第5章 chapter 5

关系数据理论与模式求精

学习目标

本章从如何构造一个好的关系模式这一问题出发，逐步深入介绍基于函数依赖的关系数据库规范化理论和方法，包括函数依赖定义、函数依赖集理论、范式定义及分解算法等。本章的学习目标为熟练掌握函数依赖和关系数据库各种范式的基本概念和定义，并能运用基本函数依赖理论对关系模式逐步求精，以满足最终应用需求。

学习方法

本章内容理论性较强，涉及的概念较多且不易理解。首先，要正确理解函数依赖的概念，它属于语义范畴，只能根据现实世界中数据的语义来确定；其次，要结合实例，深入理解部分依赖和传递依赖带来的关系模式异常问题；另外，要多实践和练习，在函数依赖理论指导下对给定关系模式进行范式分解，从而巩固所学知识。

学习指南

本章的重点是5.1节、5.2节和5.4节，难点是5.3节。

本章导读

一个“好”的关系模式应该是数据冗余尽可能少，且不会发生插入异常、删除异常和更新异常等问题。为得到一个“好”的关系模式，模式分解是常用的方法。但模式分解时应考虑分解后的模式是否具有无损连接、保持依赖等特性。关系数据理论就是用来指导设计者设计出“好”的关系模式以及对已有的模式进行模式求精。学习本章时可围绕下列问题进行：

(1) 一个数据冗余的关系模式会导致什么异常？对一个信息冗余模式进行不正确分解后又会导致什么问题？衡量一个关系模式“好坏”的依据是什么？

(2) 什么是函数依赖？它与主码有何关系？

(3) 什么是部分函数依赖和传递函数依赖？它们分别会导致哪些异常？

(4) 什么是函数依赖集闭包？如何利用Armstrong公理计算？

(5) 什么是属性闭包？如何利用属性闭包方法计算关系模式的超码和候选码？

(6) 什么是正则覆盖？正则覆盖唯一吗？

(7) 什么是无损连接分解？判断无损连接分解的条件是什么？

(8) 什么是保持依赖分解？它有什么用处？

(9) 基于函数依赖理论的关系模式具有哪几种范式？它们之间的关系是什么？

(10) 为什么要将关系分解为 BCNF 范式和 3NF 范式?

(11) 为什么要进行模式求精? 如何对关系模式进行求精?

5.1 问题提出

数据库模式设计好坏是数据库应用系统成败的关键。对同一应用而言,不同设计者可能会设计出不同的数据库模式。那么什么样的数据库模式是一个“好”的模式? 又如何设计出一个“好”模式? 这正是本章要解决的问题。本节将描述两个问题:数据冗余导致的问题和模式分解导致的问题。

1. 数据冗余导致的问题

数据冗余是指同一信息在数据库中存储了多个副本,它可能引起下列问题。

- **冗余存储**:信息被重复存储,导致浪费大量存储空间。
- **更新异常**:当重复信息的一个副本被修改,所有副本都必须进行同样的修改。因此当更新数据时,系统要付出很大的代价来维护数据库的完整性,否则会面临数据不一致的危险。
- **插入异常**:只有当一些信息事先已经存放在数据库中时,另外一些信息才能存入数据库中。
- **删除异常**:删除某些信息时可能丢失其他信息。

【例 5.1】 考虑学生选课关系模式 SCE(studentNo, studentName, courseNo, courseName, score),属性集{studentNo, courseNo}是唯一候选码,也是主码。如果允许一名学生选修多门课程,且一门课程可被多个学生选修,则该关系实例可能出现数据冗余,如图 5-1 所示。

studentNo	studentName	courseNo	courseName	score
S0700001	李小勇	C001	高等数学	98
S0700001	李小勇	C002	离散数学	82
S0700001	李小勇	C006	数据库系统原理	56
S0700002	刘方晨	C003	计算机原理	69
S0700002	刘方晨	C004	C 语言程序设计	87
S0700002	刘方晨	C005	数据结构	77
S0700002	刘方晨	C007	操作系统	90
S0700003	王红敏	C001	高等数学	46
S0700003	王红敏	C002	离散数学	38
S0700003	王红敏	C007	操作系统	50

