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，在此基础上引出函数依赖的概念和函数依赖的公理体系，并给出函数依赖集的分解方法，接着给出保持无损分解的概念，并给出保持无损性的分解方法，最后给出关系模式的规范化方法。

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，得到课程（归属）表、学生成绩表、授课（统计）表和教师信息表。将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	学生成绩表	授课（统计）表	教师信息表
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	教师出生日期
		分数	教师工号	职称
		专业名称	教师姓名	（所在）教研室
			职称	

如果直接将图1.8(a)和表5.1中4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，就可以得到如下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）课程关系（课程号，课程名，学时，教研室）。

(3) 学生成绩关系(学号,姓名,课程号,课程名,分数,专业名称)。

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

(5) 教师关系(教师工号,教师姓名,教师性别,教师出生日期,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U表示关系模式的属性集合);每个属性的值域(设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值域集合的映射或约束(设DOM表示属性集U到值域集合D的映射)。

同时,在每个关系表中都存在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,在课程关系中,给定一个"课程号"的值,就可以确定开设该课程的教研室的名称;在教师关系中,给定一个"教职工号"的值,就可以确定该教师所在的教研室的名称。由于这种依赖是用属性的值体现的,所以称为数据依赖;当用属性名集合表示其依赖关系时,就可以看作一种函数依赖(设F表示函数依赖集合)。

综上,当把所有这些要素完整地反映到关系模式的描述中时,就可以得到如下结论。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。

需要进一步说明的是,在本章后续的内容中会涉及较多且比较严格的定义、定律和公式推导,为了表述上的方便,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这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这3种函数依赖表示方法是等价的。

5.2 对关系模式规范化设计的必要性

本节先介绍关系模式规范化设计的必要性,然后介绍由此引出的关系模式分解的相关问题及解决思路。

5.2.1 对关系模式进行规范化设计的必要性

本节用一个由于构建的关系模式不合理而引起操作异常的例子说明对关系模式进行规范化设计的必要性。

对于 5.1 节的授课关系模式(调整了其中一些属性的顺序):

授课关系(课程号,课程名,学时,教研室,教师工号,教师姓名,职称)

可得用符号形式表示的授课统计关系模式为

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

则这个关系模式在使用过程中会存在以下操作异常问题。

1) 数据冗余

对于每位教师所讲的每一门课,教师姓名、职称、教研室等信息都要重复存放,会造成大量的数据冗余。

2) 更新异常

由于有数据冗余,如果某教师的职称或教研室变化,就必须对所有具有教师职称或所属教研室的元组进行修改。这不仅增加了更新的代价,而且可能出现一部分元组修改了,另一部分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,如果某一位教师刚调来,或某一位教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)而无法将该教师的教师姓名、职称、教研室等基本信息输入数据库中。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,显然是不符合实际情况的。

4) 删除异常

如果某教师不再上课,则在删除该教师所担任的课程信息的同时,就会连同该教师的教师姓名、职称、教研室等基本信息都删除了。

以上表明,上述 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以确定,而与所上课程的课程号无关。

如果将上述关系模式分解为以下两个关系模式:

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。

5.2.2 关系模式分解的思路

进一步分析授课关系 TEACHES 可知,其属性集 U 可表示为

U = {T#,TNAME,TITLEOF,TRSECTION,C#,CNAME,CLASSH}

如果给定一个教师工号 T# 的值,就能唯一地确定出该教师的姓名 TNAME、职称 TITLEOF 和所在教研室 TRSECTION 的值。例如以图 1.8 的关系当前值为例,当给出

T#的值为T0401001时,就能得到该教师工号对应的TNAME、TITLEOF和TRSECTION的值为{张国庆,教授,计算机}。这就是说,在该关系模式中存在着{T#}决定{TNAME,TITLEOF,TRSECTION}的函数依赖关系,即

{T#}→{TNAME,TITLEOF,TRSECTION}

同理,也存在着下述两个函数依赖:

{C#}→{CNAME,CLASSH}
{T#,C#}→{TNAME,TITLEOF,TRSECTION,CNAME,CLASSH}

综上分析可知,对于关系模式TEACHES(U,F)来说,有

U={T#,TNAME,TITLEOF,TRSECTION,C#,CNAME,CLASSH}
F={{T#,C#}→{TNAME,TITLEOF,TRSECTION,CNAME,CLASSH},
{T#}→{TNAME,TITLEOF,TRSECTION},{C#}→{CNAME,CLASSH}}

