

第6章 关系数据理论



本章目标

理解数据库规范化目的

“不好”的数据模式特点

掌握根据语义写出关系模式数据依赖的方法

理解和掌握函数依赖与多值依赖的概念及刻画工具

第一范式1NF

第二范式2NF

第三范式3NF

BCNF

第四范式4NF

熟练掌握求解给定关系模式所有候选码的方法

Armstrong公理系统

熟练掌握判定一个关系模式最高范式的方法

理解并掌握保持依赖与无损连接性概念

掌握多值依赖的判定方法

熟练掌握主要的模式分解方法



- 问题的提出
- 规范化
- 数据依赖的公理系统
- 保持函数依赖的模式分解
- 无损连接的模式分解*
- 本章小结



- 问题：什么是一个好的数据库逻辑设计？
- 什么是数据依赖？
- 关系模式的简化表示



■ 关系数据库逻辑设计

- 针对一个具体问题, 如何构造一个适合于它的数据模式, 即应该构造几个关系, 每个关系由哪些属性组成等
- 数据库逻辑设计的工具——关系数据库的规范化理论

■ 一个例子:

- 某个学校开发应用系统涉及的已知事实:
 - 一个学院有若干学生, 但一个学生只属于一个学院;
 - 一个学院只有一名(正职)院长;
 - 一个学生可以选修多门课程, 每门课程有若干学生选修;
 - 每个学生学习每一门课程有一个成绩。
- 为管理和存储上述信息, 最简单的做法就是设计一个单一关系模式**STUDENT**来实现
 - **STUDENT(Sno, School, Mname, Cno, Grade)** --School学院名, Mname院长名
 - **问题: 这个模式就是好的吗? 好模式的标准是什么? 应该如何度量?**



表6.1 Student表

Sno	School	Mname	Cno	Grade
S1	信息学院	张明	C1	95
S2	信息学院	张明	C1	90
S3	信息学院	张明	C1	88
S4	信息学院	张明	C1	70
S5	信息学院	张明	C1	78
:	:	:	:	:

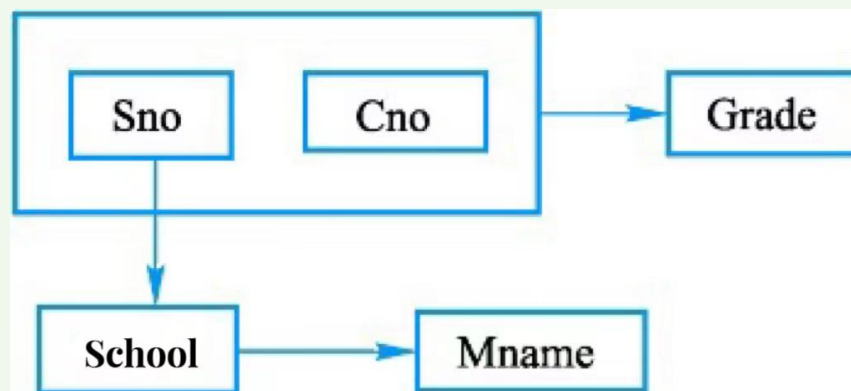
该模式包含了学院-学生,
学院-领导, 学生-课程
三种不同的业务逻辑

现象	特征	结论
<ul style="list-style-type: none"> 院长的姓名重复出现, 重复次数与该学院所有学生的所有课程成绩出现次数相同 	<ul style="list-style-type: none"> 数据冗余度大, 浪费存储空间 	Student模式不是一个好的关系模式, 好的关系模式不会发生增删改异常、冗余度尽可能小。 将使用规范化理论改造关系模式, 以消除存在的问题。
<ul style="list-style-type: none"> 如果某学院更换院长, 系统必须修改与该学院学生有关的每一个元组 	<ul style="list-style-type: none"> 更新异常 	
<ul style="list-style-type: none"> 如果新成立一个学院, 尚无学生, 我们就无法把这个学院及其院长的信息存入数据库 	<ul style="list-style-type: none"> 插入异常 	
<ul style="list-style-type: none"> 如果某个学院的学生全部毕业了, 我们在删除该学院学生信息的同时, 把这个学院及其院长的信息也丢掉了 	<ul style="list-style-type: none"> 删除异常 	



■ STUDENT(Sno, School, Mname, Cno, Grade)

- $U = \{ Sno, School, Mname, Cno, Grade \}$ -- U 为该模式的属性集合
- 借用数学上函数 $y=f(x)$ 的概念+上下文语义，可以有：
 - $School=f(Sno)$, Sno 函数确定 $School$ ，记为 $Sno \rightarrow School$
 - $Mname=f(School)$, $School$ 函数确定 $Mname$ ，记为 $School \rightarrow Mname$
 - $Grade=f((Sno, Cno))$, (Sno, Cno) 函数确定 $Grade$ ，记为 $(Sno, Cno) \rightarrow Grade$
- 将 F 记为属性组 U 上的函数依赖集合：
 - $F = \{ Sno \rightarrow School, School \rightarrow Mname, (Sno, Cno) \rightarrow Grade \}$



■ 1. 什么是数据依赖

- 属性值间的相互关联，通过属性间值的相等与否来描述，是现实世界属性间相互关联的抽象
- 是数据内在的性质，是语义的体现
- 数据库模式设计的关键

■ 2. 数据依赖的主要类型

- 函数依赖(functional dependency, 简记为FD)
- 多值依赖(multivalued dependency, 简记为MVD)
- 连接依赖...

