_{2,i+1}\cdots b_{2,j}$ 改成 $a_{i+1}\cdots a_j$，使第 2 行变成 $a_1\cdots a_k$。因此分解 ρ 具有无损连接性。

如果 $(R_1 \cup R_2)\rightarrow(R_1-R_2)$ 不在 F 中，但在 F^+ 中，则可用公理从 F 中推出 $(R_1 \cup R_2)\rightarrow A_y$，其中 $A_y\in(R_1-R_2)$，即 A_y 是 R_1-R_2 中的任一属性。所以利用算法 5.2 可以将属性 A_y 列所对应的第 2 行中的 b_{2y} 该为 a_y，这样修改后的第 2 行就变成了 $a_1\cdots a_k$，所以分解 ρ 具有无损连接性。

同理对于 $(R_1 \cup R_2)\rightarrow(R_2-R_1)$，可类似地证得表中的第 1 行为 $a_1\cdots a_k$。

(2) 必要性：假设分解 ρ 具有无损连接性，那么按照算法 5.2 构造的表中必有一行为 $a_1\cdots a_k$。按照算法 5.2 的构造方法，若第 2 行为 $a_1\cdots a_k$，则意味着成立 $(R_1 \cup R_2)\rightarrow(R_1-R_2)$；若第 1 行为 $a_1\cdots a_k$，则意味着成立 $(R_1 \cup R_2)\rightarrow(R_2-R_1)$。　　证毕。

上面的必要性证明也可以这样来理解：因为根据定义 5.10，如果关系模式 R 的分解 ρ 是满足函数依赖集 F 的无损连接分解，则对于 R 的任何一个满足 F 的具体关系 r，必然有或者 $((R_1 \cup R_2)\rightarrow(R_1-R_2))\in F$，或者用公理可由 F 推导出 $(R_1 \cup R_2)\rightarrow(R_1-R_2)$。同理有 $(R_1 \cup R_2)\rightarrow(R_2-R_1)$ 的情况。

【例 5.9】　在例 5.8 中，设有关系模式 R(A,B,C)，函数依赖集 $F=\{A\rightarrow B, C\rightarrow B\}$，分解 $\rho=\{R_1,R_2\}$，其中 $R_1=AB$，$R_2=BC$。检验分解 ρ 是否具有无损连接性。

解：

因为：
$$(R_1 \cup R_2)\rightarrow(R_1-R_2) = (AB \cup BC)\rightarrow AB-BC = (B\rightarrow A)\notin F$$
$$(R_2 \cup R_1)\rightarrow(R_2-R_1) = (BC \cup AB)\rightarrow BC-AB = (B\rightarrow C)\notin F$$

进一步分析可知,$(B\rightarrow A)\notin F^+$ 且 $(B\rightarrow C)\notin F^+$

所以,分解 ρ 不具有无损连接性。

【例5.10】 在例5.6中,已知有关系模式R(S#,SD,SM),函数依赖集F={S#→SD,SD→SM}。且对于其中的方法3,已知有分解 $\rho_3=\{R_1(S\#,SD),R_2(SD,SM)\}$,且已指出分解 ρ_3 是无损连接分解,下面进行验证。

解:

虽然 $(R_1\cup R_2)\rightarrow(R_1-R_2)=(\{S\#,SD\}\cup\{SD,SM\})\rightarrow(\{S\#,SD\}-\{SD,SM\})$

$$=SD\rightarrow S\#\notin F^+$$

但 $(R_2\cup R_1)\rightarrow(R_2-R_1)=(\{SD,SM\}\cup\{S\#,SD\})\rightarrow(\{SD,SM\}-\{S\#,SD\})$

$$=SD\rightarrow SM\in F^+$$

所以,分解 ρ_3 具有无损连接性。

这里要特别指出,定理5.5并不要求 $(R_1\cup R_2)\rightarrow(R_1-R_2)\in F^+$ 和 $(R_2\cup R_1)\rightarrow(R_2-R_1)\in F^+$ 同时成立;而是只要 $(R_1\cup R_2)\rightarrow(R_1-R_2)\in F^+$ 或 $(R_2\cup R_1)\rightarrow(R_2-R_1)\in F^+$ 成立即可。

5.5.3 保持依赖的分解

由前面的讨论可知,要求关系模式的分解具有无损连接性是必要的,因为它保证了分解后的子模式不会引起信息的失真。在关系模式的分解中,还有一个重要的性质就是分解后的子模式还应保持原有的函数依赖,即分解后的子模式应保持原有关系模式的完整性约束。这就是保持依赖的分解要讨论的问题。

定义5.11 设有关系模式R(U,F),$\rho=\{R_1,R_2,\cdots,R_k\}$ 是R上的一个分解。如果所有函数依赖集 $\pi_{Ri}(F)(i=1,2,\cdots,k)$ 的并集逻辑蕴涵F中的每一个函数依赖,则称分解ρ具有依赖保持性,即分解ρ保持依赖集F。

【例5.11】 在例5.6中,已知有关系模式R(S#,SD,SM),函数依赖集F={S#→SD,SD→SM}。且对于其中的方法3,已知有分解 $\rho_3=\{R_1(S\#,SD),R_2(SD,SM)\}$,且F1={S#→SD},F2={SD→SM}。例5.10已经验证了分解 ρ_3 具有无损连接性,下面验证分解 ρ_3 是保持依赖的分解。

解:

因为 $\pi_{R_1}(F)\cup\pi_{R_2}(F)=\{S\#\rightarrow SD\}\cup\{SD\rightarrow SM\}$

$$=\{S\#\rightarrow SD,SD\rightarrow SM\}$$

$$=F$$

所以,ρ_3 具有保持依赖性。

【例5.12】 在例5.6中,已知有关系模式R(S#,SD,SM),函数依赖集F={S#→SD,SD→SM}。且对于其中的方法2,已知有分解 $\rho_2=\{R_1(\{S\#,SD\},\{S\#\rightarrow SD\}),R_2(\{S\#,SM\},\{S\#\rightarrow SM\})\}$。验证分解 ρ_2 具有无损连接性,但不具有保持依赖性。

解:

(1) 验证无损连接性。

因为 $(R_1\cup R_2)\rightarrow(R_1-R_2)=(\{S\#,SD\}\cup\{S\#,SM\})\rightarrow(\{S\#,SD\}-\{S\#,SM\})$

$$=S\#\rightarrow SD\in F$$

所以分解 ρ_2 具有无损连接性。

(2) 验证不具有保持依赖性。

因为 $\pi_{R_1}(F) \cup \pi_{R_2}(F) = \{S\# \rightarrow SD\} \cup \{S\# \rightarrow SM\}$

$= \{S\# \rightarrow SD, S\# \rightarrow SM\}$

显然,$(SD \rightarrow SM) \notin \{S\# \rightarrow SD, S\# \rightarrow SM\}$

即 $\pi_{R_1}(F)$和 $\pi_{R_2}(F)$的并集不逻辑蕴涵 F 中的函数依赖 SD→SM。所以分解 ρ_2 不具有保持依赖性。

由例 5.10 至例 5.12 可知,一个关系模式的分解有以下 4 种情况。

(1) 既具有无损连接性,又具有保持依赖性。

(2) 具有无损连接性,但不具有保持依赖性。

(3) 不具有无损连接性,但具有保持依赖性。

(4) 既不具有无损连接性,又不具有保持依赖性。

显然,符合要求的关系模式分解应是第(1)种情况。

5.6 关系模式的规范化

利用某种约束条件对关系模式进行规范化后,就会使该关系模式变成一种规范化形式的关系模式,这种规范化形式的关系模式就称为范式(Normal Form,NF)。根据规范化程度的不同,范式分为第一范式(1NF)、第二范式(2NF)、第三范式(3NF)、"鲍依斯-柯德"范式(BCNF)等。显然最低一级的范式是 1NF。可以把范式的概念理解成符合某一条件的关系模式的集合,这样如果一个关系模式 R 为第 x 范式,就可以将其写成 R∈xNF。

一般把从低一级的范式通过模式分解达到高一级范式的过程称为关系模式的规范化。

由于在判断一个关系模式属于第几范式时,需要知道该关系模式的候选键,所以下面先介绍关系模式候选键的求解方法,然后再依次介绍各个范式。

5.6.1 候选键的求解方法

1. 关系属性的分类

定义 5.12 对于给定的关系模式 S(U,F),关系 S 的属性按其在函数依赖的左端和右端的出现分为以下 4 类。

(1) L 类:仅在 F 中的函数依赖左端出现的属性称为 L 类属性。

(2) R 类:仅在 F 中的函数依赖右端出现的属性称为 R 类属性。

(3) LR 类:在 F 中的函数依赖的左右两端都出现过的属性称为 LR 类属性。

(4) N 类:在 F 中的函数依赖的左右两端都未出现过的属性称为 N 类属性。

【例 5.13】 设有关系模式 S(A,B,C,D),和 S 的函数依赖集 F={A→C,B→AC,D→AC,BD→A}。请指出关系 S 的属性分类。

解:分析可知 L 属性有 BD;R 属性有 C;LR 属性有 A。

2. 候选键的充分条件

定义 5.13 设有关系模式 R(U,F)和属性集 $U=\{A_1, A_2, \cdots, A_n\}$的子集 X。

(1) 若 X 是 R 类属性,则 X 不是任一候选键的成员。

(2) 若X是N类属性,则X必包含在R的某一候选键中。

(3) 若X是L类属性,则X必为R的某一候选键的成员。

(4) 若X是L类属性,且X^+包含R的全部属性,则X必为R的唯一候选键;

(5) 若X是R的L类属性和N类属性组成的属性集,且X^+包含R的全部属性,则X是R的唯一候选键。

3. 单属性候选键的依赖图判定方法

定义5.14 设关系模式R(U,F)的依赖集F已经是最小依赖集(即蕴涵依赖右端只有单个属性),则关系模式R的函数依赖图G是一个有序二元组G=(R,F),且:

(1) $U=\{A_1,A_2,\cdots,A_n\}$是一个有限非空集,A_i是G中的结点。

(2) F是G的一个非空边集,$A_i\rightarrow A_j$是G中的一条有向边(A_i,A_j)。

(3) 若结点A_i与结点A_j之间存在一条有向边(A_i,A_j),则该边称为A_i的出边和A_j的入边。

(4) 只有出边而无入边的结点称为原始点(表示L类属性);只有入边而无出边的结点称为终结点(表示R类属性);既有入边又有出边的结点称为途中点(表示LR类属性);既无入边又无出边的结点称为孤立点(表示N类属性)。

(5) 原始点和孤立点统称为关键点;关键点对应的属性称为关键属性;其他结点不能到达的回路称为独立回路。

定义5.14实质上就是函数依赖图的构建方法。

算法5.3 单个属性候选键的依赖图判定算法。

输入:关系模式$R(A_1,A_2,\cdots,A_n)$,R的单属性函数依赖集F。

输出:R的所有候选键。

方法:

(1) 求F的最小依赖集F_{min}。

(2) 构造函数依赖图。

(3) 从图中找出关键属性集X(X可为空)。

(4) 查看G中有无独立回路,若无则X即为R的唯一候选键,转(6)。

(5) 从各独立回路中各取一结点对应的属性与X组合成一候选键,并重复这一过程,取尽所有可能的组合,即为R的全部候选键。

(6) 输出候选键,算法结束。

4. 多属性候选键的求解方法

算法5.4 多属性候选键的判定算法

输入:关系模式$R(A_1,A_2,\cdots,A_n)$,R的单属性函数依赖集F。

输出:R的所有候选键。

方法:

(1) 将R的所有属性分为L、R、N和LR 4类,并令X代表L和N类,Y代表LR类。

(2) 求X_F^+:若X_F^+包含R的全部属性,则X是R的唯一候选键,转(8)。

(3) 在Y中取一属性A,并求$(XA)_F^+$:若$(XA)_F^+$包含R的全部属性,则XA为R的一个候选键。

(4) 重复(3),直到Y中的属性依次取完。

(5) 从 Y 中除去所有已成为主属性的属性 A。

(6) 在剩余的属性中依次取两个属性、3 个属性，…，将其记为集合 B，并求 $(XB)_F^+$：若 $(XB)_F^+$ 包含 R 的全部属性，且自身不包含已求出的候选键，则 XB 为 R 的一个候选键；

(7) 重复(6)，直到 Y 中的属性按(6)的组合依次取完；

(8) 输出候选键，算法结束。

【例 5.14】 设有关系模式 R(A,B,C,D,E)和 R 的函数依赖集 F={A→BC,CD→E,B→D,E→A}，求 R 的所有候选键。

解：

(1) 根据 F 对 R 的所有属性进行分类：ABCDE 均为 LR 类属性，并令 Y=ABCDE；没有 L 类、R 类和 N 类属性。

(2) 从 Y 中依次取一个属性，并计算该属性关于 F 的闭包。

A^+=ABCDE，包含 R 的全部属性，所以 A 为 R 的一个候选键。

B^+=BD，没有包含 R 的全部属性，所以 B 不是 R 的候选键。

C^+=C，没有包含 R 的全部属性，所以 C 不是 R 的候选键。

D^+=D，没有包含 R 的全部属性，所以 D 不是 R 的候选键。

E^+=ABCDE，包含 R 的全部属性，所以 E 为 R 的一个候选键。

(3) 从 Y 中去掉已经是候选键中的属性 A 和 E，并令 Y=BCD。

(4) 再从 Y 中依次取两个属性，并计算该属性集合关于 Y 的闭包。

$(BC)^+$=ABCDE，包含 R 的全部属性，所以 BC 为 R 的一个候选键。

$(BD)^+$=BD，没有包含 R 的全部属性，所以 BD 不是 R 的候选键。

$(CD)^+$=ABCDE，包含 R 的全部属性，所以 CD 为 R 的一个候选键。

(5) 由于 B、C、D 也已经是主属性了，不需要考查从 Y 中取三个属性的情况了，候选键求解到此结束。

综上可知，R 的候选键有 A、E、BC 和 CD。

【例 5.15】 设有关系模式 R(A,B,C,D,E)和 R 的函数依赖集 F={AB→C,C→D,D→B,D→E}，求 R 的所有候选键。

解：

(1) 根据 F 对 R 的所有属性进行分类：A 为 L 类属性；BCD 均为 LR 类属性，并令 Y=BCD；E 为 R 类属性，没有 N 类属性。

(2) A^+=A；A^+不包含 R 的所有属性，所以 A 不是 R 的唯一候选键，但 A 为 L 属性，因此 A 必为 R 的候选键的成员。

(3) 从 Y 中取出一个属性 B，求得 $(AB)^+$=ABCDE，$(AB)^+$包含 R 的全部属性，所以 AB 是 R 的一个候选键。

(4) 从 Y 中取出一个属性 C，求得 $(AC)^+$=ABCDE，$(AC)^+$包含 R 的全部属性，所以 AC 是 R 的一个候选键。

(5) 从 Y 中取出一个属性 D，求得 $(AD)^+$=ABCDE，$(AD)^+$包含 R 的全部属性，所以 AD 是 R 的一个候选键。

(6) 由于 Y 中的 B、C、D 均已经是主属性了，不需要考查从 Y 中取三个属性的情况了，候选键求解到此结束。

综上可知，R 的候选键有 AB、AC 和 AD。

5.6.2 第一范式(1NF)

定义 5.15 如果关系模式 R 中的每一个属性的值域的值都是不可再分的最小数据单位,则称 R 为满足第一范式(1NF)的关系模式,也称 R∈1NF。

当关系 R 的属性值域中的值都是不可再分的最小数据单位时,就表示二维表格形式的关系中不再有子表。

为了与规范关系相区别,有时把某些属性有重复值(表中有子表)或空白值的二维表格称为非规范关系。图 5.12 给出了两个非规范关系。

DEPNAME	LOC	S-PART
DEP1	XIAN	P1
		P2
DEP2	WUHAN	P1
		P3
DEP3	CHENGDU	P2

(a) 属性S-PART有重复值

TNAME	ADDRESS	PHONE
徐浩	5-1-2	88992
张明敏	12-2-4	88518
李阳洋	6-4-7	88826
宋歌	23-3-8	
郭宏伟	10-2-3	88158