显然,当把关系模式TEACHES(U,F)分解成

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

时,可以对应地将它们表示为TEACHER(U1,F1)和COURSE(U2,F2),且有

U1={T#,TNAME,TITLEOF,TRSECTION}
F1={{T#}→{TNAME,TITLEOF,TRSECTION}}
U2={C#,CNAME,CLASSH}
F2={{C#}→{CNAME,CLASSH}}

5.2.3 关系模式分解的定义

将上述的关系模式分解思路可概念化地抽象成如下关系模式分解定义。

定义5.1 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($1\leqslant i,j\leqslant k$),$U_i\subseteq U_j$不成立,且$F_i$中的每个函数依赖X→Y的决定因素X和被决定因素Y中的属性都在U_i中(即F_i是F在U_i上的投影,详见5.3.6节),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5.1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$1\leqslant j\leqslant k$,$U_i\subseteq U_j$既不成立,$U_j\subseteq U_i$也不成立。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。

如何将F分解成F_1和F_2,即是5.3节将要介绍的函数依赖集分解方法;如何将U分解成U_1和U_2,即是5.4节将要介绍的关系模式(属性集)分解方法。

5.3 函数依赖集分解方法

函数依赖(functional dependency)是关系所表述的信息本身所具有的特性,换而言之,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,而仅取决于两个元组的某些分量是否相等。正是这种限制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 5.2 设有关系模式 R(A_1,A_2,…,A_n)和属性集 U={A_1,A_2,…,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:

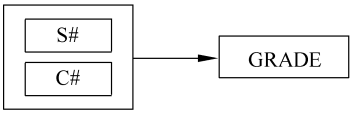
S(S#,SNAME,SSEX,SBIRTHIN,PLACEOFB,SCODE#,CLASS)

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(即关系 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”,出生日期为“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,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。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,…,A_n)和函数依赖 X→Y 来说,属性子集 X 中包括且仅包括关系 R 的主属性,对于关系 R 的任何属性子集 Y,X→Y 一定成立。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。

(2) 若 X→Y,且 Y⊆X,则称 X→Y 为平凡函数依赖。

若不特别声明,总假定本章所讨论的是非平凡函数依赖。

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

对学习关系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},{{S#,C#}→{GRADE}})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,这就是函数依赖的逻辑蕴涵要讨论的问题。

定义5.3 设有关系模式R(U,F),X、Y是属性集U={A_1,A_2,…,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

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

5.3.4 函数依赖的公理体系

由前述内容可知，对于关系模式 R(U,F)来说，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，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，这就形成了著名的 Armstrong(阿姆斯特朗)公理体系。Armstrong 公理体系包括公理和规则两部分。

1. 阿姆斯特朗公理

定义 5.4 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、W，阿姆斯特朗公理包括如下内容。

- 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 5.2，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 5.2，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 5.2，可得 X→Z。证毕。

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,根据增广律可得X→XY。

由条件X→Z,根据增广律可得XY→YZ。

根据传递律,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,根据自反律可得Y→Z。

根据传递律,由X→Y和Y→Z,可得X→Z。

3) 证明推论5.3

由条件X→Y,根据增广律可得XW→WY。

根据传递律和已知条件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{CITY,STREET}→{CITY,STREET,ZIP}和{STREET,ZIP}→{CITY,STREET,ZIP}成立。现证明后者。

证明：

已知有ZIP→CITY,由增广律可得

$$\{STREET,ZIP\}\rightarrow\{CITY,STREET\}。$$

又已知{CITY,STREET}→ZIP,由增广律可得

$$\{CITY,STREET\}\rightarrow\{CITY,STREET,ZIP\}。$$

由上述所得的两个结论,并根据传递律即可得到

$$\{STREET,ZIP\}\rightarrow\{CITY,STREET,ZIP\}。证毕。$$

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

由合并规则和分解规则即可得到这个定理,其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关系的主键的确定奠定了理论基础。

5.3.5 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,人们经过研究提出了一种使用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^+的概念

定义5.5 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,则称所有用阿姆斯特朗公理从F推出的函数依赖$X\rightarrow A_i$的属性A_i组成的集合为X关于F的闭包,记为X_F^+,通常简记为X^+,即