■ 3. 数据依赖对关系模式的影响

- 不合适的数据依赖，造成插入异常、删除异常、更新异常和数据冗余问题

■ STUDENT(Sno ,School, Mname,Cno,Grade)模式存在部分和传递函数依赖

■ 如果解决关系模式中存在的问题？

- **规范化理论**—找出并消除关系模式中不合适的数据依赖，从而解决增删改异常和冗余度较大问题



■ 关系模式的形式化定义

$R(U, D, DOM, F)$ --五元组表示

- R : 关系名, 是符号化的元组语义
- U : 组成该关系模式的属性名集合
- D : U 中的属性所来自的域
- DOM : 属性向域的映射
- F : U 上的一组数据依赖



■ 关系模式的简化表示

$R(U, F)$ --三元组表示

- 由于 D 、 DOM 与模式设计关系不大, 因此在本章把关系模式表示为 $R(U, F)$
- 当且仅当 U 上的一个关系 r 满足 F 时, r 称为关系模式 $R(U, F)$ 的一个关系



- 规范化就是根据函数依赖分解一个不好的关系模式以达到消除更新异常、减少冗余的过程
- 本节主要内容：
 - 函数依赖
 - 码
 - 范式
 - 1NF
 - 2NF
 - 3NF
 - BCNF
 - 多值依赖
 - 4NF
 - 规范化小结



1. 函数依赖定义 6.1

- 设 $R(U)$ 是一个属性集 U 上的关系模式， X 和 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 $X \rightarrow Y$ 。
 - X 称为这个函数依赖的决定属性组，也称为 **决定因素 (determinant)**
- 若 $X \rightarrow Y$ 且 $Y \rightarrow X$ ，则记为 $X \leftrightarrow Y$ ；若 Y 不函数依赖于 X ，则记为 $X \nrightarrow Y$
- 例：STUDENT(Sno, School, Mname, Cno, Grade)
 - $F = \{Sno \rightarrow School, School \rightarrow Mname, (Sno, Cno) \rightarrow Grade\}$

Sno	School	Mname	Cno	Grade
S1	信息学院	张明	C1	95
S1	人工智能学院	张明	C1	90
S3	信息学院	张明	C1	88



• 该实例违背了 $Sno \rightarrow School$



2.如何确定函数依赖?

- 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。
 - 例如“姓名→年龄”这个函数依赖只有在“学生不允许有重名”的条件下成立
- 数据库设计者可以对现实世界作强制的规定。
 - 如设计者可以强行规定不允许学生有重名，因而使函数依赖“姓名→年龄”成立
- 函数依赖是指关系模式R在任何时刻的关系实例均要满足的约束条件。
 - 不是指某个或某些关系实例满足的约束条件，而是指R的所有关系实例均要满足的约束条件

Student	Course	Room
Mary	CS145	B01
Joe	CS145	B01
Sam	CS145	B01
..

• 该实例能否得出Course→Room成立?

• 只能说明可能成立，却无法证明一定成立!



EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876 ←	Salesrep
E1111	Smith	9876 ←	Salesrep
E9999	Mary	1234	Lawyer

{Position} → {Phone}

EmpID	Name	Phone	Position
E0045	Smith	1234 →	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234 →	Lawyer

but *not* {Phone} → {Position}

课堂练习:

A	B	C	D	E
1	2	4	3	6
3	2	5	1	8
1	4	4	5	7
1	2	4	3	6
3	2	5	1	8

找到至少3个与此实例相冲突的FDs:

{ } → { }
 { } → { }
 { } → { }



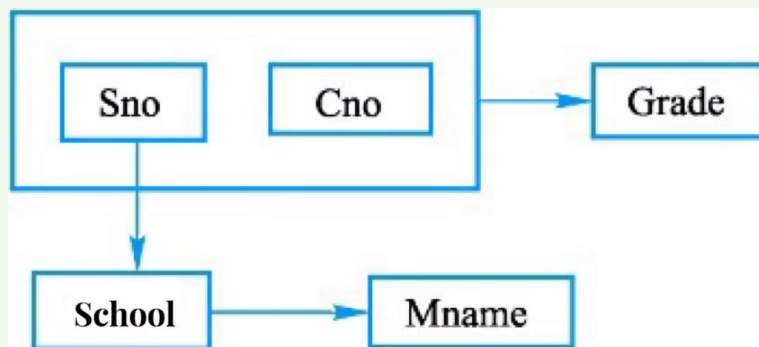
3. 平凡函数依赖与非平凡函数依赖

- $X \rightarrow Y$, 但 $Y \not\subseteq X$ 则称 $X \rightarrow Y$ 是非平凡的函数依赖
- $X \rightarrow Y$, 但 $Y \subseteq X$ 则称 $X \rightarrow Y$ 是平凡的函数依赖
 - 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, 因此若不特别声明, 我们总是讨论非平凡函数依赖
- 例: 在关系 $SC(Sno, Cno, Grade)$ 中,
 - $(Sno, Cno) \rightarrow Grade$ 为非平凡函数依赖
 - $(Sno, Cno) \rightarrow Sno$ 为平凡函数依赖
 - $(Sno, Cno) \rightarrow Cno$ 为平凡函数依赖