图 5-1 学生选课关系 SCE 实例

这种冗余会带来下列不好结果。

- 冗余存储：学生姓名和课程名被重复存储多次。
- 更新异常：当修改某学生的姓名或某课程的课程名时，可能只修改了部分副本的信息，而其他副本未被修改到。
- 插入异常：如果某学生没有选修课程，或某门课程未被任何学生选修时，则该学生或该课程信息不能存入数据库；否则，违背了实体完整性原则（主码值不能为空）。
- 删除异常：当一学生的所有选修课程信息都被删除时，则该学生的信息将被丢失。同样，当删除某门课程的全部学生选修信息时，该课程的信息也将被丢失。

关系模式 SCE 之所以会产生上述问题，是由于该模式中某些属性之间存在依赖关系，导致数据冗余而引起的。在 SCE 中，存在的属性依赖关系有：studentNo 决定 studentName，courseNo 决定 courseName，{studentNo，courseNo}共同决定 score。

如果将 SCE 分解为 S（studentNo，studentName），C（courseNo，courseName）和 E(studentNo，courseNo，score)3 个关系模式，则 SCE 中原有的 3 种属性依赖关系就分别分解到每个单独的关系模式中去了，这样就不会再出现上述异常现象，且数据冗余也得到了有效控制。

函数依赖理论正是用来改造关系模式，通过分解较大的关系模式来消除其中不合适的数据依赖，以解决数据冗余及其带来的各种问题。理想情况下，我们希望没有模式冗余，但有时出于性能方面考虑，可能会接受一些带有冗余的模式。

2. 模式分解导致的问题

SCE 转化为 S、C 和 E 3 个较小的关系模式之后，可减少冗余和消除各种异常。因此，直观上我们可得出这样的结论，冗余引起的问题可通过将一个关系模式分解为一些包含了原关系属性集的较小的关系模式集来解决，那么：

（1）什么样的关系模式需要进一步分解为较小的关系模式集？

（2）是否所有的模式分解都是有益的？

针对第一个问题人们已提出了关系模式应满足的条件即范式要求（5.3 节讨论）。通过这些范式，可以判断一个关系模式满足哪种范式要求，进而可知该模式可能存在哪些特定问题。因此，使用范式来考察特定的关系模式，有助于决定是否应该进一步分解一个已有关系模式。如果给定的关系模式不满足应用所要求的范式，那么就应选择特定的分解方法将其分解成满足范式要求的更小的关系模式集合。针对第二个问题，先来看一个实例。

【例 5.2】 设一关系模式 STU（studentNo，studentName，sex，birthday，native，classNo），其中 studentNo 为主码。假设将 STU 分解为以下两个子模式：

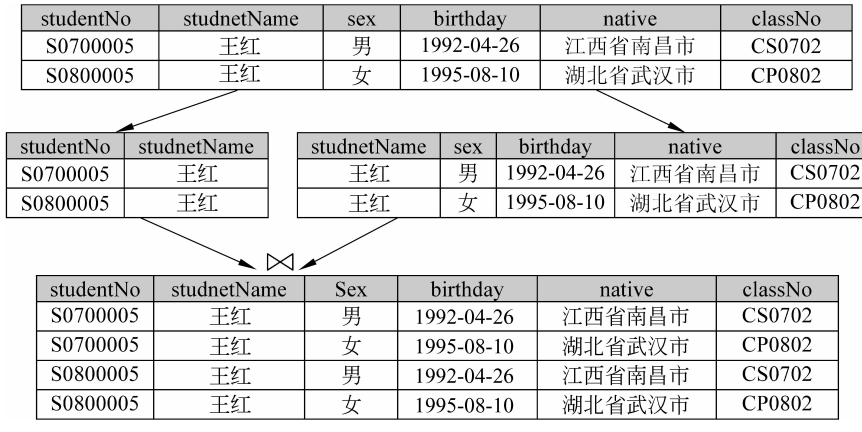
```
STU1(studentNo,studentName)
STU2(studentName,sex,birthday,native,classNo)
```