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

显然有$X\subseteq X^+$。

2. 使用X关于F的闭包定义计算X^+

【例5.3】 已知在关系模式R(A,B,C)上有函数依赖F={A→B,B→C},求X分别等

于 A、B、C 时的 X^+。

解：

(1) 对于 X=A，根据自反律有 A→A；

已知有 A→B，且由已知的 A→B，B→C，根据传递率可推得 A→C，所以有 X^+=ABC。

(2) 对于 X=B：

根据自反律有 B→B；且已知 B→C，所以有 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。

3. 使用 X^+ 判断某一函数依赖是否能从 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⊆X^+。

证明：

(1) 充分性证明：设 Y=$A_1,A_2,\cdots,A_n$，A_i⊆U(i=1,2,…,n)。假设 Y⊆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。再使用合并规则(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⊆U(i=1,2,…,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→A_1，X→A_2，…，X→A_n，由 X^+ 的定义可知，A_i⊆X^+(i=1,2,…,n)，所以 Y⊆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，转换成 X^+ 是否包含 Y 的问题：当 X^+ 包含 Y 时，说明 X→Y 能从 F 根据阿姆斯特朗公理导出。

4. 计算 X 关于 F 的闭包 X^+ 的算法

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) 如果 F 中的所有函数依赖的左端属性都不被 $X^{(i)}$ 包含，或 F 为空集，转(4)。

(3) 否则，依次考察 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→W，若有 V⊆$X^{(i)}$，则有 $X^{(i+1)}$=$X^{(i)}$ W；并得到新的 F=F−{V→W}。接下来转(2)。

(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→C，BC→D，ACD→B，D→EH，BE→C}，属性集 U={A，B，C，D，E，H}，X=BD，求 X^+。

解：

(1) $X^{(0)}$=BD。

(2) 依次考察 F 中的函数依赖，由于 $X^{(0)}$ 包含 D→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→EH}得到新的 F={AB→

C,BC→D,ACD→B,BE→C}。

(3) 重新依次考察F中的函数依赖,由于$X^{(1)}$包含BE→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,并得到新的F={AB→C,BC→D,ACD→B}。

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

(5) 进一步依次考察F中的函数依赖,发现F中剩余的函数依赖AB→C和ACD→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。

5.3.6 函数依赖集的分解方法

下面介绍函数依赖集分解过程中用到的函数依赖集的投影概念。

1. F在U_i上的投影

根据关系模式为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,由于这种划分是根据分解成的各子关系模式的属性来进行的,所以涉及F在各子关系模式的属性集U_i上的投影概念。

定义5.6 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5.6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,也要将依赖集F分解成k个依赖子集F_i且$1\leqslant i\leqslant k$。定义5.6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5.6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5.7 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→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5.7,显然有$\pi_Z(F)=\{X\rightarrow Y \mid X\rightarrow Y\in F^{+}$,且$XY\subseteq Z\}$,所以定义5.7和定义5.6本质上是相同的。

2. 保持依赖的分解

在关系模式的分解中有一个重要的性质,就是要求分解后的子模式还应保持原关系模式的函数依赖,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5.8 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$\pi_{R_i}(F)(i=1,2,\cdots,k)$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。

基于定义5.8,可得保持函数依赖分解的判定方法。

(1) 对ρ中的每一个R_i求$F_i=\pi_{R_i}(F)$。

(2) 求 $\bigcup_{i=1}^{k}\pi_{R_i}(F)$。

(3) 判断 F 中每一个函数依赖 X→Y 是否能从 $\bigcup_{i=1}^{k}\pi_{R_i}(F)$ 推出，如果均能推出，则分解 ρ 保持函数依赖，即具有函数依赖性；否则分解 ρ 不保持函数依赖，即不具有函数依赖性。

【例 5.5】 已知关系模式 R(U,F)，U={S#,SD,SM}，F={S#→SD,SD→SM}。通过对关系 R 的 3 种不同分解方法，根据定义 5.8 判断各分解方法是否为保持依赖的分解。

解：

(1) 方法 1。

设关系模式 R(U,F)被分解成 $\rho_1=\{R_1(S\#,\phi),R_2(SD,\phi),R_3(SM,\phi)\}$。

对 F 在各 R_i 上进行投影，有 $F_i=\pi_{R_i}(F)=\phi$（ϕ 表示空集），且有

$$F_1\cup F_2\cup F_3=\phi\cup\phi\cup\phi=\phi\neq F$$

所以分解 ρ_1 不保持原关系 R 的函数依赖性，即该分解不保持依赖。

(2) 方法 2。

设关系模式 R(U,F)被分解成 $\rho_2=\{R_1(\{S\#,SD\},\{S\#\rightarrow SD\}),R_2(\{S\#,SM\},\{S\#\rightarrow SM\})\}$。