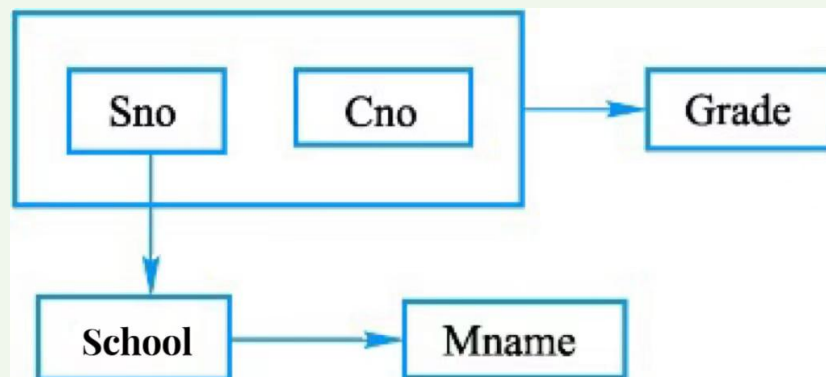
4.完全函数依赖与部分函数依赖

- **定义6.2** 在 $R(U, F)$ 中, 如果 $X \rightarrow Y$, 并且对于 X 的任何一个真子集 X' , 都有 $X' \nrightarrow Y$, 则称 Y 对 X **完全函数依赖**, 记作 $X \xrightarrow{F} Y$. 若 $X \rightarrow Y$, 但 Y 不完全函数依赖于 X , 则称 Y 对 X **部分函数依赖**, 记作 $X \xrightarrow{P} Y$
- 例: 在关系 $STUDENT(Sno, School, Mname, Cno, Grade)$ 中
 - $Sno \nrightarrow Grade, Cno \nrightarrow Grade, (Sno, Cno) \xrightarrow{F} Grade$
 - $(Sno, Cno) \xrightarrow{P} Sno, (Sno, Cno) \xrightarrow{P} Cno$



5. 传递函数依赖

- **定义6.3** 在 $R(U, F)$ 中, 如果 $X \rightarrow Y (Y \not\subseteq X)$, $Y \not\rightarrow X$, $Y \rightarrow Z$, $Z \not\subseteq Y$, 则称 Z 对 X 传递函数依赖, 记作 $X \xrightarrow{T} Z$
 - 如果 $Y \rightarrow X$, 即 $X \leftrightarrow Y$, 则 Z 直接依赖于 X
- 例: 在关系 $STUDENT(Sno, School, Mname, Cno, Grade)$ 中
 - $Sno \rightarrow School$, $School \rightarrow Mname$, $Mname$ 传递函数依赖于 Sno



- 码(key)也称键或键码，是关系模式中的一个重要概念
 - 用函数依赖的概念来定义码
- 定义6.4 设 K 为 $R\langle U, F \rangle$ 中的属性或属性组。若 $K \xrightarrow{F} U$ ，则 K 称为 R 的一个候选码(Candidate Key)
 - 如果 U 部分函数依赖于 K ，即 $K \xrightarrow{P} U$ ，则 K 称为超码(Superkey)
 - 候选码是最小的超码，即 K 的任意一个真子集都不是候选码
 - 注意：这里的“最小”不是指属性的个数为最少，而是指其真子集不能是候选码
 - 若关系模式 R 有多个候选码，则选定其中的一个做为主码
 - 整个属性组是码，称为全码 (All-key)
 - 问题：全码的所有属性都是主属性，反过来，所有属性都是主属性是否就是全码？



■ 主属性与非主属性

- 包含在任何一个候选码中的属性，称为主属性
- 不包含在任何码中的属性称为非主属性或非码属性

[例6.2] Student(Sno, Sname, Ssex, Sbirthdate, Smajor)

- Sno是码，Sno是主属性，Sname, Ssex, Sbirthdate, Smajor是非主属性

SC(Sno, Cno, Grade, Semester, Teachingclass)

- (Sno, Cno)是码，Sno, Cno是主属性，Grade, Semester, Teachingclass是非主属性

S(Sno, Sname, School, Sage), 假设学生无重名

- Sno, Sname是候选码，选定Sno为主码，Sno和Sname都是主属性，School和Sage都是非主属性，(Sno, School), (Sno, Sage), (Sno, School, Sage)都是超码。

[例6.3] 关系模式 R(P, W, A), P: 演奏者, W: 作品, A: 听众

- 语义：一个演奏者可以演奏多个作品，某一作品可被多个演奏者演奏，听众可以欣赏不同演奏者的不同作品
- R(P, W, A)码为(P, W, A)，即全码，All-Key



