第 3 章 关系数据库设计理论

本章要点

- 关系数据库设计理论概述
- 函数依赖
- 范式
- 规范化
- Armstrong 公理系统
- 两个函数依赖集的等价和最小函数依赖集
- 关系模式的分解

设计任何一种数据库系统都需要构造合适的数据模式，即解决逻辑结构问题，关系数据库设计理论正是关系数据库理论基础和逻辑设计的有力工具。数据库设计是针对一个给定的应用环境设计优化的数据库逻辑结构和物理结构，并据此建立数据库及其应用系统。在本章中介绍关系数据理论概述、规范化、数据依赖的公理系统、关系模式的分解等内容。

3.1 关系数据库设计理论概述

关系数据库设计理论最早由数据库创始人 E.F.Codd 提出，后经很多专家学者做了深入的研究与发展，形成一整套有关关系数据库设计的理论。

设计一个合适的关系数据库系统的关键是关系数据库模式的设计，即应构造几个关系模式，每个模式有哪些属性，怎样将这些相互关联的关系模式组建成一个适合的关系模型，关系数据库的设计必须在关系数据库设计理论的指导下进行。

关系数据库设计理论有 3 个方面的内容，即函数依赖、范式和模式设计。函数依赖起核心作用，它是模式分解和模式设计的基础；范式是模式分解的标准。

关系数据库设计的关键是关系模式的设计，下面举例说明好的关系模式问题。

【例 3.1】 设计一个学生课程数据库，其关系模式为 SDSC(Sno, Sname, Age, Dept, DeptHead, Cno,Grade)，各属性的含义为学号、姓名、年龄、系名、系主任姓名、课程号、成绩。根据实际情况，这些属性的语义规定如下。

（1）一个系有若干学生，一个学生只属于一个系。

（2）一个系只有一个系主任。

（3）一个学生可以选修多门课程，一门课程可被多个学生选修。

（4）每个学生学习每门课程有一个成绩。

关系模式 SDSC 在某一时刻的一个实例（即数据表）如表 3.1 所示。

表 3.1　SDSC 表

Sno	Sname	Age	Dept	DeptHead	Cno	Grade
141001	刘星宇	22	电子工程	李建明	101	94
141001	刘星宇	22	电子工程	李建明	204	92
141001	刘星宇	22	电子工程	李建明	901	95
141002	王小凤	20	电子工程	李建明	101	76
141002	王小凤	20	电子工程	李建明	204	74
141002	王小凤	20	电子工程	李建明	901	84
142001	杨燕	21	计算机应用	程海涛	204	87
142001	杨燕	21	计算机应用	程海涛	901	82
142004	周培杰	21	计算机应用	程海涛	204	90
142004	周培杰	21	计算机应用	程海涛	901	92

从上述语义规定且分析表中数据可以看出，(Sno, Cno)能唯一标识一个元组，所以(Sno, Cno)为该关系模式的主码，但在进行数据库操作时会出现以下问题。

（1）数据冗余：当一个学生选修多门课程时会出现数据冗余，导致姓名、性别和课程名属性多次重复存储，系名和系主任姓名也多次重复。

（2）插入异常：如果某个新系没有招生，由于没有学生，则系名和系主任姓名无法插入，根据关系实体完整性约束，主码(Sno,Cno)不能取空值，此时 Sno、Cno 均无值，所以不能进行插入操作。

另外，学生未选修课程，则 Cno 无值，其学号、姓名和年龄无法插入，因为实体完整性约束规定，主码(Sno, Cno)不能部分为空，也不能进行插入操作。

（3）删除异常：当某系学生全部毕业还未招生时要删除全部记录，系名和系主任姓名也被删除，而这个系仍然存在，这就是删除异常。

（4）修改异常：如果某系更换系主任，则属于该系的记录都要修改 DeptHead 的内容，若有不慎，会造成漏改或误改，造成数据的不一致性，破坏数据完整性。

由于存在上述问题，SDSC 不是一个好的关系模式。为了克服这些异常，将 S 关系分解为学生关系 S (Sno, Sname, Age, Dept)、系关系 D(Dept, DeptHead)、选课关系 SC(Sno, Cno,Grade)，这 3 个关系模式的实例如表 3.2～表 3.4 所示。

表 3.2　S 表

Sno	Sname	Age	Dept
141001	刘星宇	22	电子工程
141002	王小凤	20	电子工程
142001	杨燕	21	计算机应用
142004	周培杰	21	计算机应用

表 3.3　D 表

Dept	DeptHead
电子工程	李建明
计算机应用	程海涛

表 3.4　SC 表

Sno	Cno	Grade
141001	101	94
141001	204	92
141001	901	95
141002	101	76
141002	204	74
141002	901	84
142001	204	87
142001	901	82
142004	204	90
142004	901	92

可以看出，首先是数据冗余明显降低。当新增一个系时只需在关系 D 中增加一条记录即可，当某个学生未选修课程时只需在关系 S 中增加一条记录，而与选课关系 SC 无关，这就避免了插入异常。当某系学生全部毕业时只需在关系 S 中删除全部记录，不会影响到系名和系主任姓名等信息，这就避免了删除异常。当更换系主任时只需在关系 D 中修改一条记录中的 DeptHead 属性的内容，这就避免了修改异常。

但是，一个好的关系模式不是在任何情况下都是最优的，例如查询某个学生的系主任姓名和成绩，就需要通过 3 个表的连接操作来完成，需要的开销较大，在实际工作中要以应用系统功能与性能需求为目标进行设计。

规范化设计关系模式，将结构复杂的关系模式分解为结构简单的关系模式，使不好的关系模式转变为较好的关系模式，这是下一节要讨论的内容。

3.2　规　范　化

在第 2 章中已定义关系模式可以形式化地表示为一个五元组，即 R(*U*, D, DOM, F)。

其中 R 是关系名，*U* 是组成该关系的属性名的集合，D 是属性来自的域，DOM 是属性向域的映像集合，F 是属性间的数据依赖关系集合。

由于 D 和 DOM 对设计好的关系模式作用不大，一般将关系模式简化为一个三元组，即 R<*U*, F>，有时还简化为 R(*U*)。

数据依赖（data dependency）是一个关系内部属性与属性之间的一种约束关系，是数据内在的性质，是语义的体现。

数据依赖有多种类型，下面主要介绍函数依赖（Functional Dependency，FD），简单介绍多值依赖（Multivalued Dependency，MVD）和连接依赖（Join Dependency，JD）。

3.2.1 函数依赖、码和范式

1. 函数依赖

函数依赖是关系数据库规范化理论的基础。

定义 3.1 设 R(*U*)是属性集 *U* 上的关系模式，*X*、*Y* 是 *U* 的子集。若对于 R(*U*)的任意一个可能的关系 r，r 中不可能存在两个元组在 *X* 上的属性值相等，而在 *Y* 上的属性值不等，则称 *X* 函数确定 *Y* 或 *Y* 函数依赖于 *X*，记作 $X \rightarrow Y$，称 *X* 为决定因素，*Y* 为依赖因素。若 *Y* 不函数依赖于 *X*，则记作 $X \nrightarrow Y$。若 $X \rightarrow Y$，$Y \rightarrow X$，则记作 $X \leftrightarrow Y$。

例如，关系模式 SDSC(Sno, Sname, Age, Dept, DeptHead, Cno, Grade)有：

U={Sno, Sname, Age, Dept, DeptHead, Cno,Grade}

F={Sno→Sname, Sno→Age, Sno→Dept, Dept→DeptHead, Sno→DeptHead, (Sno, Cno)→Grade}

一个 Sno 有多个 Grade 的值与之对应，Grade 不能函数依赖于 Sno，即 Sno↛Grade，同理，Cno↛Grade，但 Grade 可被(Sno, Cno)唯一确定，所以(Sno, Cno)→Grade。

注意：函数依赖是指 R 的所有关系实例都要满足的约束条件，不是针对某个或某些关系实例满足的约束条件。

函数依赖和其他数据之间的依赖关系一样，是语义范畴的概念，人们只能根据数据的语义来确定函数依赖。

函数依赖与属性之间联系的类型有关。

（1）如果 *X* 和 *Y* 之间是 1:1 关系（一对一关系），则存在函数依赖 $X \leftrightarrow Y$，例如学生没有重名时 Sno↔Sname。

（2）如果 *X* 和 *Y* 之间是 1:*n* 关系（一对多关系），则存在函数依赖 $X \rightarrow Y$，如学号和姓名、部门名之间都有 1:*n* 关系，所以 Sno→Age，Sno→Dept。