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，对于 F 中的 SD→SM，有 $(SD\rightarrow SM)\notin\{S\#\rightarrow SD,S\#\rightarrow SM\}$

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 不保持函数依赖。

(3) 方法 3。

设关系模式 R(U,F)被分解成 $\rho_3=\{R_1(\{S\#,SD\},\{S\#\rightarrow SD\}),R_2(\{SD,SM\},\{SD\rightarrow SM\})\}$。

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

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

$$=F$$

所以分解 ρ_3 保持原关系 R 的函数依赖性。

5.4 关系模式的分解方法

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，经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，下面讨论关系模式分解中的相关问题。

5.4.1 保持无损分解的概念

定义 5.9 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，$r=\pi_{R_1}(r)\bowtie\pi_{R_2}(r)\bowtie\cdots\bowtie\pi_{R_k}(r)$成立，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作自然连接时,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,则为笛卡儿积运算。

【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。通过对关系R的3种不同分解方法,根据定义5.9判断各分解方法是否为无损连接分解。

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:

(1) 方法1。

设关系模式R(U,F)被分解成 $\rho_1=\{R_1(S\#,\phi),R_2(SD,\phi),R_3(SM,\phi)\}$。

对图5.3的已知具体关系r在 U_i 上进行投影,可得 $R_i=R(U_i)$ 的各 r_i 的当前值如图5.4所示。

S#
S1
S2
S3

(a) R_1的当前值r_1

SD
D1
D2
D3

(b) R_2的当前值r_2

SM
M1
M2

(c) R_3的当前值r_3

图5.4 R_i 的当前值 r_i

根据定义5.9,计算 $r=r_1 \bowtie r_2 \bowtie r_3=r_1\times r_2\times r_3$ 可得(在各 r_i 没有相同属性的情况下,其自然连接运算即是笛卡儿积运算):

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

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许多)。所以分解 ρ_1 是有损原来信息的一种分解,或者说分解 ρ_1 不保持信息无损,不具有无损连接性。

(2) 方法2。

设关系模式R(U,F)被分解成 $\rho_2=\{R_1(\{S\#,SD\},\{S\#\rightarrow SD\}),R_2(\{S\#,SM\},\{S\#\rightarrow SM\})\}$。

对图5.3的已知具体关系r在 U_i 上投影,可得 $R_i=R(U_i)$ 的各 r_i 的当前值如图5.5所示。

计算 $r=r_1 \bowtie r_2$ 可得其结果与图5.3相同,说明分解 ρ_2 是保持无损连接的分解。

S#	SD
S1	D1
S2	D2
S3	D3

(a) R_1的当前值r_1

S#	SM
S1	M1
S2	M1
S3	M2

(b) R_2的当前值r_2

图 5.5　R_i 的当前值 r_i

(3) 方法 3。

设关系模式 R(U,F)被分解成 $\rho_3=\{R_1(\{S\#,SD\},\{S\#\rightarrow SD\}),R_2(\{SD,SM\},\{SD\rightarrow SM\})\}$。

对图 5.3 的已知具体关系 r 在 U_i 上投影，可得 $R_i=R(U_i)$的各 r_i 的当前值如图 5.6 所示。

S#	SD
S1	D1
S2	D2
S3	D3

(a) R_1的当前值r_1

SD	SM
D1	M1
D2	M1
D3	M2

(b) R_2的当前值r_2

图 5.6　R_i 的当前值 r_i

计算 $r=r_1 \bowtie r_2$ 可得其结果与图 5.3 相同，说明分解 ρ_3 是保持无损连接的分解。

在例 5.6 中，由于元组个数比较少，根据定义 5.9 和分解后的当前关系判断分解后的各关系是否为无损分解还是比较方便的；但是当一个关系的当前值具有成千上万个元组时，判断其分解后的若干子关系是否为无损连接分解就比较麻烦。为此，引入下面的判断保持无损连接分解的方法。

5.4.2　分解成两个以上子关系模式保持无损的判别方法

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 列的表，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 列位置的元素的填入方法为：如果 A_j 在 R_i 中，则在第 i 行第 j 列的位置填符号 a_j，否则填符号 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 Y 属性上各行值的方法如下。

① 只需这些行的 Y 属性上的某一行的值为 a_j(j 为 Y 属性对应的那些列的列序号，且 $j=1,2,\cdots,n$)时，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 属性列中的值没有 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)逐个考察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ρ={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列的表,并按算法5.2的(1)填写表中的元素,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_{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5.7 例5.7第(1)步的结果