该分解存在的缺陷之一是可能导致信息损失。我们知道，在设计 STU 实体时，考虑到可能存在同名的学生，故将 studentNo 属性作为 STU 的主码，以唯一标识 STU 中的每个

学生实例。假设 STU 中有以下两个元组：

(S0700005,王红,男,1992-04-26,江西省南昌市,CS0702)
(S0800005,王红,女,1995-08-10,湖北省武汉市,CP0802)

如图 5-2 所示，首先将 STU 模式下的这两个元组分解为 STU1、STU2 模式下的元组；然后利用自然连接，试图根据分解后的元组还原原来的元组。结果显示，还原后除了得到原来的两个元组外，还多出了两个新元组。表面上看得到了更多的元组，但实际上得到的信息却变少了，因为无法区分哪个信息是属于哪个王红的？显然，这种分解是要尽量避免发生的，称之为**有损分解**(lossy decomposition)。反之，如果能够通过连接分解后所得到的较小关系完全还原被分解关系的所有实例，则称之为**无损分解**(lossless decomposition)，也称该分解具有**无损连接**特性。



studentNo	studnetName	sex	birthday	native	classNo
S0700005	王红	男	1992-04-26	江西省南昌市	CS0702
S0800005	王红	女	1995-08-10	湖北省武汉市	CP0802

studentNo	studnetName
S0700005	王红
S0800005	王红

studnetName	sex	birthday	native	classNo
王红	男	1992-04-26	江西省南昌市	CS0702
王红	女	1995-08-10	湖北省武汉市	CP0802

⋈

studentNo	studnetName	Sex	birthday	native	classNo
S0700005	王红	男	1992-04-26	江西省南昌市	CS0702
S0700005	王红	女	1995-08-10	湖北省武汉市	CP0802
S0800005	王红	男	1992-04-26	江西省南昌市	CS0702
S0800005	王红	女	1995-08-10	湖北省武汉市	CP0802

图 5-2　有损分解举例

上述分解的另一缺陷是部分属性之间的依赖关系已丢失。在 STU 中，属性 studentNo 是主码，可决定其中的全部属性。但是将关系模式 STU 分解为 STU1 和 STU2 后，由于属性 sex、birthday、age、native 和 classNo 等与属性 studentNo 分属于不同的关系模式中，那么这些属性对 studentNo 的依赖关系也就不再存在。也就是说，这种分解不是**保持依赖**(dependency preserving)的分解。而我们希望看到的是，被分解关系模式上的所有依赖关系都应该在分解得到的关系模式上保留。

因此，一个“好”的关系模式应该是数据冗余应尽可能少，且不会发生插入异常、删除异常和更新异常等问题。而且，当为减少冗余进行模式分解时，应考虑分解后的模式是否具有**无损连接**、**保持依赖**等特性。本章下列各节将讨论函数依赖、范式、函数依赖理论和模式分解算法等概念，并利用它来解决数据冗余和模式分解所引发的问题。

5.2　函数依赖定义

函数依赖(functional dependency，FD)是一种完整性约束，是现实世界事物属性之间的一种制约关系，它广泛地存在于现实世界之中。

1. 函数依赖

定义 5.1 设 $r(R)$为关系模式，$\alpha \subseteq R$，$\beta \subseteq R$。对任意合法关系 r 及其中任两个元组 t_i 和 t_j，$i \neq j$，若 $t_i[\alpha]=t_j[\alpha]$，则 $t_i[\beta]=t_j[\beta]$，则称 α **函数确定** β，或 β **函数依赖于** α，记作 $\alpha \rightarrow \beta$。

为了便于理解，可用椭圆表示属性或属性集，圆弧表示函数依赖，箭头指向被函数确定的属性集，则 $\alpha \rightarrow \beta$ 的函数依赖图如图 5-3 所示。