（3）如果 *X* 和 *Y* 之间是 *m*:*n* 关系（多对多关系），则 *X* 和 *Y* 之间不存在函数依赖，如学生和课程之间就是 *m*:*n* 关系，所以 Sno 和 Cno 之间不存在函数依赖关系。

由于函数依赖与属性之间联系的类型有关，所以可以从分析属性之间的联系入手，确定属性之间的函数依赖。

定义 3.2 若 $X \rightarrow Y$ 是一个函数依赖，且 $Y \subseteq X$，则称 $X \rightarrow Y$ 是一个平凡函数依赖，否则称为非平凡函数依赖。例如，(Sno, Cno)→Sno、(Sno, Cno)→Cno 都是平凡函数依赖。

若不特别声明，本书讨论的都是非平凡函数依赖。

定义 3.3 设 R(*U*)是属性集 *U* 上的关系模式，*X*、*Y* 是 *U* 的子集。设 $X \rightarrow Y$ 是一个函数依赖，并且对于任何 *X* 的一个真子集 *X'*，$X' \rightarrow Y$ 都不成立，则称 $X \rightarrow Y$ 是一个完全函数依赖（full functional dependency），即 *Y* 函数依赖于整个 *X*，记作 $X \xrightarrow{f} Y$。

定义 3.4 设 R(*U*)是属性集 *U* 上的关系模式，*X*、*Y* 是 *U* 的子集。设 $X \rightarrow Y$ 是一个函数依赖，但不是完全函数依赖，则称 $X \rightarrow Y$ 是一个部分函数依赖（partial functional dependency），或称 *Y* 函数依赖于 *X* 的某个真子集，记作 $X \xrightarrow{p} Y$。

例如在关系模式 SDSC 中，因为 Sno↛Grade，Cno↛Grade，所以 (Sno,Cno) $\xrightarrow{f}$ Grade 。因为 Sno→Age，所以 (Sno,Cno) $\xrightarrow{p}$ Age 。

定义 3.5 设 R(*U*)是一个关系模式，*X*、*Y*、*Z* 是 *U* 的子集，如果 $X \to Y (Y \not\subseteq X)$，$Y \not\to X$，$Y \to Z$ 成立，则称 *Z* 传递函数依赖（transitive functional dependency）于 *X*，记为 $X \xrightarrow{t} Z$ 。

注意：如果有 $Y \to X$，则 $X \leftrightarrow Y$，此时称 *Z* 对 *X* 直接函数依赖，而不是传递函数依赖。

例如在关系模式 SDSC 中，Sno→Dept，但 Dept↛Sno，且 Dept→DeptHead, 所以 Sno $\xrightarrow{t}$ DeptHead 。

2. 码

定义 3.6 设 *K* 为 R<*U*, F>中的属性或属性组，若 $K \xrightarrow{f} U$ ，则 *K* 为 R 的候选码（或称候选键、候选关键字，Candidate key）。若有多个候选码，则选定其中的一个作为主码（或称主键，Primary key）。

包含在任何一个候选码中的属性称为主属性（prime attribute）。不包含在任何候选码中的属性称为非主属性（non-prime attribute）或非码属性（non-key attribute）。最简单的情况为单个属性是码，最极端的情况为整个属性组是码，称为全码（All-key）。

例如，在关系模式 S (Sno, Age, Dept)中 Sno 是码，而在关系模式 SC(Sno, Cno, Grade)中属性组合(Sno, Cno)是码。

在后面的章节中主码和候选码都简称为码，读者可从上下文加以区分。

定义 3.7 关系 R 中的属性或属性组 X 并非 R 的码，但 X 是另一个关系模式的码，则称 X 是 R 的外部码（foreign key），也称外码。

例如，在关系模式 SC(Sno, Cno, Grade)中 Sno 不是主码，但 Sno 是关系模式 S(Sno, Sname, Age, Dept)的主码，所以 Sno 是 SC 的外码，同理，Cno 也是 SC 的外码。

主码与外码提供了一个表示关系间的联系手段，例如关系模式 S 与 SC 的联系就是通过 Sno 在 S 中是主码而在 SC 中是外码来实现的。

3. 范式

规范化的基本思想是尽量减小数据冗余，消除数据依赖中不合适的部分，解决插入异常、删除异常和更新异常等问题，这就要求设计出的关系模式满足一定的条件。在关系数据库的规范化过程中，为不同程度的规范化要求设立的不同标准或准则称为范式。满足最低要求的称为第一范式，简称1NF，在第一范式的基础上满足进一步要求的称为第二范式，简称 2NF，以此类推。

1971 年至 1972 年，E.F.Codd 系统地提出了 1NF、2NF、3NF 的概念，讨论了关系模式的规范化问题。1974 年，Codd 和 Boyce 又共同提出了一个新范式，即 BCNF。1976 年有人提出了 4NF，后来又有人提出了 5NF。

各个范式之间的集合关系可以表示为 5NF⊂4NF⊂BCNF⊂3NF⊂2NF⊂1NF，如图 3.1 所示。

一个低一级范式的关系模式通过模式分解可以转换成若干个高一级范式的关系模式的集合，该过程称为规范化。

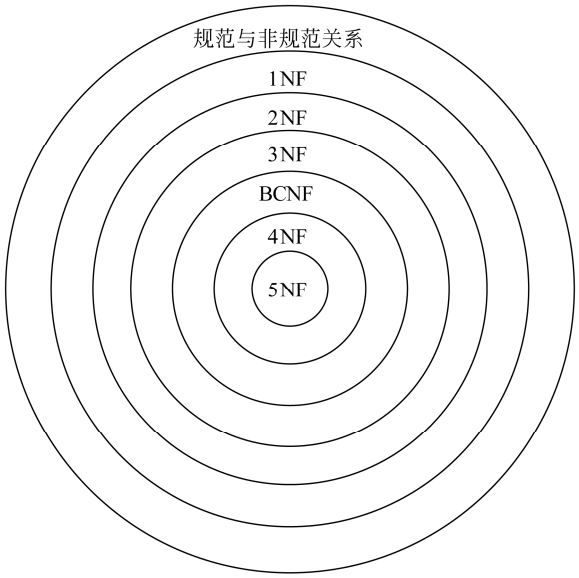

图 3.1　各范式之间的关系

3.2.2　1NF

定义 3.8　在一个关系模式 R 中，如果 R 的每一个属性都是不可再分的数据项，则称 R 属于第一范式（1NF），记作 R∈1NF。

第一范式是最基本的范式，在关系中每个属性都是不可再分的简单数据项。

【例 3.2】　第一范式规范化举例。

表 3.5 所示的关系 R 不是 1NF，关系 R 转化为 1NF 的结果如表 3.6 所示。

表 3.5　关系 R

Sno	Sname	Cname
141001	刘星宇	数字电路, 英语
141002	王小凤	数字电路, 英语
142001	杨燕	数据库系统, 英语
142004	周培杰	数据库系统, 英语

表 3.6　关系 R 转化为 1NF

Sno	Sname	Cname
141001	刘星宇	数字电路
141001	刘星宇	英语
141002	王小凤	数字电路
141002	王小凤	英语
142001	杨燕	数据库系统
142001	杨燕	英语

续表

Sno	Sname	Cname
142004	周培杰	数据库系统
142004	周培杰	英语

3.2.3 2NF

定义 3.9 对于关系模式 R∈1NF，且 R 中的每一个非主属性都完全函数依赖于任意一个候选码，该关系模式 R 属于第二范式，记作 R∈2NF。

第二范式的规范化指将 1NF 关系模式通过投影分解，消除非主属性对候选码的部分函数依赖，转换成 2NF 关系模式的集合过程。

在分解时遵循“一事一地”的原则，即一个关系模式描述一个实体或实体间的联系，如果多于一个实体或联系，则进行投影分解。

【例 3.3】 第二范式规范化举例。

在例 3.1 的关系模式 SDSC(Sno, Sname, Age, Dept, DeptHead, Cno, Grade)中，各属性的含义为学号、姓名、年龄、系名、系主任姓名、课程名、成绩，(Sno, Cno)为该关系模式的候选码。

该模式属于第一范式，函数依赖关系如下。

$(Sno, Cno) \xrightarrow{f} Grade$

$Sno \rightarrow Sname$，$(Sno, Cno) \xrightarrow{p} Sname$

$Sno \rightarrow Age$，$(Sno, Cno) \xrightarrow{p} Age$

$Sno \rightarrow Dept$，$(Sno, Cno) \xrightarrow{p} Dept$，$Dept \rightarrow DeptHead$