■ 外码

- **定义6.5** 关系模式R中属性或属性组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**也称**外码**
- **【例】** SC(Sno, Cno, Grade, Semester, Teachingclass)中，Sno不是码，但Sno是关系模式Student(Sno, Sname, Ssex, Sbirthdate, Smajor)的码，则Sno是关系模式SC的外码

■ 主码与外码一起提供了表示关系间联系的手段

■ 技能：如何根据已知的F, 求一个关系模式R的候选码？

- 简单情况下，可以**试凑**的方法
- 一般情况下，使用算法（见6.3节数据依赖的公理系统）



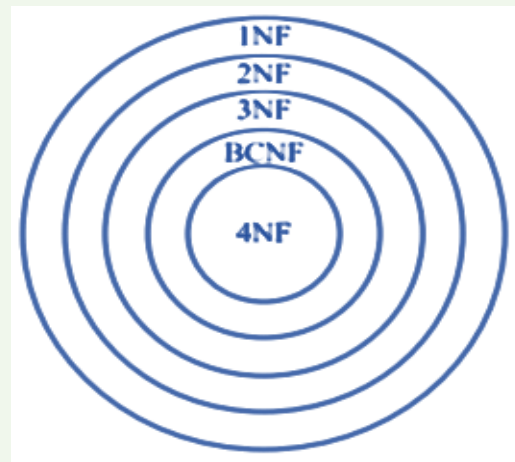
- **范式**是符合某一种级别的关系模式的集合

- 关系数据库中的关系必须满足一定的要求。满足不同程度要求的为不同范式

- 范式的种类及关系

- $1NF \supset 2NF \supset 3NF \supset BCNF \supset 4NF \supset 5NF$

- 某一关系模式R为第n范式，可简记为 $R \in nNF$



- 一个低一级范式的关系模式，通过**模式分解**可以转换为**若干个**高一级范式的关系模式的集合，这种过程就叫**规范化(normalization)**



- 1NF的定义

- 如果一个关系模式R的所有属性都是不可分的基本数据项, 则 $R \in 1NF$

- 第一范式是对关系模式的最起码的要求, 不满足第一范式的数据库模式不能称为关系数据模式

- 但是满足第一范式的关系模式并不一定是一个好的关系模式

- [例6.4] 关系模式S-L-C(Sno, School, Sloc, Cno, Grade), Sloc为学生住所, 假设每个学院的学生住在同一个宿舍楼
函数依赖有:

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

$Sno \rightarrow School, (Sno, Cno) \xrightarrow{P} School$

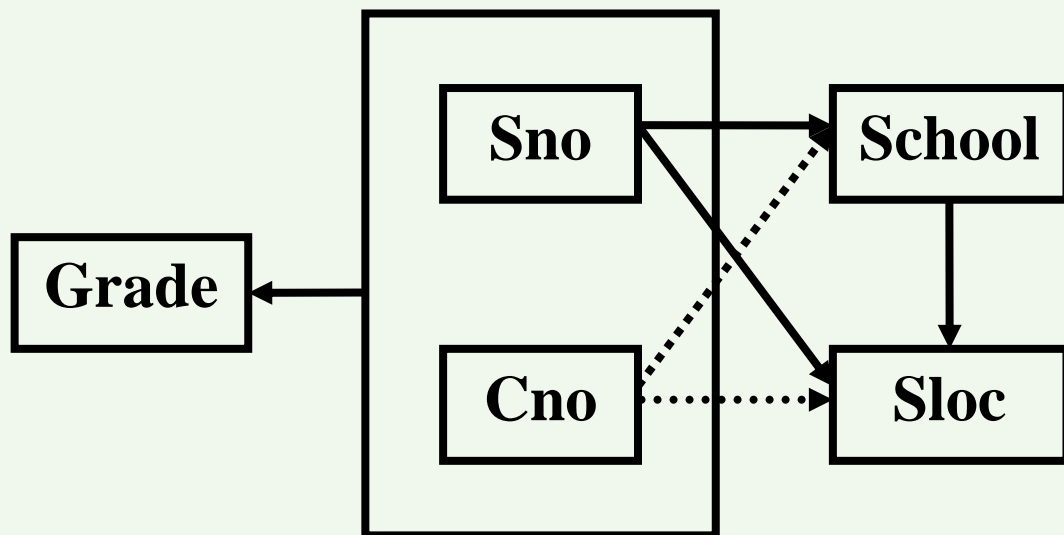
$Sno \rightarrow Sloc, (Sno, Cno) \xrightarrow{P} Sloc$

$School \rightarrow Sloc$

▪ S-L-C的码为(Sno, Cno)

S-L-C的主属性: Sno, Cno; 非主属性: School, Sloc, Grade





结论:

- $S-L-C \in 1NF$
- 非主属性School, Sloc并不完全依赖于码
- 关系模式 $S-L-C \notin 2NF$

■ 思考: 属于1NF的关系模式是好的吗? 会存在哪些问题?