【例 5.3】 图 5-4 所示的是满足函数依赖 $AB \rightarrow C$ 的关系模式 $r(A,B,C,D)$的一个关系实例。从图中可以看出，对于任意两个在属性集$\{A,B\}$上取值相同的元组，它们在属性 C 上的取值也相同。例如，对于第 1 个和第 2 个元组：$t_1[A,B]=t_2[A,B]=(a1,b1)$，且 $t_1[C]=t_2[C]=c1$。如果在图中再增加一个元组$(a1,b1,c2,d1)$，此时就违背了函数依赖 $AB \rightarrow C$。

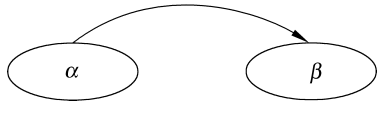


图 5-3 $\alpha \rightarrow \beta$ 函数依赖图

A	B	C	D
$a1$	$b1$	$c1$	$d1$
$a1$	$b1$	$c1$	$d2$
$a1$	$b2$	$c2$	$d1$
$a2$	$b1$	$c3$	$d1$

图 5-4 满足函数依赖 $AB \rightarrow C$ 的一个关系实例

对于函数依赖，需做如下说明：

(1) 函数依赖不是指关系模式 $r(R)$的某个或某些关系实例满足的约束条件，而是指关系模式 $r(R)$的所有关系实例均要满足的约束条件。

(2) 函数依赖是语义范畴的概念，只能根据数据的语义来确定函数依赖，是不能够被证明的。例如，“姓名→年龄”这个函数依赖只有在没有重名的条件下成立。如果有相同名字的人，则“年龄”就不再函数依赖于“姓名”了。

(3) 数据库设计者可以对现实世界作强制的规定。例如，在上例中，设计者可以强行规定不允许同名人出现，因而使函数依赖“姓名→年龄”成立。这样当插入某个元组时该元组上的属性值必须满足规定的函数依赖，若发现有同名人存在，则拒绝插入该元组。

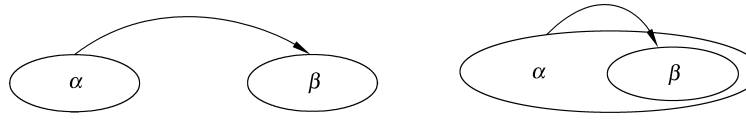
(4) 码约束是函数依赖的一个特例。码属性(集)相当于定义 5.1 中的 α，关系中的所有属性相当于定义 5.1 中的 β。

2. 平凡与非平凡函数依赖

定义 5.2 在关系模式 $r(R)$中，$\alpha \subseteq R$，$\beta \subseteq R$。若 $\alpha \rightarrow \beta$，但 $\beta \not\subseteq \alpha$，则称 $\alpha \rightarrow \beta$ 是**非平凡函数依赖**。否则，若 $\beta \subseteq \alpha$，则称 $\alpha \rightarrow \beta$ 是**平凡函数依赖**。

非平凡函数依赖和平凡函数依赖的依赖图分别如图 5-5(a)和图 5-5(b)所示。对于任一关系模式，平凡函数依赖都是必然成立的，它不反映新的语义。例如 $A \rightarrow A$ 在所有包含属性 A 的关系上都是满足的，因为对所有满足 $t_i[A]=t_j[A]$，都有 $t_i[A]=t_j[A]$。同样 $AB \rightarrow A$ 也在所有包含 A 属性的关系中都是满足的。因此若不特别声明，总是讨论

非平凡函数依赖。



(a) 非平凡函数依赖　　(b) 平凡函数依赖

图 5-5　非平凡函数依赖及平凡函数依赖图

3. 完全函数依赖和部分函数依赖

定义 5.3　在关系模式 $r(R)$中，$\alpha \subseteq R$，$\beta \subseteq R$，且 $\alpha \rightarrow \beta$ 是非平凡函数依赖。若对任意的 $\gamma \subset \alpha$，$\gamma \rightarrow \beta$ 都不成立，则称 $\alpha \rightarrow \beta$ 是**完全函数依赖**，简称**完全依赖**。否则，若存在非空的 $\gamma \subset \alpha$，使 $\gamma \rightarrow \beta$ 成立，则称 $\alpha \rightarrow \beta$ 是**部分函数依赖**，简称**部分依赖**。