(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,将该列第2、5行位置的元素修改成b_{13},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}	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5.8 例5.7第(2)步的结果

(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,将该列第3行位置的元素修改成b_{13},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}	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_{54}	a_5

图5.9 例5.7第(3)步的结果

(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,所以将该列第2、3、5行位置的元素修改成a_4,其结果如图5.10所示。

(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,所以将该列第3、5行位置的元素修改成a_3,其结果如图5.11所示。

(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,所以将该列第3、4行位置的元素修改成a_1,其结果如图5.12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 5.10　例 5.7 第(4)步的结果

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 5.11　例 5.7 第(5)步的结果

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 5.12　例 5.7 第(6)步的结果

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。

算法 5.2 适用于关系模式 R 被分解成两个以上子关系模式的情况。如果只将关系模式 R 分解为两个关系模式，可以用定理 5.4 给出的检验方法进行检验。

5.4.3　分解成两个子关系模式保持无损的判别方法

定理 5.4　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 $(R_1\cap R_2)\rightarrow(R_1-R_2)\in F^+$ 或 $(R_1\cap R_2)\rightarrow(R_2-R_1)\in F^+$ 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，并用算法 5.2 构造如图 5.13 所示的 2 行 k 列的表。

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 5.13　用算法 5.2 构造 2 行 k 列的表

(1) 充分性证明：假设 $(R_1\cap R_2)\rightarrow(R_1-R_2)$ 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 $(R_1\cap R_2)\rightarrow(R_1-R_2)$ 不在 F 中，但在 F^+ 中，则可用公理从 F 中推出 $(R_1\cap R_2)\rightarrow A_y$，其中 $A_y\in(R_1-R_2)$，即 A_y 是 R_1-R_2 中的任一属性。所以用算法 5.2 可以将属性 A_y 列所对应的第 2 行中的 b_{2y} 改为 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 $(R_1\cap R_2)\rightarrow(R_2-R_1)$，可类似地证得表中的第 1 行为 $a_1\cdots a_k$。

(2) 必要性证明：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 $(R_1 \cap R_2) \rightarrow (R_1 - R_2)$ 成立；若第 1 行为 $a_1 \cdots a_k$，则意味着 $(R_1 \cap R_2) \rightarrow (R_2 - R_1)$ 成立。证毕。

上面的必要性证明也可以这样来理解：根据定义 5.9，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 的任一满足 F 的具体关系 r，必然有 $((R_1 \cap R_2) \rightarrow (R_1 - R_2)) \in F$，或者用公理可由 F 推出 $(R_1 \cap R_2) \rightarrow (R_1 - R_2)$。同理，有 $(R_1 \cap R_2) \rightarrow (R_2 - R_1)$ 的情况。

【例 5.8】 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：

$$\begin{aligned} \text{因为}(R_1 \cap R_2) \rightarrow (R_1 - R_2) &= (AB \cap BC) \rightarrow AB - BC \\ &= (B \rightarrow A) \notin F \\ (R_2 \cap R_1) \rightarrow (R_2 - R_1) &= (BC \cap AB) \rightarrow BC - AB \\ &= (B \rightarrow C) \notin F \end{aligned}$$

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 5.9】 在例 5.6 中，已知有关系模式 R(S#,SD,SM)，函数依赖集 $F=\{S\# \rightarrow SD, SD \rightarrow SM\}$。对于其中的方法(2)，已知有分解 $\rho_2=\{R_1(\{S\#, SD\}, \{S\# \rightarrow SD\}), R_2(\{S\#, SM\}, \{S\# \rightarrow SM\})\}$，且已求解知该分解是无损连接分解。用定理 5.4 的方法验证分解 ρ_2 具有无损连接性。

解：

$$\begin{aligned} \text{因为 } (R_1 \cap R_2) \rightarrow (R_1 - R_2) &= (\{S\#, SD\} \cap \{S\#, SM\}) \rightarrow (\{S\#, SD\} - \{S\#, SM\}) \\ &= S\# \rightarrow SD \in F \end{aligned}$$

所以分解 ρ_2 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 $F=\{S\# \rightarrow SD, SD \rightarrow SM\}$。对于其中的方法(3)，已知有分解 $\rho_3=\{R_1(S\#, SD), R_2(SD, SM)\}$，且已求解知该分解是无损连接分解。用定理 5.4 的方法验证分解 ρ_2 具有无损连接性。