$Sno \xrightarrow{t} DeptHead$，$(Sno, Cno) \xrightarrow{p} DeptHead$

以上函数依赖关系可用函数依赖图表示，如图 3.2 所示。

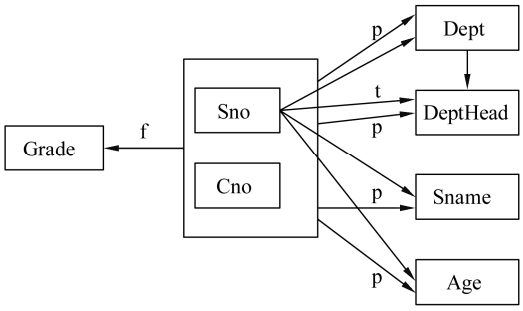

图 3.2 SDSC 中的函数依赖图

可以看出，Sno、Cno 为主属性，Sname、Age、Dept、DeptHead、Grade 为非主属性，由于存在非主属性 Sname 对候选码(Sno, Cno)的部分依赖，所以 $SDSC \notin 2NF$ 。

在 SDSC 中既存在完全函数依赖，又存在部分函数依赖和传递函数依赖，导致数据冗余、插入异常、删除异常、修改异常等问题，这在数据库中是不允许的。

根据“一事一地”的原则，将关系模式 SDSC 分解为两个关系模式。

SD(Sno, Sname, Age, Dept, DeptHead)

SC(Sno, Cno, Grade)

分解后的函数依赖图如图 3.3 和图 3.4 所示。

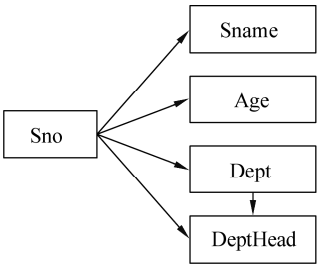

图 3.3　SD 中的函数依赖图

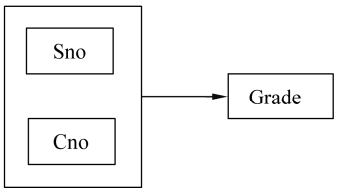

图 3.4　SC 中的函数依赖图

分解后的关系模式 SD 的候选码是 Sno，关系模式 SC 的候选码是(Sno, Cno)，非主属性对候选码都是完全函数依赖的，从而消除了非主属性对候选码的部分函数依赖，所以 SD∈2NF，SC∈2NF，它们之间通过 SC 中的外键 Sno 相联系，在需要时进行自然连接，恢复原来的关系，这种分解不会损失任何信息，具有无损连接性。

注意：如果 R 的候选码都是单属性，或 R 的全体属性都是主属性，则 R∈2NF。

3.2.4　3NF

定义 3.10　如果关系模式 R∈2NF，R 中的所有非主属性对任何候选码都不存在传递函数依赖，则称 R 属于第三范式，记作 R∈3NF。

第三范式具有以下性质。

（1）如果 R∈3NF，则 R 也是 2NF。

（2）如果 R∈2NF，则 R 不一定是 3NF。

2NF 的关系模式解决了 1NF 中存在的一些问题，但 2NF 的关系模式 SD 在进行数据操作时仍然存在以下问题。

（1）数据冗余：每个系名和系主任姓名存储的次数等于该系的学生人数。

（2）插入异常：当一个新系没有招生时有关该系的信息无法插入。

（3）删除异常：当某系的学生全部毕业还没有招生时，在删除全部学生记录的同时也删除了该系的信息。

（4）修改异常：更换系主任时需要变动较多的学生记录。

存在以上问题是因为在 SD 中存在非主属性对候选码的传递函数依赖，消除传递函数依赖就可转换为 3NF。

第三范式的规范化指将 2NF 关系模式通过投影分解，消除非主属性对候选码的传递函数依赖，转换成 3NF 关系模式的集合过程。

在分解时遵循“一事一地”的原则。

【例 3.4】　第三范式规范化举例。

将属于 2NF 的关系模式 SD(Sno, Sname, Age, Dept, DeptHead)分解为：

S (Sno, Sname, Age, Dept)

D (Dept, DeptHead)

分解后的函数依赖图如图 3.5 和图 3.6 所示。

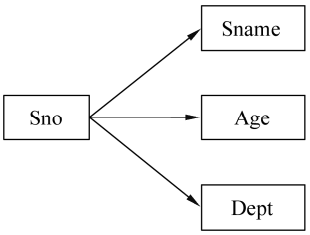

图 3.5　S 中的函数依赖图

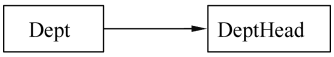

图 3.6　D 中的函数依赖图

分解后的关系模式 S 的候选码是 Sno，关系模式 D 的候选码是 Dept，不存在传递函数依赖，所以 S∈3NF，D∈3NF。

关系模式 SD 由 2NF 分解为 3NF 后，函数依赖关系变得更简单，既无主属性对候选码的部分依赖，又无主属性对候选码的传递依赖，解决了 2NF 存在的 4 个问题，3NF 的关系模式 S 和 D 的特点如下。

（1）降低了数据冗余度：系主任姓名存储的次数与该系的学生人数无关，只在关系 D 中存储一次。

（2）不存在插入异常：当一个新系没有招生时，该系的信息可直接插入到关系 D 中，与学生关系 S 无关。

（3）不存在删除异常：删除全部学生记录仍然保留该系的信息，可以只删除学生关系 S 中的记录，不影响关系 D 中的数据。

（4）不存在修改异常：更换系主任时只需修改关系 D 中一个相应元组的 DeptHead 属性值，不影响关系 S 中的数据。

由于 3NF 只限制了非主属性对码的依赖关系，未限制主属性对码的依赖关系，如果发生这种依赖，仍然可能存在数据冗余、插入异常、删除异常、修改异常，需要对 3NF 进一步规范化，消除主属性对码的依赖关系转换为更高一级的范式，这就是接下来要介绍的 BCNF 范式。

3.2.5　BCNF

定义 3.11　对于关系模式 R∈1NF，若 X→Y 且 Y ⊈ X 时 X 必含有码，则 R∈BCNF。

即若 R 中的每一决定因素都包含码，则 R∈BCNF。

由 BCNF 的定义可以得到如下结论，一个满足 BCNF 的关系模式有：

（1）所有非主属性对每一个码都是完全函数依赖。

（2）所有主属性对每一个不包含它的码也是完全函数依赖。

（3）没有任何属性完全函数依赖于非码的任何一组属性。

若 R∈BCNF，按定义排除了任何属性对码的部分依赖和传递依赖，所以 R∈3NF。但若 R∈3NF，则 R 未必属于 BCNF。

BCNF 的规范化指将 3NF 关系模式通过投影分解转换成 BCNF 关系模式的集合。

【例 3.5】 BCNF 范式规范化举例。

设有关系模式 SCN(Sno, Sname, Cno, Grade)，各属性的含义为学号、姓名、课程名、成绩，并假定姓名不重名。

可以看出，SCN 有两个码(Sno, Cno)和(Sname, Cno)，其函数依赖如下。

$Sno \leftrightarrow Sname$

$(Sno, Cno) \xrightarrow{p} Sname$

$(Sname, Cno) \xrightarrow{p} Sno$

唯一的非主属性 Grade 对码不存在部分依赖和传递依赖，所以 SCN∈3NF。但是，由于 $Sno \leftrightarrow Sname$，即决定因素 Sno 或 Sname 不包含码，从另一个角度看，存在主属性对码的部分依赖，即 $(Sno, Cno) \xrightarrow{p} Sname$，$(Sname, Cno) \xrightarrow{p} Sno$，所以 $SCN \notin BCNF$。

根据分解的原则，将 SCN 分解为以下两个关系模式：

S(Sno, Sname)

SC(Sno, Cno, Grade)

S 和 SC 的函数依赖图如图 3.7 和图 3.8 所示。

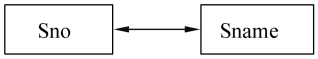

图 3.7　S 中的函数依赖图

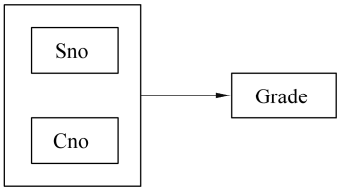

图 3.8　SC 中的函数依赖图

对于 S，两个候选码为 Sno 和 Sname，对于 SC，主码为(Sno,Cno)。在上述两个关系模式中，主属性和非主属性都不存在对码的部分依赖和传递依赖，所以 S∈BCNF，SC∈BCNF。

在关系 SCN 转换为 BCNF 后，数据冗余度明显降低，学生姓名只在关系 S 中存储一次，学生改名时只需改动一条学生记录中相应 Sname 的值即可，不会发生修改异常。

【例 3.6】 设有关系模式 STC(S, T, C)，其中 S 表示学生、T 表示教师、C 表示课程，语义假设是每一位教师只教一门课，每门课由多名教师讲授，某一学生选定某一门课程就对应一名确定的教师。