$\alpha \rightarrow \beta$ 是完全依赖，意指 β 不依赖于 α 的任何子属性(集)，而部分依赖则是指 β 依赖于 α 的部分属性(集)。部分依赖 $\alpha \rightarrow \beta$ 的依赖图如图 5-6 所示。

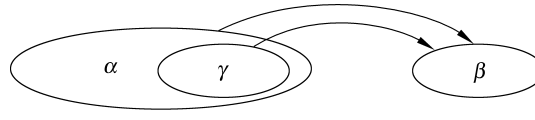


图 5-6　部分依赖$\alpha \rightarrow \beta$的依赖图

可以看出，当 α 是单属性时，则 $\alpha \rightarrow \beta$ 完全函数依赖总是成立的。例如，在关系 SCE 中，存在下列完全依赖：

```
studentNo→studentName
courseNo→courseName
{studentNo,courseNo}→score
```

而下列依赖则是部分依赖：

```
{studentNo,courseNo}→studentName
{studentNo,courseNo}→courseName
```

部分依赖导致的数据冗余及各种异常已在例 5.1 中分析过。

4. 传递函数依赖

定义 5.4　在关系模式 $r(R)$中，设 $\alpha \subseteq R$，$\beta \subseteq R$，$\gamma \subseteq R$。若 $\alpha \rightarrow \beta$，$\beta \rightarrow \gamma$，则必存在函数依赖 $\alpha \rightarrow \gamma$；若 $\alpha \rightarrow \beta$，$\beta \rightarrow \gamma$ 和 $\alpha \rightarrow \gamma$ 都是非平凡函数依赖，且 $\beta \not\rightarrow \alpha$，则称 $\alpha \rightarrow \gamma$ 是**传递函数依赖**，简称**传递依赖**。

传递依赖 $\alpha \rightarrow \gamma$ 的依赖图如图 5-7 所示。与部分依赖一样，传递依赖也可能会导致数据冗余及产生各种异常。

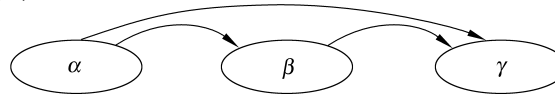


图 5-7　传递依赖$\alpha \rightarrow \gamma$的依赖图

【例 5.4】 在关系模式 SCI(studentNo, classNo, className, institute)中,属性 studentNo 决定属性 classNo,属性 classNo 决定 className 和 institute。因此,关系模式 SCI 中存在下列函数依赖:

```
studentNo→{classNo,className,institute}
classNo→{className,institute}
```

由于关系模式 SCI 中存在传递依赖 studentNo→{className,institute},故也导致了下列数据冗余、更新异常、插入异常及删除异常。

(1) 数据冗余:由于每一个班的学生都是同一学院,且具有相同的班级名称,因此班级名称和学院的信息会重复出现,重复次数与该班学生人数相同。

(2) 更新异常:当班级在学院之间调整时(虽然实际上很少发生),比如信息学院某班的学生全部调整到计算机学院,修改时必须同时更新该班所有学生的 institute 属性值。

(3) 插入异常:如果某个班因种种原因(如刚刚成立),目前暂时没有在校学生,就无法把这个班的信息存入数据库(不能插入主码为空值的元组)。

(4) 删除异常:如果某个班的学生全部毕业了,则在删除该班学生信息的同时,把这个班的信息也丢掉了。

函数依赖是指关系模式中属性之间存在的一种约束关系。这种约束关系既可以是现实世界事物或联系的属性之间客观存在的约束,也可以是数据库设计者根据应用需求或设计需要强加给数据的一种约束。但不论是哪种约束,一旦确定,进入数据库中的所有数据都必须严格遵守。因此,正确了解数据的意义及确定属性之间的函数依赖关系,对设计一个好的关系模式是十分重要的。