解：

$$\begin{aligned} \text{虽然}(R_1 \cap R_2) \rightarrow (R_1 - R_2) &= (\{S\#, SD\} \cap \{SD, SM\}) \rightarrow (\{S\#, SD\} - \{SD, SM\}) \\ &= SD \rightarrow S\# \notin F^+ \end{aligned}$$

$$\begin{aligned} \text{但是}(R_2 \cap R_1) \rightarrow (R_2 - R_1) &= (\{SD, SM\} \cap \{S\#, SD\}) \rightarrow (\{SD, SM\} - \{S\#, SD\}) \\ &= SD \rightarrow SM \in F^+ \end{aligned}$$

所以分解 ρ_3 具有无损连接性。

这里要特别指出，定理 5.4 并不要求 $(R_1 \cap R_2) \rightarrow (R_1 - R_2) \in F^+$ 和 $(R_2 \cap R_1) \rightarrow (R_2 - R_1) \in F^+$ 同时成立，而是只要 $(R_1 \cap R_2) \rightarrow (R_1 - R_2) \in F^+$ 或 $(R_2 \cap R_1) \rightarrow (R_2 - R_1) \in F^+$ 之一成立即可。

例 5.5 和例 5.6 分别对同一关系 R 的 3 种不同分解方法进行了是否为保持依赖的分解和保持无损的分解的判断，其后的相关例子又对其进行了验证，结果是：方法 1 既不保持信息无损，也不保持函数依赖；方法 2 保持信息无损，但不保持函数依赖；方法 3 既保持信息

无损，又保持函数依赖。所以，对一个关系模式的保持无损和保持依赖的判断结果应该具有以下 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。

5.5 关系模式的规范化

在使用某种约束条件对关系模式进行规范化后，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。

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，再依次介绍各范式。

5.5.1 候选键的求解方法

1. 关系属性的分类

定义 5.10 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出现分为以下 4 类。

(1) L 类：仅在 F 中的函数依赖左端(Left)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(Right)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 5.11】 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 5.11 设有关系模式 S(U，F)和属性集 U＝{A_1，A_2，…，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 S 的某一候选键中。

(3) 若 X 是 L 类属性，则 X 必为 S 的某一候选键的成员。

(4) 若 X 是 L 类属性，且 X^+ 包含了 S 的全部属性，则 X 必为 S 的唯一候选键。

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

3. 多属性候选键的求解方法

算法 5.3 多属性候选键的判定算法。

输入：关系模式 $R(A_1,A_2,\cdots,A_n)$和 R 的函数依赖集 F。

输出：R 的所有候选键。

方法：

(1) 将关系模式 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^+ 包含了关系模式 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$(XA)_F^+$ 包含了关系模式 R 的全部属性，则 XA 为 R 的一个候选键。

(4) 重复(3)，直到 Y 中的属性依次取完为止。

(5) 从 Y 中去掉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$(XB)_F^+$ 包含了关系模式 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)的组合依次取完为止。

(8)输出候选键，算法结束。

【例 5.12】 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：根据算法 5.3

(1) 根据 F 对 R 的所有属性进行分类：A 为 L 类属性；B、C、D 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$(AB)^+=ABCDE$，$(AB)^+$包含了 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$(AC)^+=ABCDE$，$(AC)^+$包含了 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$(AD)^+=ABCDE$，$(AD)^+$包含了 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，不需要考察从 Y 中取 3 个属性的情况，候选键的求解到此结束。

综上可知，R 的候选键有 AB、AC 和 AD。

【例 5.13】 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：根据算法 5.3

(1) 根据 F 对 R 的所有属性进行分类：A、B、C、D、E 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，由于本例中没有 L 属性，所以仅需要计算从 Y 中取出的属性关于 F 的闭包。

$A^+=ABCDE$，包含了 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，包含了 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，包含了 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，包含了 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，不需要考察从 Y 中取 3 个属性的情况，候选键的求解到此结束。

综上可知，R 的候选键有 A、E、BC 和 CD。

5.5.2 第一范式(1NF)

定义 5.12 如果关系模式 R 中每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，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 5.14 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) S-PART属性有重复值

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) PHONE属性有空白值

图 5.14 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 5.14 的非规范关系对应的规范关系如图 5.15 所示。

5.5.3 第二范式(2NF)

1. 完全依赖

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、Y，如果 X→Y，并且对于 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 $X \xrightarrow{f} Y$。