由语义假设，STC 的函数依赖如下。

$(S,C) \xrightarrow{f} T$，$(S,T) \xrightarrow{p} C$，$T \xrightarrow{f} C$

其中，(S, C)和(S, T)都是候选码。

函数依赖图如图 3.9 所示。

由于 STC 没有任何非主属性对码的部分依赖和传递依赖（因为 STC 没有非主属性），所以 STC∈3NF，但不是 BCNF，因为有 T→C，T 是决定因素，而 T 不包含候选码。

非 BCNF 关系模式分解为 ST(S, T)和 TC(T, C)，它们都是 BCNF。

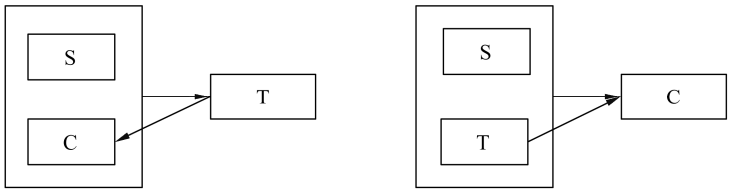

图 3.9　STC 中的函数依赖图

3.2.6　多值依赖与 4NF

函数依赖表示的关系模式中属性间是一对一或一对多的联系，不能表示属性间多对多的联系，本节讨论属性间多对多的联系（即多值依赖问题）以及第四范式。

1. 多值依赖

为了说明多值依赖的概念，举例如下。

【例 3.7】 设一门课程可由多名教师讲授，他们使用相同的一套参考书，可用如图 3.10 所示的非规范关系 CTR 表示课程 C、教师 T 和参考书 R 之间的关系。

课程C	教师T	参考书R
数据库原理与应用	刘俊松	数据库系统概念
	李智强	数据库系统概论
		SQL Server数据库教程
数学	罗燕芬	数学分析
	陈诗雨	线性代数

图 3.10　非规范关系 CTR

转换成规范化的关系 CTR(C, T, R)，如图 3.11 所示。

课程C	教师T	参考书R
数据库原理与应用	刘俊松	数据库系统概念
数据库原理与应用	刘俊松	数据库系统概论
数据库原理与应用	刘俊松	SQL Server数据库教程
数据库原理与应用	李智强	数据库系统概念
数据库原理与应用	李智强	数据库系统概论
数据库原理与应用	李智强	SQL Server数据库教程
数学	罗燕芬	数学分析
数学	罗燕芬	线性代数
数学	陈诗雨	数学分析
数学	陈诗雨	线性代数

图 3.11　规范后的关系 CTR

关系模式 CTR(C, T, R)的码是(C, T, R)，即全码，所以 CTR∈BCNF，但存在以下问题。

（1）数据冗余：课程、教师和参考书都被多次存储。

（2）插入异常：当课程“数据库原理与应用”增加一名讲课教师“周丽”时必须插入多个元组，即(数据库原理与应用, 周丽, 数据库系统概念)、(数据库原理与应用, 周丽, 数

据库系统概论)、(数据库原理与应用, 周丽, SQL Server 数据库教程)。

（3）删除异常：当课程“数学”要去掉一本参考书“数学分析”时必须删除多个元组，即(数学, 罗燕芬, 数学分析)、(数学, 陈诗雨, 数学分析)。

分析上述关系模式，发现存在一种称为多值依赖（Multi-Valued Dependency，MVD）的数据依赖。

定义 3.12 设 R(U)是属性集 U 上的一个关系模式，X、Y、Z 是 U 的子集，且 $Z=U-X-Y$。如果 R 的任一关系 r，对于给定的(X, Z)上的每一对值都存在一组 Y 值与之对应，且 Y 的这组值仅仅决定于 X 值而与 Z 的值不相关，则称 Y 多值依赖于 X，或 X 多值决定 Y，记为 $X\rightarrow\rightarrow Y$。

若 $X\rightarrow\rightarrow Y$，而 $Z=\phi$，则称 $X\rightarrow\rightarrow Y$ 为平凡的多值依赖，否则称 $X\rightarrow\rightarrow Y$ 为非平凡的多值依赖。

在上例的关系模式 CTR(C, T, R)中，对于给定的(C, R)的一对值（数据库原理与应用, 数据库系统概念），对应的一组 T 值为{刘俊松, 李智强}，这组值仅仅决定于 C 值。对于另一对值（数据库原理与应用，SQL Server 数据库教程），对应的一组 T 值仍为{刘俊松, 李智强}，尽管此时参考书 R 的值已改变，所以 T 多值依赖于 C，记为 C→→T。

2. 4NF

定义 3.13 设关系模式 R<U, F>∈1NF，如果对于 R 的每个非平凡多值依赖 $X\rightarrow\rightarrow Y(Y\not\subseteq X)$，$X$ 都含有码，则称 R<U, F>∈4NF。

由定义可知：

（1）根据定义，4NF 要求每一个非平凡的多值依赖 $X\rightarrow\rightarrow Y$，$X$ 都含有码，则必然是 $X\rightarrow Y$，所以 4NF 允许的非平凡多值依赖实际上是函数依赖。

（2）一个关系模式是 4NF，则必是 BCNF，而一个关系模式是 BCNF，不一定是 4NF，所以 4NF 是 BCNF 的推广。

例 3.7 的关系模式 CTR(C, T, R)是 BCNF，分解后产生 CTR1(C, T)和 CTR2(C, R)，因为 C→→T、C→→R 都是平凡的多值依赖，已不存在非平凡的非函数依赖的多值依赖，所以 CTR1∈4NF，CTR2∈4NF。

函数依赖和多值依赖是两种最重要的数据依赖。如果只考虑函数依赖，则属于 BCNF 的关系模式规范化程度已达到最高；如果只考虑多值依赖，则属于 4NF 的关系模式规范化程度已达到最高。在数据依赖中，除函数依赖和多值依赖以外还有其他数据依赖，例如连接依赖。函数依赖是多值依赖的一种特殊情况，而多值依赖又是连接依赖的一种特殊情况。如果消除了属于 4NF 的关系模式中存在的连接依赖，则可进一步达到 5NF 的关系模式，这里就不再讨论了。

3.2.7 规范化小结

关系模式规范化的目的是使结构更合理，消除插入异常、删除异常和更新异常，使数据冗余尽量小，便于插入、删除和更新。

关系模式规范化遵循“一事一地”的原则，即一个关系模式描述一个实体或实体间的一种联系。规范化的实质就是概念的单一化，方法是将关系模式投影分解为两个或两个以

上的模式。

一个关系模式只要其每一个属性都是不可再分的数据项，就称为 1NF；消除 1NF 中非主属性对码的部分函数依赖，得到 2NF；消除 2NF 中非主属性对码的传递函数依赖，得到 3NF；消除 3NF 中主属性对码的部分函数依赖和传递函数依赖，得到 BCNF；消除 BCNF 中非平凡且非函数依赖的多值依赖，得到 4NF，如图 3.12 所示。

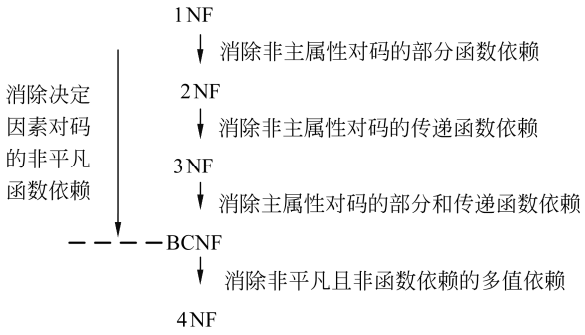

图 3.12　规范化过程

3.3　数据依赖的公理系统

数据依赖的公理系统是函数分解算法的理论基础，下面介绍的 Armstrong 公理系统是一个有效且完备的数据依赖公理系统。

3.3.1　Armstrong 公理系统

定义 3.14　对于满足一组函数依赖 F 的关系模式 R<*U*, F>，其任何一个关系 r，若函数依赖 $X\rightarrow Y$ 都成立（即 r 中的任意两元组 t、s，若 $t[X]=s[X]$，则 $t[Y]=s[Y]$），则称 F 逻辑蕴涵 $X\rightarrow Y$，或称 $X\rightarrow Y$ 是 F 的逻辑蕴涵。

那么怎样从一组函数依赖求得蕴涵的函数依赖？怎样求得给定关系模式的码？问题的关键在于已知一组函数依赖 F，问 $X\rightarrow Y$ 是否为 F 的逻辑蕴涵。这就需要一组推理规则，这组推理规则就是 Armstrong 公理系统。