■ S-L-C存在的问题

– 1. 插入异常

- 假设Sno = S7, School = INF, Sloc = BLD2的学生还未选课, 因课程号是主属性, 因此该学生的信息无法插入S-L-C

Sno	School	Sloc	Cno	Grade
S1	INF	BLD2	C3	89
S2	INF	BLD2	C2	97
S3	INF	BLD2	C5	88
S4	INF	BLD2	C1	86
S5	INF	BLD2	C3	92
S6	INF	BLD2	C3	79
S7	INF	BLD2	null	null



■ S-L-C存在的问题

– 2. 删除异常

- 假定S4学生只选修了C3课程这一门课。现在因身体不适，他连C3课程也不选修了。因课程号是主属性，此操作将导致该学生信息的整个元组都要删除

Sno	School	Sloc	Cno	Grade
S1	INF	BLD2	C3	89
S2	INF	BLD2	C2	97
S3	INF	BLD2	C5	88
S4	INF	BLD2	C3	86
S5	INF	BLD2	C3	92
S6	INF	BLD2	C3	79



■ S-L-C存在的问题

– 3. 修改复杂

- 例如学生从人工智能学院转到信息学院，在修改此学生元组的**School**值的同时，还可能需要修改住处(**Sloc**)。如果这个学生选修了**K**门课，则必须无遗漏地修改**K**个元组中全部**School**、**Sloc**信息

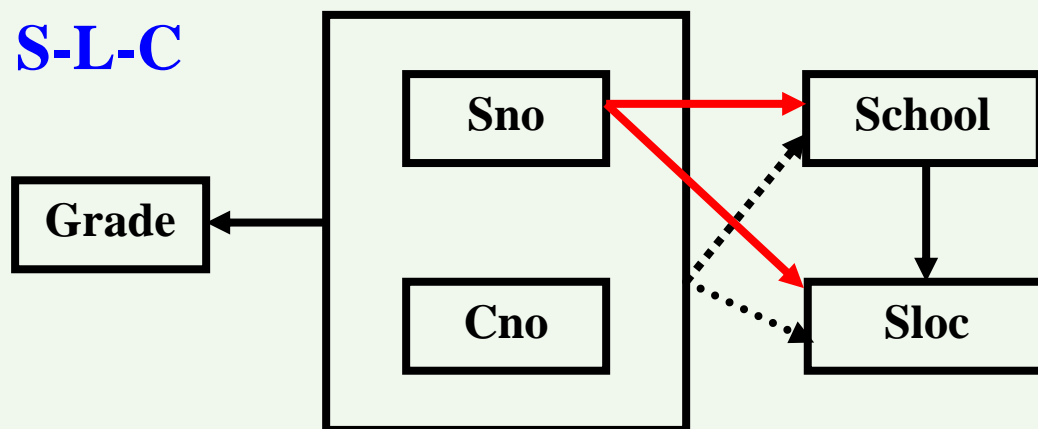
Sno	School	Sloc	Cno	Grade
S1	INF	BLD2	C3	89
S2	INF	BLD2	C2	97
S3	INF	BLD2	C5	88
S4	INF	BLD2	C1	86
S5	INF	BLD2	C3	92
S6	INF	BLD2	C3	79
S7	INF	BLD2	C1	72
S8	INF	BLD2	C4	65
S9	INF	BLD2	C6	99
S10	INF	BLD2	C7	83
S11	INF	BLD2	C8	75