(b) 属性PHONE有空白值

图 5.12 非规范关系示例

对于有重复值的非规范关系,一般采用把重复值所在行的其他属性的值也予以重复的方法将其转换成规范关系。对于有空白值的非规范关系,由于目前的数据库管理系统支持"空值"处理功能,所以采用的方法是将空白值赋予空值标志(NULL)。图 5.12 的非规范关系对应的规范关系如图 5.13 所示。

DEPNAME	LOC	S-PART
DEP1	XIAN	P1
DEP1	XIAN	P2
DEP2	WUHAN	P1
DEP2	WUHAN	P3
DEP3	CHENGDU	P2

(a) 属性S-PART的规范关系

TNAME	ADDRESS	PHONE
徐浩	5-1-2	88992
张明敏	12-2-4	88518
李阳洋	6-4-7	88826
宋歌	23-3-8	NULL
郭宏伟	10-2-3	88158

(b) 属性PHONE的规范关系

图 5.13 图 5.12 的非规范关系对应的规范关系

5.6.3 第二范式(2NF)

1. 部分依赖与完全依赖

定义 5.16 设有关系模式 R(U,F)和属性集 U=$\{A_1,A_2,\cdots,A_n\}$的子集 X、Y。如果 X→Y,且对于 X 的任何真子集 X′,都有 X′→Y 不成立,则称 Y 完全依赖于 X,记为 $X \xrightarrow{f} Y$。

【例 5.16】 在课程关系 C(C#,CNAME,CLASSH)中,有:

$\{C\#\} \xrightarrow{f} \{CNAME\}$、$\{C\#\} \xrightarrow{f} \{CLASSH\}$和$\{C\#\} \xrightarrow{f} \{CNAME,CLASSH\}$

在学习关系 SC(S#,C#,GRADE)中,有:

$\{S\#\} \rightsquigarrow \{GRADE\}$、$\{C\#\} \rightsquigarrow \{GRADE\}$和$\{S\#,C\#\} \xrightarrow{f} \{GRADE\}$。

定义 5.17 设有关系模式 R(U,F)和属性集 U＝{A_1,A_2,…,A_n}的子集 X、Y。如果 X→Y,但 Y 不完全依赖于 X,则称 Y 部分依赖于 X,记为 $X \xrightarrow{P} Y$。

比较定义 5.16 和定义 5.17 可知,所谓完全依赖,就是不存在 X 的真子集 X′(X′⊆X,X′≠X)使 X′→Y 成立;若存在 X 的真子集 X′使 X′→Y 成立,则称为 Y 部分依赖于 X。

同时,由定义 5.16 和定义 5.17 可知,当 X 是仅包含有一个属性的属性子集时,Y 都是完全依赖于 X 的,只有当 X 是由多个属性组成的属性子集时,才可能会有 Y 完全依赖于 X 和 Y 部分依赖于 X 两种情况。

2. 第二范式

定义 5.18 如果一个关系模式 R 属于 1NF,并且它的每一个非主属性都完全依赖于它的一个候选键,则称 R 为满足第二范式(2NF)的关系模式,也称 R∈2NF。

显然,如果一个关系模式 R 属于 1NF,并且它的主键只由一个属性组成(单属性主键)时,就不可能存在非主属性对候选键的部分依赖,所以 R 一定属于 2NF。如果关系模式 R 的候选键是复合候选键(由多个属性组成的候选键),才可能出现非主属性部分依赖于候选键的情况。显然,如果在一个属于 1NF 的关系模式 R 中存在非主属性对候选键的部分依赖,则 R 就不属于 2NF。

【例 5.17】 对于图 5.14 所示的规范化关系 SCT(S#,C#,GRADE,TNAME,TRSECTION)及其具体关系。该关系的主键为{S#,C#},表示在已知一个学号值和一个课程号值的情况下,就可以获知该学生学习该门课程的分数、(任课)教师名称及其所属的教研室。

S#	C#	GRADE	TNAME	TRSECTION
201401001	C401001	90	徐浩	计算机
201401001	C402002	90	李阳洋	指挥信息系统
201401001	C403001	85	宋歌	通信工程
201401002	C401001	75	徐浩	计算机
201401002	C402002	88	李阳洋	指挥信息系统
201401003	C402002	69	李阳洋	指挥信息系统
201402001	C401001	87	徐浩	计算机
201402001	C401002	90	张国庆	计算机
201402002	C403001	92	宋歌	通信工程

图 5.14 一个 SCT 关系模式的具体关系

但在假设一门课程只能由一位教师讲授的情况下,显然有 C#→TNAME,即关系 SCT 的非主属性 TNAME 部分依赖于侯选键{S#,C#}。所以图 5.14 的关系 SCT 是 1NF 而不是 2NF。然而,当把关系模式 SCT 分解成如图 5.15 的两个关系 SC 和 CT 时,SC 和 CT 既是 1NF,也是 2NF。

注意:当一个关系模式不是 2NF 时,会产生以下问题。

- 插入异常。例如,在上述的 SCT 关系模式中,当某一个新调来的教师还没有担任讲课任务时,就无法登记他的所属教研室信息(TRSECTION)。因为在关系 SCT 中要插入新记录时,必须给定主键的值,而在没有担任讲课任务时,主键中的课程编号由于无法确定而就无法插入。

S#	C#	GRADE
201401001	C401001	90
201401001	C402002	90
201401001	C403001	85
⋮	⋮	⋮

(a) SC

C#	TNAME	TRSECTION
C401001	徐浩	计算机
C402002	李阳洋	指挥信息系统
C403001	宋歌	通信工程
C401002	张国庆	计算机