Armstrong 公理系统（Armstrong's axiom）设 U 为属性集总体，F 是 U 上的一组函数依赖，有关系模式 R<*U*, F>，对于 R<*U*, F>来说有以下推理规则。

（1）自反律（reflexivity rule）：如果 $Y\subseteq X\subseteq U$，则 $X\rightarrow Y$ 为 F 所蕴涵。

（2）增广律（augmentation rule）：如果 $X\rightarrow Y$ 为 F 所蕴涵，且 $Z\subseteq U$，则 $XZ\rightarrow YZ$ 为 F 所蕴涵。

（3）传递律（transitivity rule）：如果 $X\rightarrow Y$ 及 $Y\rightarrow Z$ 为 F 所蕴涵，则 $X\rightarrow Z$ 为 F 所蕴涵。

提示：由自反律得到的函数依赖都是平凡的函数依赖，自反律的使用并不依赖于 F。

注意：XZ 表示 $X\cup Z$，YZ 表示 $Y\cup Z$。

定理 3.1　Armstrong 推理规则是正确的。

证明：

（1）设 $Y \subseteq X \subseteq U$，对于 R<*U*, F>的任一个关系中的任意两元组 *t*、*s*，若 *t*[*X*]=*s*[*X*]，因为 $Y \subseteq X$，有 *t*[*Y*]=*s*[*Y*]，所以 $X \rightarrow Y$。

（2）设 $X \rightarrow Y$ 为 F 所蕴涵，且 $Z \subseteq U$，设 *R*<*U*, F>的任一个关系中的任意两元组 *t*、*s*，若 *t*[*XZ*]=*s*[*XZ*]，有 *t*[*X*]=*s*[*X*]，*t*[*Z*]=*s*[*Z*]，由 $X \rightarrow Y$，有 *t*[*Y*]=*s*[*Y*]，所以 *t*[*YZ*]=*s*[*YZ*]，$XZ \rightarrow YZ$ 为 F 所蕴涵。

（3）设 $X \rightarrow Y$ 及 $Y \rightarrow Z$ 为 F 所蕴涵，对于 R<*U*, F>的任一个关系中的任意两元组 *t*、*s*，若 *t*[*X*]=*s*[*X*]，因为 $X \rightarrow Y$，有 *t*[*Y*]=*s*[*Y*]，由 $Y \rightarrow Z$，有 *t*[*Y*]=*s*[*Y*]，所以 $X \rightarrow Z$ 为 F 所蕴涵。

注意：*t*[*X*]表示元组 *t* 在属性组 *X* 上的分量，等价于 *t*.*X*。

根据（1）、（2）、（3）这 3 条推理规则可得以下 3 条推理规则。

（4）合并规则（union rule）：如果 $X \rightarrow Y$，$Y \rightarrow Z$，则 $X \rightarrow YZ$。

（5）分解规则（decomposition rule）：如果 $X \rightarrow Y$，Z ⊆Y，则 $X \rightarrow Z$。

（6）伪传递规则（pseudotransivity rule）：如果 $X \rightarrow Y$，$WY \rightarrow Z$，则 $XW \rightarrow Z$。

由合并规则和分解规则可得：

引理 3.1 $X \rightarrow A_1A_2 \cdots A_k$ 成立的充要条件是 $X \rightarrow A_i$ 成立（i=1,2,⋯,k）。

【例 3.8】 设有关系模式 R，*A*、*B*、*C*、*D*、*E*、*F* 是它的属性集的子集，R 满足的函数依赖为{$A \rightarrow BC$，$CD \rightarrow EF$}，证明函数依赖 $AD \rightarrow F$ 成立。

证明：

$A \rightarrow BC$	题中给定
$A \rightarrow C$	引理 3.1
$AD \rightarrow CD$	增广律
$CD \rightarrow EF$	题中给定
$AD \rightarrow EF$	传递律
$AD \rightarrow F$	引理 3.1

3.3.2 闭包及其计算

定义 3.15 在关系模式 R<*U*, F>中，为 F 所逻辑蕴涵的函数依赖的全体称为 F 的闭包（Closure），记为 F^+。

把自反律、增广律和传递律称为 Armstrong 公理系统。Armstrong 公理系统是有效的、完备的。其有效性是指由 F 出发根据 Armstrong 公理推导出来的每一个函数依赖一定在 F^+ 中；其完备性是指 F^+中的每一个函数依赖必定可以由 F 出发根据 Armstrong 公理推导出来。

如果要证明完备性，首先要解决如何判定一个函数依赖是否属于由 F 根据 Armstrong 公理推导出来的函数依赖的集合。如果能求出这个集合，也就解决了这个问题。但这是一个 NP 完全问题，例如从 F={ $X \rightarrow A_1, \cdots, X \rightarrow A_n$}出发至少可以推导出 2^n 个不同的函数依赖。为此引入以下概念。

定义 3.16 设 F 是属性集 *U* 上的一组函数依赖，*X*、*Y*⊆*U*，X_F^+={*A*|$X \rightarrow A$ 能由 F 根据 Armstrong 公理推导出}，X_F^+ 称为属性集 *X* 关于函数依赖集 F 的闭包。

由引理 3.1 可得出引理 3.2。

引理 3.2 设 F 是属性集 U 上的一组函数依赖，X、$Y \subseteq U$，$X \rightarrow Y$ 能由 F 根据 Armstrong 公理推导出的充分必要条件是 $Y \subseteq X_F^+$。

这样，判定 $X \rightarrow Y$ 能否由 F 根据 Armstrong 公理推导出的问题转化为求出 X_F^+，判定 Y 是否为 X_F^+ 的子集问题，该问题可由算法 3.1 解决。

算法 3.1 求属性集 $X(X \subseteq U)$关于 U 上的函数依赖集 F 的闭包 X_F^+。

输入：X、F。

输出：X_F^+。

步骤：计算属性集序列 $X^{(i)}$（i=0, 1, …）。

（1）令 $X^{(0)}=X$，i=0。

（2）求 B，$B=\{A \mid (\exists V)(\exists W)(V \rightarrow W \in F \wedge V \subseteq X^{(i)} \wedge A \in W)\}$。即在 F 中寻找尚未用过的左边是 $X^{(i)}$的子集的函数依赖：$Y_j \rightarrow Z_j$（j=0, 1, …, k），其中 $Y_j \subseteq X^{(i)}$。然后在 Z_j 中寻找 $X^{(i)}$ 中未出现过的属性构成属性集 B。

（3）$X^{(i+1)}=B \cup X^{(i)}$。

（4）判断 $X^{(i+1)}=X^{(i)}$是否成立，若不成立转（2）。

（5）输出 $X^{(i)}$，即为 X_F^+。

对于（4）的计算停止条件，以下 4 种方法是等价的。

- $X^{(i+1)}=X^{(i)}$。
- 当发现 $X^{(i)}$包含了全部属性时。
- 在 F 中的函数依赖的右边属性中再也找不到 $X^{(i)}$中未出现过的属性。
- 在 F 中未用过的函数依赖的左边属性集中已没有 $X^{(i)}$的子集。

【例 3.9】 已知关系模式 R<U, F>，其中 $U=\{A, B, C, D, E\}$，$F=\{AB \rightarrow C, B \rightarrow D, C \rightarrow E, EC \rightarrow B, AC \rightarrow B\}$，求 AB_F^+。

解：

（1）设 $X^{(0)}=AB$。

（2）在 F 中找出左边是 AB 子集的函数依赖，其结果是 $AB \rightarrow C$，$B \rightarrow D$，则 $X^{(1)}=X^{(0)} \cup CD=AB \cup CD = ABCD$，显然 $X^{(1)} \neq X^{(0)}$。

（3）在 F 中找出左边是 $ABCD$ 子集的函数依赖，其结果是 $C \rightarrow E$, $AC \rightarrow B$，则 $X^{(2)}=X^{(1)} \cup BE= ABCD \cup BE = ABCDE$，显然 $X^{(2)} \neq X^{(1)}$。

（4）由于 $X^{(2)}$已等于全部属性的集合，所以 $AB_F^+ = ABCDE$。

【例 3.10】 设有关系模式 R<U, F>，其中 $U=\{A, B, C, D, E, G\}$，函数依赖集 $F=\{A \rightarrow D, AB \rightarrow E, BG \rightarrow E, CD \rightarrow G, E \rightarrow C\}$，$X=AE$，计算 X_F^+。

解：

（1）设 $X^{(0)}=AE$。

（2）在 F 中找出左边是 AE 子集的函数依赖，其结果是 $A \rightarrow D$，$E \rightarrow C$，则 $X^{(1)}=X^{(0)}DC= ACDE$，显然 $X^{(1)} \neq X^{(0)}$。