5.3 范 式

给定一个关系模式,需要确定它是否是一个"好"的设计。如果不是,则需要将其分解为一些小的关系模式。因此,首先需要了解当前关系模式中属性之间的关系。

基于函数依赖理论,关系模式可分成第一范式(1NF),第二范式(2NF),第三范式(3NF)和 Boyce-Codd 范式(BCNF)。这几种范式的要求一个比一个严格,它们之间的联系为 BCNF⊂3NF⊂2NF⊂1NF。即满足 BCNF 范式的关系一定满足 3NF 范式,满足 3NF 范式的关系一定满足 2NF 范式,满足 2NF 范式的关系一定满足 1NF 范式。

5.3.1 第一范式(1NF)——码

定义 5.5 如果一关系模式 $r(R)$的每个属性对应的域值都是不可分的,则称 $r(R)$属于**第一范式**,记为 $r(R)\in 1NF$。

第一范式的目标是:将基本数据划分成称为实体集或表的逻辑单元,当设计好每个实体后,需要为其指定**主码**。

关系数据库的模式至少应是第一范式。图 5-8 所示的关系模式是一个非规范化的

关系模式。因为 address 的值域是可分的。

studentNo	studentName	sex	birthday	age	address			classNo
					province	city	street	

图 5-8　非规范化的关系模式

将上述关系模式变为图 5-9 所示的形式才是满足 1NF 范式的关系模式。

studentNo	studentName	sex	birthday	age	province	city	street	classNo

图 5-9　1NF 规范化后的关系模式

5.3.2　第二范式(2NF)——全部是码

定义 5.6　设有一关系模式 $r(R)$，$\alpha \subseteq R$。若 α 包含在 $r(R)$ 的某个候选码中，则称 α 为**主属性**，否则 α 为**非主属性**。

在 SCE 关系中，属性集{studentNo，CourseNo}是 SCE 的唯一候选码。因此，属性 studentNo 和 courseNo 为主属性，其余属性为非主属性。

定义 5.7　如果一个关系模式 $r(R) \in 1NF$，且所有非主属性都完全函数依赖于 $r(R)$ 的候选码，则称 $r(R)$ 属于**第二范式**，记为 $r(R) \in 2NF$。

由于 SCE 中存在依赖关系 studentNo→studentName 和 courseNo→courseName，即非主属性 studentName 和 courseName 部分依赖于 SCE 的候选码，故 SCE$\notin$2NF。

也就是说，对于满足第一范式(1NF)的关系模式，如果有复合候选码(即多个属性共同构成的候选码)，那么非主属性不允许依赖于部分的候选码属性，必须依赖于全部的候选码属性——**全部是码**。

第二范式的目标是：将只部分依赖于候选码(即依赖于候选码的部分属性)的非主属性通过关系模式分解移到其他表中去。

违背了 2NF 的关系模式，即存在非主属性对候选码的部分依赖，则可能导致例 5.1 所述的数据冗余及异常问题。对于非 2NF 的关系模式，可通过分解进行规范化，以消除部分依赖。如将关系模式 SCE 分解为关系模式 S、C 和 E。这样在每个关系模式中，所有非主属性对候选码都是完全函数依赖，因此都属于 2NF。

第二范式虽然消除了由于非主属性对候选码的部分依赖所引起的冗余及各种异常，但并没有排除传递依赖。根据 2NF 定义，例 5.4 中的 SCI 关系满足 2NF，但由于其存在传递依赖，故仍然存在冗余及各种异常。因此，还需要对其进一步规范化。

5.3.3　第三范式(3NF)——仅仅是码

定义 5.8　如果一个关系模式 $r(R) \in 2NF$，且所有非主属性都直接函数依赖于 $r(R)$ 的候选码(即不存在非主属性传递依赖于候选码)，则称 $r(R)$ 属于**第三范式**，记为

$r(R)\in 3NF$。

也就是说，对于满足第二范式(2NF)的关系模式，非主属性不能依赖于另一个(组)非主属性(这样就形成了对候选码的传递依赖)，即非主属性只能直接依赖于候选码——**仅仅是码**。