(b) CT

图 5.15　2NF 关系

- 删除异常。例如,在上述的 SCT 关系模式中,当某教师暂时不担任讲课任务时,例如临时负责一段时间的实验室,该教师原来的讲课信息就要删除掉,显然其他信息也就跟着被删掉了,从而造成了删除异常,即不应删除的信息也被删掉了。
- 数据冗余,修改异常。例如,在上述的 SCT 关系模式中,当某教师同时担任多门课程时,他的姓名和所属教研室信息就要重复存储,造成大量的信息冗余。而且当教师的自身信息变化时,就要对数据库中所有相关的记录同时进行修改,造成修改的复杂化。如果漏掉一个记录还会给数据库造成信息的不一致。

所以保证数据库中各关系模式属于 2NF 是数据库逻辑设计中的最低要求。

在一个 1NF 关系模式转换成 2NF 关系模式后,可以在一定程度上减少原 1NF 关系中存在的插入异常、删除异常、数据冗余等问题,但并不能完全消除该关系中的所有异常和数据冗余。于是需要进一步引入第三范式。

5.6.4　第三范式(3NF)

1. 传递依赖

定义 5.19　设有关系模式 R(U,F)和属性集 $U=\{A_1,A_2,\cdots,A_n\}$的子集 X、Y、Z。如果有 $X\to Y$,$Y\to Z$,$Z-Y\neq\phi$,$Z-X\neq\phi$ 和 $Y\nrightarrow X$,则称 Z 传递依赖于 X,记为 $X\xrightarrow{t}Z$。

在定义 5.19 中,$Z-Y\neq\phi$ 说明在属性子集 Z 中至少存在一个属性不在属性子集 Y 中,因而保证了 $Y\to Z$ 是非平凡依赖。同理,$Z-X\neq\phi$ 说明在属性子集 Z 中至少存在一个属性不在属性子集 X 中,因而保证了 $X\to Z$ 是非平凡依赖。如果同时存在 $X\to Y$,$Y\to X$,则有 $X\leftrightarrow Y$,而 $Y\nrightarrow X$ 保证了该定义中只成立 $X\to Y$ 而不成立 $X\leftrightarrow Y$。

2. 第三范式

定义 5.20　如果一个关系模式 R 属于第一范式,并且 R 的任何一个非主属性都不传递依赖于它的任何一个候选键,则称 R 为满足第三范式的关系模式,也称 $R\in 3NF$。

【例 5.18】　设有关系模式 SDR(S,I,D,M),其中 S 表示商店名,I 表示商品,D 表示商品部,M 表示商品部经理。并有函数依赖:SI→D,表示每一个商店的每一个商品至多由一个商品部经销;SD→M,表示每一个商店的每一个商品部只有一个经理。关系 SDR 的唯一主键为 SI。

如果设 X=SI,Y=SD,A=M。显然有 SI→SD,即 X→Y,和 Y→A。出现了非主属性 A(通过 Y)传递依赖于候选键 X,所以关系模式 SDR 不属于 3NF。但在关系 SDR 中,既不存在非主属性 D 对候选键 SI 的部分依赖(即,D 不依赖于 S,D 也不依赖于 I),也不存在非

主属性 M 对候选键 SI 的部分依赖(即,M 不依赖于 S,M 也不依赖于 I)。所以关系 SDR 属于 2NF。

【例 5.19】 在例 5.17 由关系模式 SCT 分解成的两个关系模式 SC 和 CT 中,SC 是第三范式,因为非主属性 GRADE 既不部分依赖于 S# 或 C#,也不传递依赖于{S#,C#}。但 CT 不是 3NF,因为在 CT 中有 C#→TNAME 和 TNAME→TRSECTION,所以有 C#→TRSECTION,即存在非主属性传递依赖于主键。事实上在关系模式 CT 中,当某个教师主讲多门课程时,该教师所在的教研室名就要重复存储多次,即存在信息冗余。如果把关系模式 CT 分解成 CT1(C#,TNAME)和 CT2(TNAME,TRSECTION),在 CT1 和 CT2 中都不会存在 C#→TRSECTION,所以 CT1 和 CT2 都是 3NF。

定理 5.6 一个 3NF 的关系模式一定是 2NF 的。

证明:用反证法。

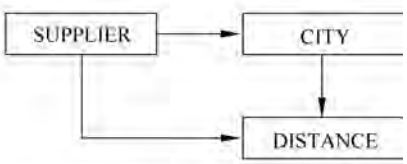
设 R 是 3NF 的,但不是 2NF 的,那么一定存在非主属性 A、候选键 X 和 X 的真子集 Y,使得 Y→A。由于 A 是非主属性,所以,A − X≠ϕ,A − Y≠ϕ。由于 Y 是候选键 X 的真子集,所以 X→Y,但 Y↛X。这样在 R 上存在着非主属性 A 传递依赖于候选键 X,所以 R 不是 3NF 的,这与假设矛盾,所以 R 也是 2NF 的。 证毕。

可以证明,如果一个关系模式 R 是 3NF,则它的每一个非主属性既不部分依赖于候选键,也不传递依赖于候选键。

【例 5.20】 图 5.16(a)给出了一个是第二范式而不是第三范式的例子。该关系的主键为 SUPPLIER,即供应商总部的名称,DISTANCE 属性的值是供应商总部到一个城市的距离。CITY 和 DISTANCE 都是非主属性,且都完全依赖于主键属性 SUPPLIER,所以该关系是第二范式。

SUPPLIER	CITY	DISTANCE
S1	西安	300
S2	西安	300
S3	上海	1050
S4	上海	1050

(a) 关系SUPPLIERS



(b) SUPPLIERS关系属性间的函数依赖

图 5.16 2NF 的规范化关系及其属性间的函数依赖

然而,如图 5.16(b)所示,由于在该关系中存在 SUPPLIER→CITY 和 CITY→DISTANCE,即非主属性传递依赖于主键,所以导致了供应商总部到城市的距离存储了不止一次的不良特性。