（3）在 F 中找出左边是 $ACDE$ 子集的函数依赖，其结果是 $CD \rightarrow G$，则 $X^{(2)}=X^{(1)}G=ACDEG$。

（4）虽然 $X^{(2)} \neq X^{(1)}$，但 F 中未用过的函数依赖的左边属性集中已没有 $X^{(2)}$的子集，所以不必再计算下去，即 $X_F^+ = ACDEG$。

定理 3.2 Armstrong 公理是有效的、完备的。

Armstrong 公理的有效性可由定理 3.1 证明，对完备性的证明略。

Armstrong 公理的完备性及有效性说明“导出”与“蕴涵”是两个完全等价的概念，F^+也可以说成是由 F 出发根据 Armstrong 公理导出的函数依赖的集合。

3.3.3 确定候选码

设关系模式为 R<*U*, F>，F 为函数依赖集，将 *U* 中的属性分为以下 4 类。

- L 类属性：只在 F 中各个函数依赖的左部出现。
- R 类属性：只在 F 中各个函数依赖的右部出现。
- LR 类属性：在 F 中各个函数依赖的左部和右部都出现。
- N 类属性：不在 F 中的各个函数依赖中出现。

L 类属性集中的每一个属性必定是候选码中的属性，R 类和 N 类属性集中的每一个属性都必定不是候选码中的属性，LR 类属性集中的每一个属性不能确定是否在候选码中。

确定候选码的步骤如下。

（1）划分属性类别：令 *X* 为 L 类属性集的集合，*Y* 为 LR 类属性集的集合。

（2）基于 F 计算 X^+：若 X^+包含了 R 的全部属性，则 *X* 是 R 的唯一候选码，算法结束，否则转（3）。

（3）逐一取 *Y* 中的单一属性 *A*，与 *X* 组成属性组 *XA*，如果 $(XA)_F^+ = U$，则 *XA* 为候选码，令 *Y*=*Y*−{*A*}，转（4）。

（4）如果已找出所有候选码，转（5）；否则，依次取 *Y* 中的任意两个、3 个等属性，与 *X* 组成属性组 *XZ*，如果 $(XZ)_F^+ = U$，且 *XZ* 不包含已求得的候选码，则 *XZ* 为候选码。

（5）算法结束。

【例 3.11】 设 R(*A*, *B*, *C*, *D*, *E*, *F*)，*G*={*AB*→*E*, *AC*→*F*, *AD*→*B*, *B*→*C*, *C*→*D*}，求 R 的所有候选码。

解：

（1）R 中的 L 类属性：*A*；LR 类属性：*B*、*C*、*D*。

（2）$A_F^+ = A \neq U$

（3）因为 $(AB)_F^+ = ABCDEF$，所以 *AB* 为候选码。

因为 $(AC)_F^+ = ABCDEF$，所以 *AC* 为候选码。

因为 $(AD)_F^+ = ABCDEF$，所以 *AD* 为候选码。

故 R 的所有候选码为 *AB*、*AC*、*AD*。

3.3.4 函数依赖集的等价和最小函数依赖集

从蕴涵（或导出）的概念出发，引出两个函数依赖集的等价和最小函数依赖集的概念。

1. 两个函数依赖集的等价

定义 3.17 如果 $G^+ = F^+$，就说函数依赖集 F 覆盖 G（F 是 G 的覆盖，或 G 是 F 的覆盖），或 F 和 G 等价。

引理 3.3 $F^+ = G^+$的充分必要条件是 $F \subseteq G^+$和 $G \subseteq F^+$。

证明：必要性显然，这里只证充分性。

如果 $F \subseteq G^+$，则 $X_F^+ \subseteq X_{G^+}^+$，任取 $X \to Y \in F^+$，有 $Y \subseteq X_F^+ \subseteq X_{G^+}^+$，所以 $X \to Y \subseteq (G^+)^+ \subseteq G^+$，即 $F^+ \subseteq G^+$,同理可证 $G^+ \subseteq F^+$，所以 $F^+ = G^+$。

引理 3.3 给出了判定两个函数依赖集的等价的算法。

如果要判定 $F \subseteq G^+$，只需逐一对 F 中的函数依赖 $X \to Y$ 考查 Y 是否属于 G^+即可。

【例 3.12】 设有 F 和 G 两个函数依赖集，$F=\{A \to B, B \to C\}$，$G=\{A \to BC, B \to C\}$，判断它们是否等价。

解：

首先检查 F 中的每个函数依赖是否属于 G^+。

因为 $A_G^+ = ABC$，$B \subseteq A_G^+$，所以 $A \to B \in A_G^+$。

因为 $B_G^+ = BC$，$C \subseteq B_G^+$，所以 $B \to C \in B_G^+$。

故 $F \subseteq G^+$。

同样有 $G \subseteq F^+$，所以两个函数依赖集 F 和 G 是等价的。

2. 最小函数依赖集

定义 3.18 如果函数依赖集 F 满足以下条件，则称 F 为一个极小函数依赖集，也称为最小函数依赖集或最小覆盖（minimal cover）。

（1）F 中的任一函数依赖的右部仅含有一个属性。

（2）F 中不存在这样一个函数依赖 $X \to A$，X 有真子集 Z，使得 $F-\{X \to A\} \cup \{Z \to A\}$与 F 等价，即左部无多余的属性。

（3）F 中不存在这样一个函数依赖 $X \to A$，使得 F 与 $F-\{X \to A\}$等价，即无多余的函数依赖。

【例 3.13】 以下 3 个函数依赖集中哪一个是最小函数依赖集？

$F_1=\{A \to D, BD \to C, C \to AD\}$

$F_2=\{AB \to C, B \to A, B \to C\}$

$F_3=\{BC \to D, D \to A, A \to D\}$

解：

在 F_1 中有 $C \to AD$，即右部没有单一化，所以 F_1 不是最小函数依赖集。

在 F_2 中有 $AB \to C$，$B \to C$，即左部存在多余的属性，所以 F_2 不是最小函数依赖集。

F_3 满足最小函数依赖集的所有条件，它是最小函数依赖集。

【例 3.14】 在关系模式 R<U, F>中，U={Sno, Dept, DeptHead, Cno,Grade}，考查下面的函数依赖中哪一个是最小函数依赖集？

F={Sno→Dept, Dept→DeptHead, (Sno, Cno)→ Grade}

F_1={Sno→Dept, Sno→ DeptHead, Dept→DeptHead, (Sno, Cno)→ Grade, (Sno, Dept)→

Dept}

解：

F 是最小函数依赖集。

F_1不是最小函数依赖集，因为F_1−{ Sno→ DeptHead }与F_1等价，F_1−{ (Sno, Dept)→ Dept}与 F_1 等价。

定理 3.3 每一个函数依赖集 F 均等价于一个极小函数依赖集 F_m，此 F_m 称为 F 的最小依赖集。

证明：

这是一个构造性的证明，分 3 步对 F 进行“极小化”处理。

（1）逐一检查 F 中的各函数依赖 FD_i，使 F 中每一函数依赖的右部属性单一化。

$X\to Y$，若 $Y=A_1A_2\cdots A_k$，$k\geqslant 2$，则用$\{X\to A_j | (j=1,2,\cdots,k)\}$来取代 $X\to Y$。

（2）逐一取出 F 中的各函数依赖 FD_i，去掉各函数依赖左部多余的属性。

$X\to A$，设 $X=B_1B_2\cdots B_m$，$m\geqslant 2$，逐一考查 $B_i(j=1,2,\cdots,m)$，若 $B\in (X-B_i)_F^+$，则以 $X-B_i$ 取代 X。

（3）逐一检查 F 中的各函数依赖 FD_i，去掉多余的函数依赖。

$X\to A$，令 G=F−$\{X\to A\}$，若 $A\in X_G^+$，则从 F 中去掉此函数依赖。

F 的最小函数依赖集不一定是唯一的，它与对各函数依赖 FD_i 及 $X\to A$ 中 X 各属性的处理顺序有关。

【例 3.15】 求函数依赖集 F=$\{A\to B, B\to A, B\to C, A\to C, C\to A\}$的最小函数依赖集。

解：

下面给出 F 的两个最小函数依赖集。

$F_{m1}=\{A\to B, B\to C, C\to A\}$

$F_{m2}=\{A\to B, B\to A, A\to C, C\to A\}$

3.4 关系模式的分解

关系模式的分解过程就是将一个关系模式分解成一组等价的关系子模式的过程。对一个关系模式的分解可能有多种方式，但分解后产生的模式应与原来的模式等价。

3.4.1 模式分解的定义

定义 3.19 设有关系模式 R<U, F>，它的一个分解是指 ρ=$\{R_1<U_1, F_1>, R_2<U_2, F_2>, \cdots, R_n<U_n, F_n>\}$。