例如，如果有 AB→C，且不存在 A→C 和 B→C 成立，则 C 完全依赖于 AB。

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

(a) S-PART属性的规范关系

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) PHONE属性的规范关系

图 5.15　图 5.14 的非规范关系转换成的规范关系

【例 5.14】 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#}$\nrightarrow${GRADE}、{C#}$\nrightarrow${GRADE}和{S#,C#}$\xrightarrow{f}${GRADE}。

2. 部分依赖

定义 5.14 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,但 Y 不完全依赖于 X,则称 Y 部分依赖于 X,记为 $X\xrightarrow{p}Y$。

例如,如果有 AB→C,且至少存在 A→C 或 B→C 之一成立,则 C 部分依赖于 AB。

比较定义 5.13 和定义 5.14 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 5.13 和定义 5.14 可知,当 X 是仅包含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。

3. 第二范式

定义 5.15 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),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 不属于 2NF。

【例 5.15】 对于图 5.16 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。

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 部分依赖于候选键{S#, C#},所以图 5.16 的关系 SCT 是 1NF 而不是 2NF。然而,当把关系模式 SCT 分解成如图 5.17 所示的两个关系 SC 和 CT 时,SC 和 CT 既是 1NF,又是 2NF。

注意:当一个关系模式不是 2NF 时会产生以下问题。

(1) 插入异常。例如,在上述的 SCT 关系模式中,当某一位新调来的教师还没有担任

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
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 5.16　一个关系模式 SCT 的具体关系

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 5.17　2NF 关系

讲课任务时，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，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无法插入。

(2) 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，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掉，显然其他信息也就随着被删掉，从而造成了删除异常，即不应该删除的信息也被删掉。

(3) 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，需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。

5.5.4　第三范式(3NF)

1. 传递依赖

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、Z，如果有 X→Y，Y→Z，Z−Y≠ϕ，Z−X≠ϕ 和 Y↛X，则称 Z 传递依赖于 X，记为 $X \xrightarrow{t} Z$。

在定义 5.16 中，Z−Y≠ϕ 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，因此保证了 Y→Z 是非平凡依赖。同理，Z−X≠ϕ 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，因此保证了 X→Z 是非平凡依赖。如果同时存在 X→Y，Y→X，则有 X↔Y，而 Y↛X 保证了该定义中只有 X→Y 成立，X↔Y 不成立。

2. 第三范式

定义 5.17 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∈3NF。

【例 5.16】 设有关系模式 SDR(S,I,D,M)和函数依赖集合 F={SI→D,SD→M},关系 SDR 的唯一主键为 SI,请分析 SDR 是第几范式。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。函数依赖 SI→D 表示每一个商店的每一个商品最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。

解:

如果设 X=SI,Y=SD,A=M。由 SI→D,可得 SI→SD,即 X→Y 和 Y→A 同时成立。这样就出现了非主属性(通过)传递依赖于候选键,所以关系模式 SDR 不属于 3NF。

但是在关系 SDR 中,既不存在非主属性 D 对候选键 SI 的部分依赖(D 完全依赖于 SI),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,也不依赖于 I),所以关系 SDR 属于 2NF。

【例 5.17】 分析例 5.15 中由关系模式 SCT 分解成的两个关系模式 SC(S#,C#,GRADE)和 CT(C#,TNANE,TRSECTION)是第几范式。

解:

在关系模式 SC 中,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#},所以 SC 是 3NF。

在关系模式 CT 中,有 C#→TNAME 和 TNAME→TRSECTION,所以可推知 C#→TRSECTION,即存在非主属性传递依赖于主键,所以 CT 不是 3NF。由于不存在非主属性对候选键的部分依赖,所以 CT 是 2NF。事实上,当某个教师主讲多门课程时,该教师所在的教研室信息要重复存储多次,即存在信息冗余。进一步,如果再把关系模式 CT 分解成 CT1(C#,TNAME)和 CT2(TNAME,TRSECTION),CT1 和 CT2 就都是 3NF。

定理 5.5 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。

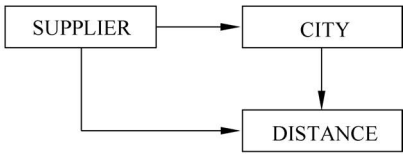
【例 5.18】 图 5.18(a)给出了一个是第二范式而不是第三范式的例子。该关系的主键为 SUPPLIER,即供应商总部的名称,DISTANCE 属性的值是供应商总部到一个城市的距离。CITY 和 DISTANCE 都是非主属性,且都完全依赖于主键属性 SUPPLIER,所以该关系是第二范式。