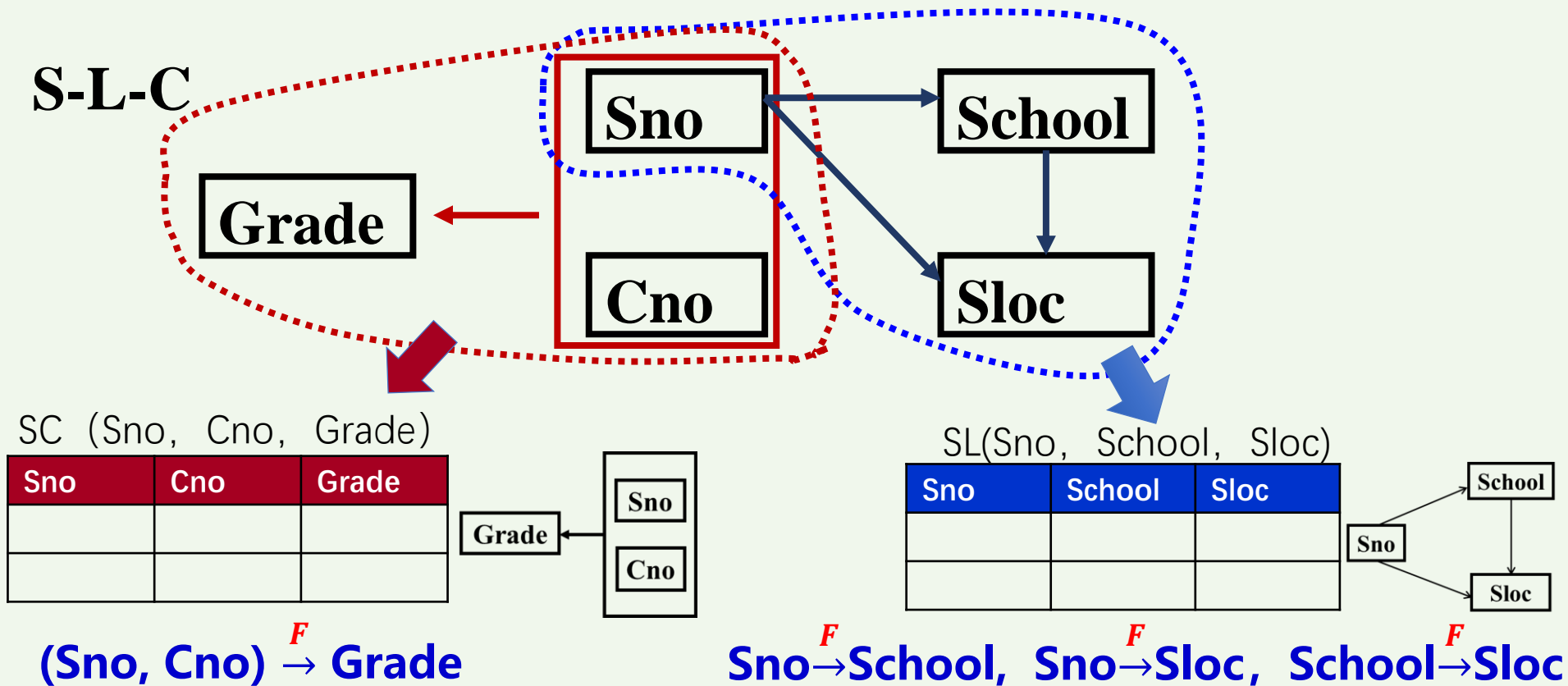
Sno	School	Sloc	Cno	Grade
S1	AI	BLD1	C3	89
S2	AI	BLD1	C2	97
S3	AI	BLD1	C5	88
S4	INF	BLD2	C1	86
S5	INF	BLD2	C3	92
S6	INF	BLD2	C3	79
S7	AI	BLD1	C1	72
S8	AI	BLD1	C4	65
S9	AI	BLD1	C6	99
S10	AI	BLD1	C7	83
S11	AI	BLD1	C8	75



- 结论：S-L-C不是一个好的关系模式
- 原因：
 - S-L-C(Sno, School, Sloc, Cno, Grade) 中School、Sloc部分函数依赖于码。S-L-C的码为(Sno, Cno)



- 解决方法：
 - 采用投影分解法，把S-L-C分解为两个关系模式，以消除这些部分函数依赖



- 关系模式SC的码为(Sno, Cno), 关系模式SL的码为Sno
- 非主属性对码都是完全函数依赖的
- 从而使上述四个问题在一定程度上得到了一定的解决

SC

Sno	Cno	Grade

SL

Sno	School	Sloc

1. 由于学生选修课程的情况与学生的基本情况是分开存储在两个关系中的，在SL关系中可以插入尚未选课的学生
2. 删除一个学生的所有选课记录，只是SC关系中没有关于该学生的记录了，SL关系中关于该学生的记录不受影响
3. 不论一个学生选多少门课程，他的School和Sloc值都只存储1次。这就大大降低了数据冗余
4. 学生转学院只需修改SL关系中该学生元组的School值和Sloc值，由于School、Sloc并未重复存储，因此减化了修改操作



■ 2NF定义6.6

– 若 $R \in 1NF$ ，且每一个非主属性完全函数依赖于任何一个候选码，则 $R \in 2NF$

■ 例：

– $S-L-C(Sno, School, Sloc, Cno, Grade) \in 1NF$

– $S-L-C \notin 2NF$

– $SC(Sno, Cno, Grade) \in 2NF$

– $SL(Sno, Sdept, Sloc) \in 2NF$

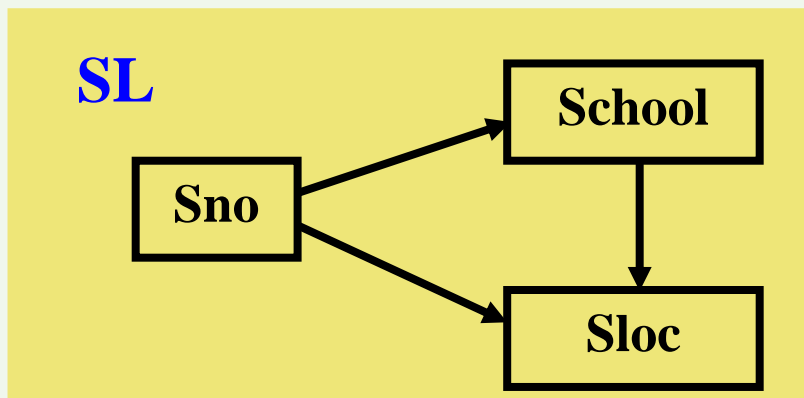
- 采用投影分解法将一个1NF的关系分解为多个2NF的关系，可以在一定程度上减轻原1NF关系中存在的插入异常、删除异常、数据冗余度大、修改复杂等问题，但并不能完全消除关系模式中的各种异常情况和数据冗余