其中，$U=\bigcup_{i=1}^{n}U_i$，并且没有 $U_i\subseteq U_j$（$1\leqslant i、j\leqslant n$），$F_i$ 是 F 在 U_i 上的投影，并有 $F_i=\Pi_{R_i}(F)=\{X\to Y | X\to Y\in F^+ \wedge XY\subseteq U_i\}$。

对一个关系模式进行分解有多种方式，但分解后产生的模式应与原来的模式等价。由

“等价”的概念形成以下 3 种不同的定义。

- 分解具有无损连接性（lossless join）。
- 分解要保持函数依赖（preserve functional dependency）。
- 分解既要保持函数依赖，又要具有无损连接性。

3.4.2 分解的无损连接性

定义 3.20 设 $\rho=\{R_1<U_1, F_1>, R_2<U_2, F_2>, \cdots, R_n<U_k, F_k>\}$是关系模式 $R<U, F>$的一个分解，如果对于 R 的任一满足 F 的关系 r 都有 $r=\Pi_{R_1}(r)\bowtie\Pi_{R_2}(r)\bowtie\cdots\bowtie\Pi_{R_k}(r)$，则称这个分解 ρ 具有无损连接性，简称 ρ 为无损分解。

【例 3.16】 在关系 R(*A*, *B*, *C*)中函数依赖集为 F={*A*→*B*，*A*→*C*}，有两种分解方式，分解方式 1 为 $\rho_1=\{\Pi_{AB}(R),\Pi_{AC}(R)\}$，分解方式 2 为 $\rho_2=\{\Pi_{AB}(R),\Pi_{BC}(R)\}$，如图 3.13 所示。

R

A	*B*	*C*
3	1	2
1	3	1
2	3	3

(a) 关系 R

$R_1=\Pi_{AB}(R)$

A	*B*
3	1
1	3
2	3

$R_2=\Pi_{AC}(R)$

A	*C*
3	2
1	1
2	3

（b）分解方式 1

$R_1=\Pi_{AB}(R)$

A	*B*
3	1
1	3
2	3

$R_2=\Pi_{BC}(R)$

B	*C*
1	2
3	1
3	3

（c）分解方式 2

图 3.13 关系 R 中的两种分解方式

两种分解方式的自然连接结果如图 3.14 所示，可以看出，分解方式 1 是无损分解，分解方式 2 不是无损分解。

$\Pi_{AB}(R)\bowtie\Pi_{AC}(R)$

A	*B*	*C*
3	1	2
1	3	1
2	3	3

（a）分解方式 1 的自然连接

$\Pi_{AB}(R)\bowtie\Pi_{BC}(R)$

A	*B*	*C*
3	1	2
1	3	1
1	3	3
2	3	5
2	3	3

（b）分解方式 2 的自然连接

图 3.14 两种分解方式的自然连接结果

一般直接由定义判断一个分解是否为无损分解是不可能的，下面给出检验一个分解是否为无损分解的算法。

算法 3.2 检验无损连接性的算法。

输入：关系模式 R($A_1, A_2, \cdots, A_n$)，它的函数依赖集 F 以及分解 $\rho=\{R_1, R_2, \cdots, R_k\}$。

输出：确定 ρ 是否具有无损连接性。

步骤：

（1）构造一个 n 列 k 行的表，每一列对应于属性，每一行对应于分解中的一个关系模式，如果 $A_j \in R_i$，则第 j 列第 i 行上放符号 a_i，否则放符号 b_{ij}。

（2）逐个检查 F 中的每一个函数依赖，并修改表中的元素。取 F 中的一个函数依赖 $X \rightarrow Y$，在 X 的分量中寻找相同的行，然后将这些行中 Y 的分量改为相同的符号，如果其中有 a_j，将 b_{ij} 改为 a_j，如果其中无 a_j，则改为 b_{ij}。

（3）如果发现某一行变成了 $a_1, a_2, \cdots, a_n$，算法结束，分解 ρ 具有无损连接性；如果 F 中的所有函数依赖都不能再修改表中的内容，且没有发现这样的行，则分解 ρ 不具有无损连接性。

【例 3.17】 检验例 3.16 中关系 R(A, B, C)、函数依赖集 F=$\{A \rightarrow B, A \rightarrow C\}$的两种分解方式 $\rho_1=\{R_1=\Pi_{AB}(R), R_2=\Pi_{AC}(R)\}$、$\rho_2=\{R_1=\Pi_{AB}(R), R_2=\Pi_{BC}(R)\}$的无损连接性。

解：

1）对于分解方式 $\rho_1=\{R_1=\Pi_{AB}(R), R_2=\Pi_{AC}(R)\}$

（1）构造初始表：对于 R_1，包括 A、B 两个属性，第 1 行 A、B 列的值分别为 a_1、a_2，R_1 中没有 C 属性，该行 C 列的值为 $b_{1,3}$。对于 R_2，包括 A、C 两个属性，第 1 行 A、C 列的值分别为 a_1、a_3，R_2 中没有 B 属性，该行 B 列的值为 $b_{2,2}$，初始表如图 3.15（a）所示。

（2）检查 $A \rightarrow B$ 并对表中元素进行修改：检查 F 中的第 1 个函数依赖 $A \rightarrow B$，由于 R_1、R_2 的 A 列相同，所以将 R_2 的 B 列修改为 a_2（用粗体表示），如图 3.15（b）所示。因为第 2 行全为 a，所以 ρ_1 具有无损连接性。

R_i	A	B	C
AB	a_1	a_2	$b_{1,3}$
AC	a_1	$b_{2,2}$	a_3

（a）构造初始表

R_i	A	B	C
AB	a_1	a_2	$b_{1,3}$
AC	a_1	$\boldsymbol{a_2}$	a_3

（b）检查 $A \rightarrow B$ 并对表中元素进行修改

图 3.15　检验 ρ_1 的无损连接性

2）对于分解方式 $\rho_2=\{R_1=\Pi_{AB}(R), R_2=\Pi_{BC}(R)\}$

（1）构造初始表：对于 R_1，包括 A、B 两个属性，第 1 行 A、B 列的值分别为 a_1、a_2，R_1 中没有 C 属性，该行 C 列的值为 $b_{1,3}$。对于 R_2，包括 B、C 两个属性，第 1 行 B、C 列的值分别为 a_2、a_3，R_2 中没有 A 属性，该行 A 列的值为 $b_{2,1}$，初始表如图 3.16（a）所示。

（2）检查 $A \rightarrow B$、$A \rightarrow C$ 并对表中元素进行修改：检查 F 中的第 1 个函数依赖 $A \rightarrow B$，在表中找不到 A 列相同的行，对表中的元素值不做修改，如图 3.16（b）所示。检查 F 中的第 2 个函数依赖 $A \rightarrow C$，也找不到 A 列相同的行，不修改表中的元素值，如图 3.16（c）所示。因为没有全为 a 的行，所以 ρ_2 不具有无损连接性。

R_i	A	B	C
AB	a_1	a_2	$b_{1,3}$
BC	$b_{2,1}$	a_2	a_3

（a）构造初始表

R_i	A	B	C
AB	a_1	a_2	$b_{1,3}$
BC	$b_{2,1}$	a_2	a_3

（b）检查 $A\rightarrow B$ 并对表中元素进行修改

R_i	A	B	C
AB	a_1	a_2	$b_{1,3}$
BC	$b_{2,1}$	a_2	a_3

（c）检查 $A\rightarrow C$ 并对表中元素进行修改

图 3.16　检验 ρ_2 的无损连接性

3.4.3　分解的保持依赖性

保持关系模式等价的另一个重要条件是原模式所满足的函数依赖在分解后的模式中保持不变。

定义 3.21　若 $F^+=\left(\bigcup_{i=1}^{k}F_i\right)^+$，则 R<$U$, F>的分解 ρ={$R_1$<$U_1$, F_1>, R_2<U_2, F_2>, …, R_n<U_k, F_k>}保持函数依赖。