然而,如图 5.18(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 5.18(b)可知,该关系中存在的非主属性 DISTANCE 对主键 SUPPLIER 的传递

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

(a) SUPPLIERS关系



(b) SUPPLIERS关系属性间的函数依赖

图 5.18　2NF 的规范化关系及其属性间的函数依赖

依赖，又可以看作存在非主属性 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，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 5.19 所示。

SUPPLIER	CITY
S1	西安
S2	西安
S3	上海
S4	上海

(a) SUPPLIERS1关系

CITY	DISTANCE
西安	300
上海	1050

(b) DISTANCE关系

图 5.19　SUPPLIERS 关系的分解

【例 5.19】 假设关系模式 R 的候选键都是单属性，请判断该关系模式的最高范式能达到第几范式。

解：因为 R 的候选键都是单属性，所以一定不会存在非主属性对候选键的部分依赖，所以 R 满足 2NF。本例提供的已知条件还无法确认函数依赖集 F 中是否存在非主属性对候选键的传递依赖，所以该关系模式的最高范式只能达到 2NF。

【例 5.20】 全键属性的关系属于哪几范式？为什么？

解：全键属性的关系同时属于 1NF、2NF、3NF。

因为是全键关系，所以该关系中的属性都是主属性而不存在非主属性，也就不存在非主属性对键的部分函数依赖，所以属于 2NF；且不存在非主属性对键的传递函数依赖，所以全键关系也属于 3NF。属于 2NF 和 3NF 的关系自然也属于 1NF。

【例 5.21】 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，显然不存在非主属性对候选键（主键）的部分依赖，也不存在非主属性对候选键（主键）的传递依赖，所以这 7 个关系都满足 3NF，且保持无损和保持依赖。

5.5.5　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。但是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}。

R中没有非主属性,因此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5.18 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5.18说明,如果R属于BCNF,则R中的每一个函数依赖的决定因素都是候选键。进一步讲,R中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。

对于不是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都属于BCNF。

定理5.6 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的,若有Y→A和A∉Y,则必定有Y是R的候选键,因此应有Y→X,这与假设Y↛X矛盾。证毕。

与定理5.6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。例如,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。但是由于关系模式SC的函数依赖{S#,C#}→GRADE的被决定因素是非主属性,所以SC不属于BCNF。

5.5.6 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⊇BCNF;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(3NF)

↓ 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)

通过比较可知,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。但是在大多数情况下,一个关系模式中既有主属性,又有非主属性,所以不会存在主属性对主属性的部分依赖和传递依赖,因此

数据库模式中的关系模式都达到 3NF 一般就可以了。

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。

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为鲍依斯-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比较少见,所以在目前的数据库模式设计中只考虑像 3NF 的保持无损连接和保持依赖分解就足够了。

5.6 小　　结

关系模式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第一步所设计的关系模式进行规范化设计,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,直到所有关系模式都满足三范式为止。

(7) 通过以上模式分解过程后,可能会出现某些完全相同的关系模式,这一步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,即去掉多余的关系模式。

习　题　5

扫一扫 作业　扫一扫 自测题

5-1 解释下列术语。

(1) 非平凡依赖　　(2) 函数依赖的逻辑蕴含

(3) 部分依赖　　(4) 完全依赖

(5) 无损连接分解　　(6) 保持函数依赖的分解

(7) 2NF　　(8) 3NF

5-2 设有关系模式 R(A,B,C,D,E,H),R 的函数依赖集为 F={A→D,E→C,AB→E,CD→H}。

(1) 当 X=AE 时,求 X 关于 F 的闭包 X^+。

(2) 当 X=AB 时,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={A→D,E→D,D→B,BC→D,DC→A},求 R 的所有候选键。

5-4 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。

5-5 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 ρ={R1(AB),R2(BC)},判断分解 ρ 是否为无损连接分解。

5-6 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(AC),R2(BC)},判断分解 ρ 是否具有无损连接性。

5-7 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 ρ={R1(AB),R2(BC)},判断分解 ρ 是否保持依赖性。

5-8 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 ρ={R1(AC),R2(BC)},判断分解 ρ 是否保持依赖性。

5-9 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},判断 R 能达到第几范式。

5-10 设有关系模式 R(A,B,C,D),其函数依赖集为 F={B→D,AB→C},判断 R 能达到第几范式。

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},并有分解 ρ={R1(ABE),R2(CDEP)},判断 R1 为第几范式。