第三范式的目标是：将不直接依赖于候选码(即传递依赖于候选码)的非主属性通过关系模式分解移到其他表中去。

总之，所有的非主属性应该直接依赖于(即不能存在传递依赖，这是 3NF 的要求)全部的候选码(即必须完全依赖，不能存在部分依赖，这是 2NF 的要求)。

【例 5.5】 $r(R)=r(A,B,C,D)$，函数依赖集 $F=\{AB\rightarrow C,B\rightarrow D\}$。$r(R)$的候选码为 AB，$r(R)\notin 2NF$，因为函数依赖 $B\rightarrow D$ 中的决定属性 B 只是候选码的一部分，即 D 部分依赖于候选码 AB。可将 $r(R)$分解为 $r_1(R_1)=r_1(A,B,C)$、$r_2(R_2)=r_2(B,D)$，则分解得到的 $r_1(R_1)$和 $r_2(R_2)$都属于 3NF，$r_1(R_1)$的候选码为 AB，$r_2(R_2)$的候选码为 B。

【例 5.6】 $r(R)=r(A,B,C)$，函数依赖集 $F=\{A\rightarrow B,B\rightarrow C\}$。$r(R)$的候选码为 A，$r(R)\in 2NF$，但 $r(R)\notin 3NF$，因为函数依赖 $B\rightarrow C$ 中的决定属性 B 不是候选码。可将 $r(R)$分解为 $r_1(R_1)=r_1(A,B)$、$r_2(R_2)=r_2(B,C)$，则分解得到的 $r_1(R_1)$和 $r_2(R_2)$都属于 3NF，$r_1(R_1)$的候选码为 A，$r_2(R_2)$的候选码为 B。

【例 5.7】 $r(R)=r(A,B,C,D,E)$，函数依赖集 $F=\{AB\rightarrow C,B\rightarrow D,C\rightarrow E\}$。$r(R)$的候选码为 AB，$r(R)\notin 2NF$，因为函数依赖 $B\rightarrow D$ 中的决定属性 B 只是候选码的一部分，即 D 部分依赖于候选码 AB。

【例 5.8】 $r(R)=r(A,B,C)$，函数依赖集 $F=\{AB\rightarrow C,C\rightarrow A\}$。$r(R)$的候选码为 AB 或 BC，$r(R)\in 3NF$，因为关系模式 $r(R)$没有非主属性，也就不可能有非主属性对候选码的部分依赖和传递依赖。

对于非 3NF 的关系模式，可通过分解进行规范化，以消除部分依赖和传递依赖。例如，可将例 5.7 中的 $r(R)$分解为 $r_1(R_1)=r_1(A,B,C)$、$r_2(R_2)=r_2(B,D)$、$r_3(R_3)=r_3(C,E)$，显然该分解得到的 $r_1(R_1)$、$r_2(R_2)$和 $r_3(R_3)$都属于 3NF，$r_1(R_1)$的候选码为 AB，$r_2(R_2)$的候选码为 B，$r_3(R_3)$的候选码为 C。

5.3.4 Boyce-Codd 范式(BCNF)

下面介绍能排除所有部分依赖和传递依赖的 BNCF 范式。

定义 5.9 给定关系模式 $r(R)\in 1NF$，函数依赖集 F，若 F^+(表示 F 的闭包，5.4.1 节讨论)中的所有函数依赖 $\alpha\rightarrow\beta(\alpha\subseteq R,\beta\subseteq R)$至少满足下列条件之一：

(1) $\alpha\rightarrow\beta$ 是平凡函数依赖(即 $\beta\subseteq\alpha$)；

(2) α 是 $r(R)$的一个超码(即 α 中包含 R 的候选吗)。

则称 $r(R)$属于 **Boyce-Codd 范式**，记为 $r(R)\in BCNF$。

换句话说，在关系模式 $r(R)$中，如果 F^+ 中的每一个非平凡函数依赖的决定属性集 α 都包含候选码，则 $r(R)\in BCNF$。必须说明的是，为确定 $r(R)$是否满足 BCNF，必须考虑 F^+ 而不是 F 中的每个函数依赖。

直观上，若一关系模式 $r(R)$满足 BCNF，则其所有非平凡函数依赖都是由“候选码”