一个无损分解不一定具有函数依赖保持性；同样，一个依赖保持性分解不一定具有无损分解。

3.4.4　模式分解的算法

对于模式分解：

（1）若要求分解具有无损连接性，模式分解一定可达到 4NF。

（2）若要求分解保持函数依赖，模式分解可以达到 3NF，但不一定能达到 BCNF。

（3）若要求分解既要保持函数依赖，又要具有无损连结性，模式分解可以达到 3NF，但不一定能达到 BCNF。

算法 3.3　转换为 3NF 的保持函数依赖的分解。

步骤如下。

（1）求出 R<U, F>中函数依赖集 F 的最小函数依赖集 F_{min}。

（2）找出 F_{min} 中不出现的属性，把这样的属性构成一个关系模式。把这些属性从 U 中去掉，剩余的属性仍记为 U。

（3）若有 $X\rightarrow A$，且 $XA=U$，则输出 ρ={R}（即 R 也为 3NF，不用分解），算法终止。

（4）对 F_{min} 按具有相同左部的原则分组（假设分为 k 组），每一组函数依赖 F_i 所涉及的全部属性形成一个属性集 U_i，于是 ρ={R_1<U_1, F_1>, R_2<U_2, F_2>, …, R_k<U_k, F_k>}构成 R<U, F>的一个保持函数依赖的分解，R_i 均属 3NF。

（5）若 ρ 中没有一个子模式含 R 的候选码 X，则令 ρ=ρ∪{X}；若 $U_i\subseteq U_j$（$i\neq j$），去掉 U_i。

算法 3.4　转换为 3NF 既有无损连接性又保持函数依赖的分解。

步骤如下。

（1）根据算法 3.3 求出保持函数依赖的分解 ρ={R_1<U_1, F_1>, R_2<U_2, F_2>, …, R_k<U_k, F_k>}。

（2）选取 R 的主码 X，将主码与函数依赖相关的属性组成一个关系模式 R_{k+1}。

（3）如果 $X\subseteq U_i$，输出 ρ，否则输出 ρ∪{R_{k+1}}。

算法 3.5　转换为 BCNF 的无损连接分解。

步骤如下。

（1）令 ρ={R<*U*, F>}。

（2）如果 ρ 中的所有关系模式都是 BCNF，算法终止。

（3）如果 ρ 中有一个关系模式 R_i<U_i, F_i>不是 BCNF，则输出 R_i 中必须 $X \rightarrow A \in F_i^+$（$A$ 不属于 X），且 X 不是 R_i 的码。设 $S_1=XA$，$S_2=U_i-A$，用分解{S_1, S_2}代替 $R_i(U_i, F_i)$，返回步骤（2）。

3.5 小　　结

本章主要介绍了以下内容。

（1）关系数据库设计理论有 3 个方面的内容，即函数依赖、范式和模式设计。函数依赖起核心作用，它是模式分解和模式设计的基础，范式是模式分解的标准，关系数据库设计的关键是关系模式的设计

（2）函数依赖是关系数据库规范化理论的基础。

（3）在关系数据库的规范化过程中为不同程度的规范化要求设立的不同标准或准则称为范式。

一个低一级范式的关系模式通过模式分解可以转换成若干个高一级范式的关系模式的集合，该过程称为规范化。

关系模式规范化的目的是使结构更合理，消除插入异常、删除异常和更新异常，使数据冗余尽量小，便于插入、删除和更新。

一个关系模式只要其每一个属性都是不可再分的数据项，则称为 1NF；消除 1NF 中非主属性对码的部分函数依赖，得到 2NF；消除 2NF 中非主属性对码的传递函数依赖，得到 3NF；消除 3NF 中主属性对码的部分函数依赖和传递函数依赖，得到 BCNF；消除 BCNF 中非平凡且非函数依赖的多值依赖，得到 4NF。

（4）把自反律、增广律和传递律称为 Armstrong 公理系统。Armstrong 公理系统是有效的、完备的。其有效性是指由函数依赖集 F 出发根据 Armstrong 公理推导出来的每一个函数依赖一定在 F^+中；其完备性是指 F^+中的每一个函数依赖必定可以由 F 出发根据 Armstrong 公理推导出来。

在关系模式 R<*U*, F>中，为 F 所逻辑蕴涵的函数依赖的全体称为 F 的闭包（closure），记为 F^+。

从蕴涵（或导出）的概念出发，引出两个函数依赖集的等价和最小函数依赖集的概念。

（5）对一个关系模式进行分解有多种方式，但分解后产生的模式应与原来的模式等价。由“等价”的概念形成以下 3 种不同的定义：分解具有无损连接性（lossless join）、分解要保持函数依赖（preserve functional dependency）、分解既要保持函数依赖又要具有无损连接性。

习　题　3

一、选择题

3.1　在规范化过程中需要克服数据库逻辑结构中的冗余度大、插入异常和_____。

A．结构不合理　　B．删除异常

C．数据丢失　　D．数据的不一致性

3.2　关系规范化的插入异常是指_____。

A．不该删除的数据被删除　　B．应该删除的数据被删除

C．不该插入的数据被插入　　D．应该插入的数据未被插入

3.3　关系规范化的删除异常是指_____。

A．不该删除的数据被删除　　B．应该删除的数据被删除

C．不该插入的数据被插入　　D．应该插入的数据未被插入

3.4　在关系模式中，如果属性 A 和 B 存在 1:1 的联系，则说明_____。

A．A→B　　B．A↔B

C．B→A　　D．以上都不是

3.5　$X \rightarrow Y$，下列_____成立称为平凡函数依赖。

A．$Y \subset X$　　B．$X \subset Y$　　C．$X \cap Y = \phi$　　D．$X \cap Y \neq \phi$

3.6　下列说法中_____是错误的。

A．2NF 必然属于 1NF　　B．3NF 必然属于 2NF

C．3NF 必然属于 BCNF　　D．BCNF 必然属于 3NF

3.7　若关系模式 R(*A*, *B*)已属于 3NF，下列说法中正确的是_____。

A．一定消除了插入异常和删除异常　　B．仍存在一定的插入异常和删除异常

C．一定属于 BCNF　　D．A 和 C 都是

3.8　设有关系模式 R(*A*, *B*, *C*, *D*)，其数据依赖集 F={(*A*, *B*)→*C*, *C*→*D*}，则关系模式 R 的规范化程度最高达到_____。

A．1NF　　B．2NF　　C．3NF　　D．BCNF

3.9　在关系模式 S (Sno, Sname, Dept，DeptHead)中，各属性的含义为学号、姓名、系名、系主任姓名，S 的最高范式为_____。

A．1NF　　B．2NF　　C．3NF　　D．BCNF

3.10　设有关系模式 R(*A*, *B*, *C*, *D*, *E*)，其数据依赖集 F={*A*→*D*, *B*→*C*, *E*→*A*}，则关系模式 R 的候选码为_____。

A．*AB*　　B．*CD*　　C．*DE*　　D．*BE*

二、填空题

3.11　关系数据库设计理论有 3 个方面的内容，即函数依赖、范式和_____。

3.12　在关系数据库的规范化过程中为不同程度的规范化要求设立的不同_____称为范式。

3.13　一个低一级范式的关系模式通过_____可以转换成若干个高一级范式的关系模式的集合，该过程称为规范化。

3.14　关系模式规范化的目的是使结构更合理，消除插入异常、删除异常和_____，使数据冗余尽量小。

3.15　任何一个二目关系是属于_____的。

3.16　把自反律、_____和传递律称为 Armstrong 公理系统。

3.17　Armstrong 公理系统是有效的、_____的。

3.18　若 R.A→R.B，R.B→R.C，则_____。

3.19　若 R.A→R.B，R.A→R.C，则_____。

3.20　若 R.B→R.A，R.C→R.A，则_____。

三、问答题

3.21　什么是函数依赖？简述完全函数依赖、部分函数依赖和传递函数依赖。

3.22　什么是范式？什么是关系模式规范化？关系模式规范化的目的是什么？

3.23　简述关系模式规范化的过程。

3.24　简述 Armstrong 公理系统的推理规则。

3.25　什么是函数依赖集 F 的闭包？

3.26　什么是最小函数依赖集？简述求最小函数依赖集的步骤。

3.27　简述模式分解的定义。

四、应用题

3.28　设有关系模式 R(A, B, C, D)，其函数依赖集 F={CD→B, B→A}。

（1）说出 R 不是 3NF 的理由。

（2）将 R 分解为 3NF 的模式集。

3.29　设有关系模式 R(W, X, Y, Z)，其函数依赖集 F={X→Z, WX→Y}。

（1）R 属于第几范式。

（2）如果关系 R 不属于 BCNF，将 R 分解为 BCNF。

3.30　设有关系模式 R(A, B, C)，其函数依赖集 F={C→B，B→A}。

（1）求 R 的候选码。

（2）判断 R 是否为 3NF，并说出理由。

（3）如果不是，将 R 分解为 3NF 模式集。

3.31　设有关系模式 R(A, B, C, D)，其函数依赖集 F={A→C, C→A, B→AC, D→AC, BD→A}。

（1）计算$(AD)^+$。

（2）求 R 的候选码。

（3）求 F 的最小函数依赖集。

（4）将 R 分解为 3NF，使其既具有无损连接性又保持函数依赖。