由图 5.16(b)可知,该关系中存在的非主属性 DISTANCE 对主键 SUPPLIER 的传递依赖,又可以看作是存在非主属性 DISTANCE 对非主属性 CITY 的函数依赖。所以,如果某关系存在非主属性对非主属性的函数依赖时,该关系即为第二范式。

SUPPLIERS 关系可以分解成两个第三范式的关系 SUPPLIERS1 和 DISTANCE,如图 5.17 所示。

SUPPLIER	CITY
S1	西安
S2	西安
S3	上海
S4	上海

(a) 关系SUPPLIERS1

CITY	DISTANCE
西安	300
上海	1050

(b) 关系DISTANCE

图 5.17　SUPPLIERS 关系的分解

5.6.5　鲍依斯-柯德范式

第三范式的关系消除了非主属性对主属性的部分依赖和传递依赖，解决了存储异常问题，基本上满足了实际应用的需求。但在实际中还可能存在主属性间的部分依赖和传递依赖，同样会出现存储异常。

例如，在关系模式 R(CITY，STREET，ZIP)中，R 的候选键为{CITY，STREET}和{ZIP，STREET}，R 上的函数依赖集为 F={{CITY，STREET}→ZIP，ZIP→CITY}。

由于 R 中没有非主属性，因而不存在非主属性对主属性的部分依赖和传递依赖，所以 R 是属于第三范式的。由于有 ZIP→CITY，当选取{ZIP，STREET}为主键时，主属性间存在着部分函数依赖，会引起更新异常等问题。因此，针对此类问题而提出了修正的第三范式，即鲍依斯-柯德(Boyce-Codd)范式，简称 BCNF 范式。

定义 5.21　设有关系模式 R(U，F)和属性集 U 的子集 X 和 A，且 A⊆X。如果对于 F 中的每一个函数依赖 X→A，X 都是 R 的一个候选键，则称 R 是鲍依斯-柯德范式，记为 BCNF。

定义 5.21 说明，如果 R 属于 BCNF，则 R 上的每一个函数依赖中的决定因素都是候选键。进一步讲，R 中的所有可能的非平凡依赖都是一个或多个属性对不包含它们的候选键的函数依赖。

对不是 BCNF 的关系模式，可通过模式分解使其成为 BCNF。例如，当把关系模式 R(CITY，STREET，ZIP)分解成 R_1(STREET，ZIP)和 R_2(ZIP，CITY)时，R_1 和 R_2 就都属于 BCNF 了。

定理 5.7　一个 BCNF 的关系模式一定是 3NF 的。

证明：用反证法。

设 R 是 BCNF 的，但不是 3NF，那么必定存在非主属性 A、候选键 X、属性集 Y，使得 X→Y，Y ↛ X，Y→A，A∉Y。但由于 R 是 BCNF，若有 Y→A 和 A∉Y，则必定有 Y 是 R 的候选键，因而应有 Y→X，这与假设 Y ↛ X 矛盾。　　证毕。

与定理 5.7 的结论不同，关系模式 R(CITY，STREET，ZIP)的例子说明，一个属于 3NF 的关系模式不一定属于 BCNF。

【例 5.21】　对于关系模式 SC(S#，C#，GRADE)，不存在非主属性对候选键{S#，C#}的部分依赖和传递依赖，所以 SC 属于 3NF。同时对于 SC 中的函数依赖{S#，C#}→GRADE，决定因素是主键，所以 SC 属于 BCNF。

5.6.6 范式之间的关系和关系模式的规范化

1. 范式之间的关系

对于前面介绍的 4 种范式，就范式的规范化程度来说，因为 BCNF 一定是 3NF，3NF 一定是 2NF，2NF 一定是 1NF，所以它们之间的关系满足 1NF⊇2NF⊇3NF⊇BCBF。就对函数依赖的要求（消除程度）来说，它们之间的关系如下。

第一范式（1NF）

↓　消除了非主属性对候选键的部分函数依赖

第二范式（2NF）

↓　消除了非主属性对候选键的传递函数依赖

第三范式（1NF）

↓　消除了主属性对候选键的部分函数依赖和传递函数依赖

鲍依斯-柯德（BCNF）

比较可知，一个数据库模式中的关系模式如果都是 BCNF，那么它就消除了整个关系模式中的存储异常，在函数依赖范畴内达到了最大程度的分解。3NF 的分解不彻底性表现在可能存在主属性对候选键的部分依赖和传递依赖。但在大多数情况下，一个关系模式中一般都既有主属性，也有非主属性，所以不会存在有主属性对主属性的部分依赖和传递依赖，所以数据库模式中的关系模式都达到 3NF 一般就足可以了。

2. 关系模式的规范化概念

为了把一个规范化程度较低（设为 x 范式）的关系模式转换成规范化程度较高（设为 x+1 范式）的关系模式，需要对规范化程度较低的关系模式进行分解。其分解过程要求满足保持原信息的无损和保持原来的函数依赖。这种通过模式分解使满足低一级范式的关系模式转换为满足高一级范式的关系模式的过程称为关系模式的规范化。