确定的依赖关系。这样,关系实例 r 中的每个元组都可看作是一个实体或联系,即由候选码来标识并由其余属性来描述。因此,从函数依赖角度可得出,一个满足 BCNF 的关系模式必然满足下列结论:

(1) 所有非主属性都完全函数依赖于每个候选码;

(2) 所有主属性都完全函数依赖于每个不包含它的候选码;

(3) 没有任何属性完全函数依赖于非候选码的任何一组属性。

因此,BCNF 不仅排除了任何属性(包括主属性和非主属性)对候选码的部分依赖和传递依赖,而且排除了主属性之间的传递依赖,其依赖关系如图 5-10 所示。其中,类似"$\alpha\rightarrow\beta$,$\beta\rightarrow$非主属性 1,非主属性 1$\rightarrow$非主属性 2"的函数依赖都是不可能存在的。

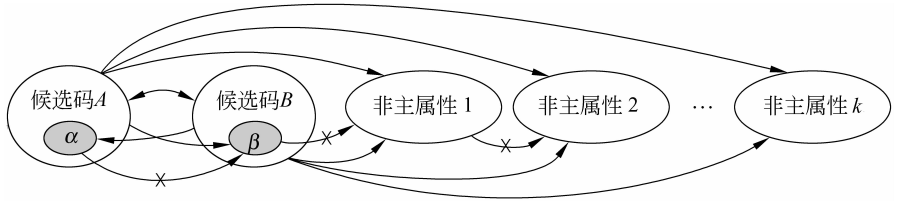


图 5-10 BCNF 关系模式中的函数依赖

因此,BCNF 确保了通过函数依赖不能再检查出任何冗余,即只考虑函数依赖关系时,BCNF 已是最好的范式。

下面给出几个例子。

【例 5.9】 $r(R)=r(A,B,C)$,$F=\{A\rightarrow B,B\rightarrow C\}$。$r(R)$的候选码为 A,$r(R)\notin$ BCNF,因为函数依赖 $B\rightarrow C$ 中的决定属性 B 不是超码。

【例 5.10】 $r(R)=r(A,B,C)$,$F=\{AB\rightarrow C,C\rightarrow A\}$。$r(R)$的候选码为 AB 或 BC,$r(R)\notin$BCNF,因为 $C\rightarrow A$ 的决定属性 C 不是超码。

【例 5.11】 $r(R)=r(A,B,C)$,$F=\{AB\rightarrow C,BC\rightarrow A\}$。$r(R)$的候选码为 AB 或 BC,$r(R)\in$BCNF,因为两个函数依赖中的决定属性 AB 或 BC 都是 $r(R)$的候选码。

对于非 BCNF 的关系模式,可通过分解进行规范化,以消除部分依赖和传递依赖。例如,可将例 5.9 中的 $r(R)$ 分解为 $r_1(R_1)=r_1(A,B)$、$r_2(R_2)=r_2(B,C)$,或 $r_1(R_1)=r_1(A,B)$、$r_2(R_2)=r_2(A,C)$。显然,这两种分解得到的 $r_1(R_1)$和 $r_2(R_2)$都属于 BCNF。但是,后一种分解不是保持依赖分解(参见例 5.23)。

因此,满足 BCNF 要求的模式分解,可能不是保持依赖分解。

下面从 BCNF 的定义出发给出 3NF 的另一种定义。

定义 5.10 给定关系模式 $r(R)\in$1NF,函数依赖集 F,若对 F^+ 中的所有函数依赖 $\alpha\rightarrow\beta(\alpha\subseteq R,\beta\subseteq R)$至少满足下列条件之一:

(1) $\alpha\rightarrow\beta$ 是平凡函数依赖(即 $\beta\subseteq\alpha$);

(2) α 是 $r(R)$的一个超码(即 α 中包含 R 的候选码);

(3) $\beta-\alpha$ 中的每个属性是 $r(R)$的候选码的一部分。

则称 $r(R)$属于**第三范式**,记为 $r(R)\in$3NF。

从定义 5.9 和定义 5.10 可以看出,3NF 与 BCNF 的前两个条件是相同的,区别之处