5.6.7 向 3NF 的保持无损和保持依赖分解算法

对于任何一个关系模式，都可以将其分解成 3NF。并且已有理论证明，将一个关系模式分解成 3NF 时既可保持函数依赖又具有无损连接性。

1. 满足 3NF 的保持依赖分解

算法 5.5　一个关系模式向 3NF 的保持依赖性分解算法。

输入：关系模式 R(U,F)及其函数依赖集 F。不失一般性，假设 F 已经是最小依赖集。

输出：R 的一个保持依赖的分解 $\rho=\{R_1,R_2,\cdots,R_k\}$，每个 R_i 为 3NF($i=1,2,\cdots,k$)。

方法：

(1) 若有函数依赖 $X\to A\in F$，且 $XA=R$，则 $\rho=\{R\}$，转(5)。

(2) 找出 R 的不在 F 中出现的所有属性，并把这些属性构成一个关系模式（某个 R 中如果有这样的不出现在 F 中的属性，则它们即是分解出的第一个子关系模式 R_1，虽然这些属性不出现在函数依赖中，但可以由它们的全部属性构成关系模式 R_1 的主键。在一个多对多的关系模式中就可能有这种情况）。然后把这些属性从 U 中去掉，将剩余的属性仍记为 U。

(3) 对 F 中的函数依赖按具有相同左部的原则进行分组，并按合并规则将每一组合并成一个新的函数依赖。例如若有 $X\to A_1$，$X\to A_2$，…，$X\to A_m$，则可以将它们合并成 $X\to$

$A_1A_2\cdots A_m$。

(4) 对于按上述过程得到的F中的每一个X→Y,均构成一个关系模式R_i=XY。

(5) 停止分解,输出ρ。

【例5.22】 设有关系模式R_1(ABCDEH),最小函数依赖集F={A→B,CD→A,BC→D,AE→H,CE→D}。

按照算法5.5,生成的关系模式为ρ={AB,ACD,BCD,AEH,CDE},且ρ中的每一个关系模式都是3NF的。

定理5.8 算法5.5给出的分解是满足第三范式的保持函数依赖的分解。

2. 向3NF的保持无损和保持依赖分解算法

算法5.6 一个关系模式向3NF的保持无损和保持依赖分解算法。

输入:关系模式R(U,F),已经是最小函数依赖集的F。

输出:R的一个既保持无损又保持依赖的分解ρ={R_1,R_2,…,R_k},且每个R_i为3NF(i=1,2,…,k)。

方法:

(1) 若有函数依赖X→A∈F,且XA=R,则ρ={R},转(6)。

(2) 如果R中存在N属性,则把从R中找出的所有N属性构成一个子关系模式R_1,然后把这些N属性从U中去掉,将剩余的属性仍记为U。

(3) 对F中的函数依赖按具有相同左部的原则进行分组,并按合并规则将每一组合并成一个新的函数依赖。例如若有X→A_1,X→A_2,…,X→A_m,则可以将它们合并成X→$A_1A_2\cdots A_m$。

(4) 将按上述过程得到的F中的每一个X→Y,均构成一个关系模式R_i=XY。

(5) 按算法5.2或定理5.5判断分解ρ={R_1,R_2,…,R_k}是否保持无损性,如果保持无损,转(6);如果不保持无损,设X是R的一个候选键,有ρ={R_1,R_2,…,R_k,X}。

(6) 输出分解ρ。

算法5.6的第(5)步中最后的"如果不保持无损,设X是R的一个候选键,有ρ={R_1,R_2,…,R_k,X}"的依据是下面的定理5.9。

定理5.9 设σ={R_1,R_2,…,R_k}是由算法5.5得到的R的3NF分解,X是R的一个候选键,则τ={R_1,R_2,…,R_k,X}也是R的一个分解。分解τ中的所有关系模式是3NF的,且分解τ保持依赖和具有无损连接性。

鉴于篇幅所限,定理5.8和定理5.9的证明从略,详细的证明过程见参考文献[1]。

【例5.23】 图5.2给出了大学教学信息管理数据库应用系统中的7个具有函数依赖约束的关系模式。由于其中的每个关系的函数依赖集中都只有一个函数依赖,根据算法5.6,这7个关系都满足3NF,且保持无损和保持依赖。

最后还需要说明的是,Beeri和Bernstein在1979年已经证明,仅确定一个关系模式是否是鲍依斯-柯德范式就是一个NP完全性问题(一个问题的NP完全性几乎肯定地隐含了它的计算时间是指数级的),所以目前还很难找到比较好的像鲍依斯-柯德范式的无损连接分解和保持依赖分解的算法。再加上实际中存在主属性间的部分依赖和传递依赖的情况也比较少见,所以在目前的数据库模式设计中,只考虑向3NF的保持无损连接和保持依赖分解就足够了。

5.7 关系模式的规范化方法小结

关系模型的规范化设计就是按照函数依赖理论和范式理论，对逻辑结构设计中的第一步所设计的关系模型进行规范化设计，基本设计方法可归纳如下。

(1) 根据每个关系模式的内涵，从语义的角度，分别确定每个关系模式中各个属性之间的数据依赖，进而确定每个关系模式的函数依赖集。

(2) 求每个关系模式的函数依赖集的最小依赖集，即按照函数依赖理论中的最小依赖集的求法：使每个关系模式的函数依赖集中没有多余依赖；每个依赖的左端没有多余属性；每个依赖的右端只有一个属性。

(3) 将求得的每个关系模式的函数依赖集中的决定因素相同的函数依赖进行合并。例如，如果求得的最小依赖集为 G={X→A,X→B,X→C,YZ→D,YZ→E}，那么将其决定因素相同的函数依赖合并后的结果为 G={X→ABC,YZ→DE}。

(4) 确定每个关系模式的候选键。

(5) 分析每个关系模式中存在的非主属性对候选键的部分依赖性和传递依赖性，确定其范式级别。

(6) 对不满足三范式的关系模式，按照关系模式分解理论和函数依赖理论，对每个关系模式及与之相关的函数依赖进行既保持无损连接性又保持函数依赖性的模式分解，直至所有关系模式都满足三范式。

(7) 通过以上的模式分解过程后，可能出现某些完全相同的关系模式，这一步就是要将完全相同的几个关系模式合并成一个单独的关系模式，即消除掉多余的关系模式。

习 题 5

单选题

5-1 解释下列术语

(1) 完全依赖　　(2) 部分依赖

(3) 函数依赖的逻辑蕴涵　　(4) 属性集 X 关于 F 的闭包 X^+

(5) 最小依赖集　　(6) 1NF

(7) 2NF　　(8) 3NF

(9) 无损连接分解　　(10) 保持函数依赖的分解

5-2 设有关系模式 R(A,B,C,D,E,P)，R 的函数依赖集为 F={A→D,E→C,AB→E,CD→H}，X=AE，求 X 关于 F 的闭包 X^+。

5-3 设有关系模式 R(A,B,C,D,E)，R 的函数依赖集为 F={AB→D,B→CD,DE→B,C→D,D→A}，计算$(AB)^+$，$(AC)^+$，$(DE)^+$。

5-4 证明函数依赖集 F={A→BC,A→D,CD→E}和函数依赖集 G={A→BCE,A→ABD,CD→E}的等价性。

5-5 设有关系模式 R(A,B,C,D,E)，R 的函数依赖集为 F={AB→D,B→CD,DE→B,C→D,D→A}，求 F 的最小依赖集。

5-6 设有关系模式R(A,B,C,D,E,P),R的函数依赖集为F={A→C,AB→C,C→DP,CE→AB,CD→P,EP→C},求F的最小依赖集。

5-7 设有关系模式R(A,B,C,D),R的函数依赖集为F={A→C,C→A,B→AC,D→AC,BD→A},求F的最小依赖集。

5-8 设有关系模式R(A,B,C,D,E),R的函数依赖集为F={A→D,E→D,D→B,BC→D,DC→A},求R的所有候选键。

5-9 设有关系模式R(A,B,C,D,E),R的函数依赖集为F={AB→C,C→D,D→E},求R的所有候选键。

5-10 设有关系模式R(A,B,C,D,E),R的函数依赖集为F={A→D,E→D,D→B,BC→D,DC→A},判断分解ρ={AB,AE,EC,DBC,AC}是否为无损连接分解。

5-11 设有关系模式R(A,B,C,D,E,P),R的函数依赖集为F={A→B,C→P,E→A,CE→D},判断分解ρ={R1(CP),R2(BE),R3(ECD),R4(AB)}是否为无损连接分解。

5-12 设有关系模式R(A,B,C),R的函数依赖集为F={A→B,B→C },并有分解ρ1={R1(AB),R2(AC)},ρ2={R1(AB),R3(BC)},ρ3={R2(AC),R3(BC)}。判断分解ρ1、ρ2、ρ3是否为无损连接分解。

5-13 设有关系模式R(A,B,C),R的函数依赖集为F={AB→C,C→A },并有分解ρ={R1(BC),R2(AC)}。判断分解ρ是否具有无损连接性?是否具有保持依赖性。

5-14 设有关系模式R(A,B,C),R的函数依赖集为F={A→B,B→C },并有分解ρ1={R1(AB),R2(AC)},ρ2={R3(AB),R4(BC)},ρ3={R5(AC),R6(BC)}。判断分解ρ1、ρ2、ρ3是否保持依赖性。

5-15 设有关系模式R(A,B,C,D),R的函数依赖集为F={A→B,B→C,C→D,D→A},判断分解ρ={R1(AB),R2(BC),R3(CD)}是否具有保持依赖性。

5-16 设有关系模式R(A,B,C,D,E,P),R的函数依赖集为F={C→P,EC→D,E→A,A→B},当把R分解成{R1(CP),R2(AE),R3(CDE),R5(BCE)}时,判断该分解是否保持依赖性。

5-17 设有关系模式R(A,B,C,D),其函数依赖集为F={D→A,C→D,B→C},请判断R能达到第几范式。

5-18 设有关系模式R(A,B,C,D),其函数依赖集为F={B→D,AB→C},请判断R能达到第几范式。

5-19 设有关系模式R(A,B,C,D,E,P),R的函数依赖集为F={A→B,C→P,E→A,CE→D},并有分解ρ={R1(ABE),R2(CDEP)},判断R1和R2分别为哪几范式。

5-20 设有关系模式R(A,B,C,D,E,P),R的函数依赖集为F={C→P,EC→D,E→A,A→B},把R分解为3NF并具有无损连接性和保持依赖性。

5-21 下面结论哪些是正确的?哪些是错误的?如果是错误的,请举一个反例说明。

(1) 任何一个二元关系模式都属于3NF。

(2) 任何一个二元关系模式都属于BCNF。

(3) 关系模式R(A,B,C)中如果有A→B,B→C,则有A→C。

(4) 关系模式R(A,B,C)中如果有A→B,A→C,则有A→BC。

(5) 关系模式 R(A,B,C)中如果有 B→A,C→A,则有 BC→A。

(6) 关系模式 R(A,B,C)中如果有 BC→A,则有 B→A 和 C→A。

5-22 证明：一个属于 3NF 的关系模式也一定属于 2NF。

5-23 证明：在关系模式中,如果关系的候选键由关系模式中的全部属性组成,则该关系一定是 3NF。