■ 例：2NF关系模式SL(Sno, School, Sloc)中

– 函数依赖：

- $Sno \rightarrow School$
- $School \rightarrow Sloc$
- $Sno \rightarrow Sloc$



Sloc传递函数依赖于Sno，即SL中存在非主属性对码的传递函数依赖



■ SL关系存在的问题:

– 1.插入异常

- 如果某个学院因种种原因（例如刚刚成立），目前暂时没有在校学生，我们就无法把这个学院的信息，如MA, Sloc，存入数据库

Sno	School	Sloc
S1	INF	N
S2	INF	N
S3	INF	N
S4	INF	N
null	MA	null




■ SL关系存在的问题:

– 2.删除异常

- 如果某个学院的学生全部毕业了，我们在删除该学院学生信息的同时，把这个学院的信息，如INF, Sloc，也丢掉了

Sno	School	Sloc
S1	INF	N
S2	INF	N
S3	INF	N
S4	INF	N
S5	AI	S
S6	AI	S



Sno	School	Sloc
S5	AI	S
S6	AI	S



■ SL关系存在的问题:

– 3. 修改复杂

- 学校调整学生住处时，由于关于每个学院的住处信息是重复存储的，修改时必须同时更新该学院所有学生的Sloc属性值

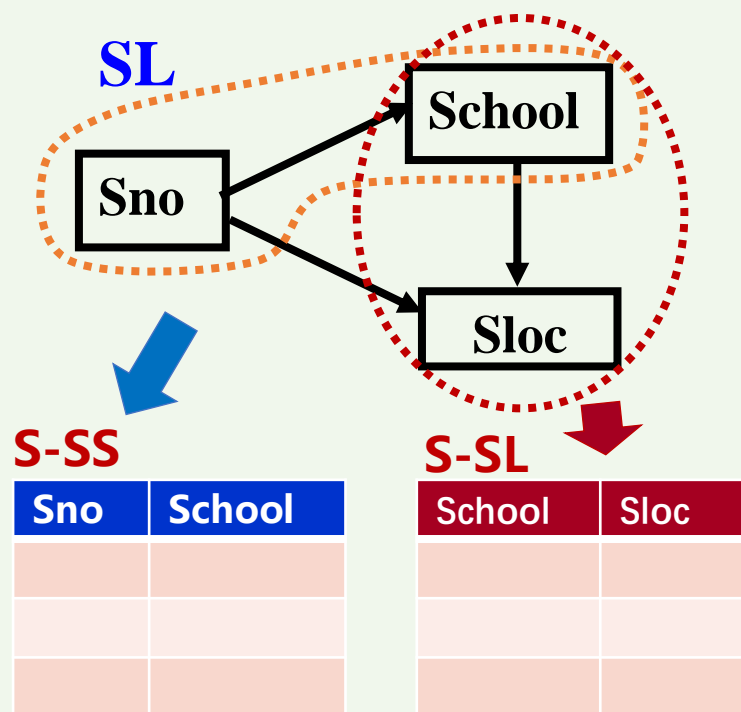
Sno	School	Sloc
S1	INF	N
S2	INF	N
S3	INF	N
S4	INF	N
S5	AI	S
S6	AI	S
S7	AI	S
S8	AI	S
.....



Sno	School	Sloc
S1	INF	S
S2	INF	S
S3	INF	S
S4	INF	S
S5	AI	S
S6	AI	S
S7	AI	S
S8	AI	S
.....



- **结论：**SL仍然不是一个好的关系模式
- **原因：**SL中Sloc传递函数依赖于Sno
- **解决方法：**采用投影分解法，把SL分解为两个关系模式，以消除传递函数依赖
 - S-SS(Sno, School), 码为Sno
 - S-SL(School, Sloc), 码为School
- 在分解后的关系模式中既没有非主属性对码的部分函数依赖，也没有非主属性对码的传递函数依赖，在一定程度上解决了上述四个问题



S-SS

Sno	School

S-SL

School	Sloc

1. S-SL关系中可以插入无在校学生的学院的信息
2. 某个学院的学生全部毕业了，只是删除S-SS关系中的相应元组，S-SL关系中关于该学院的信息仍存在
3. 关于学院的住处的信息只在S-SL关系中存储一次
4. 当学校调整某个学院的学生住处时，只需修改S-SL关系中一个相应元组的Sloc属性值



■ 课堂练习:

- 设有关系模式 $R\langle U, F \rangle$, $U = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9\}$, $F = \{(A_1, A_2) \rightarrow (A_5, A_6), A_1 \rightarrow A_3, A_2 \rightarrow (A_4, A_7, A_8, A_9), A_8 \rightarrow A_9\}$, 其中 (A_1, A_2) 为主码
- 请判断 R 的最高范式。如果可能请将 R 分解到高级范式

参考答案:

- R 的最高范式为 1NF

- 把部分依赖独立出去:

$R1(\{A_1, A_2, A_5, A_6\}, (A_1, A_2) \rightarrow (A_5, A_6)),$

$R2(\{A_1, A_3\}, A_1 \rightarrow A_3),$

$R3(\{A_2, A_4, A_7, A_8, A_9\}, A_2 \rightarrow (A_4, A_7, A_8, A_9), A_8 \rightarrow A_9)$



■ 3NF 定义6.7:

- 设关系模式 $R\langle U, F \rangle \in 1NF$, 若 R 中不存在这样的码 X 、属性组 Y 及非主属性 $Z (Y \not\supset Z)$, 使得 $X \rightarrow Y (Y \not\rightarrow X)$, $Y \rightarrow Z$ 成立, 则称 $R\langle U, F \rangle \in 3NF$

■ 【前例】

- $SL(Sno, School, Sloc) \in 2NF$

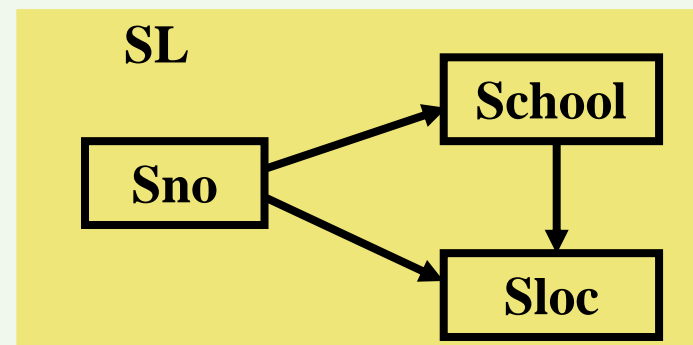
- $SL \notin 3NF$

- 因为存在非主属性 $Sloc$ 对码 Sno 的传递函数依赖

采用投影分解法将 SL 分解为 $S-SS$ 模式和 $S-SL$ 模式

- $S-SS(Sno, School) \in 3NF$

- $S-SL(School, Sloc) \in 3NF$



- 若 $R \in 3NF$ ，则 R 的每一个非主属性既不部分函数依赖于候选码也不传递函数依赖于候选码
- 如果 $R \in 3NF$ ，则 $R \in 2NF$ (教材习题8之(2))
- 采用投影分解法将一个 $2NF$ 的关系分解为多个 $3NF$ 的关系，可以在一定程度上解决原 $2NF$ 关系中存在的插入异常、删除异常、数据冗余度大、修改复杂等问题
- 将一个 $2NF$ 关系分解为多个 $3NF$ 的关系后，**并不能完全消除**关系模式中的各种异常情况和数据冗余
 - 如果可能还可以分解为高一级的 $BCNF$



■ 范式判断及分解示例：

R0{Suppl#, SupplName, Item#, ItemName, Quantity, SupplStatus, Location} PK
[Suppl#, Item#]
[Suppl#, Item#] \rightarrow {Quantity, SupplName, SupplStatus, Location, ItemName}
Suppl# \rightarrow {SupplName, SupplStatus, Location}
Item# \rightarrow ItemName



R1{Suppl#, SupplName, Location, SupplStatus} PK[Suppl#]
R2{Item#, ItemName} PK [Item#]
R3{Suppl#, Item#, Quantity} PK[Suppl#, Item#]

- 根据范式定义分析，**R1, R2, R3**都属于**3NF**

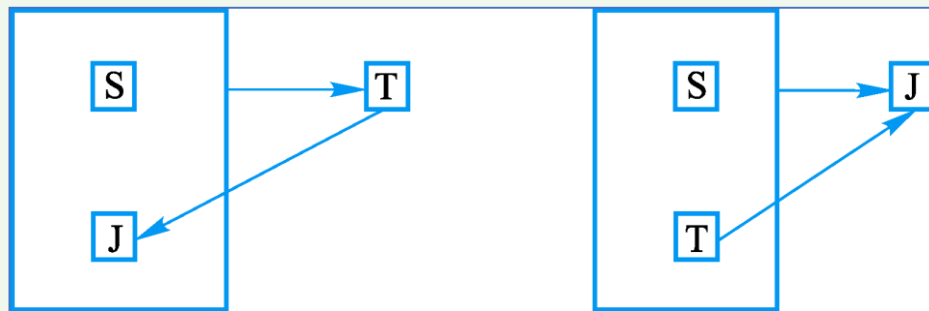


■ BCNF(Boyce Codd Normal Form)

- 由Boyce和Codd提出，比3NF更进了一步。通常认为BCNF是修正的第三范式，有时也称为扩充的第三范式

[例6.8] 关系模式STJ(S, T, J)中，S表示学生，T表示教师，J表示课程。每一教师只教一门课，每门课有若干教师，某一学生选定某门课就对应一个固定的教师。得到的函数依赖：

- $T \rightarrow J$, $(S, J) \rightarrow T$, $(S, T) \rightarrow J$



- (S, J) 和(S, T)都是候选码
- $STJ \in 3NF$
- $T \rightarrow J$, T是主属性，不是候选码



■ STJ(S, T, J)存在的问题:

– 插入异常

- 如果某个教师开设了某门课程，但尚未有学生选修，则有关信息也无法存入数据库中

– 删除异常

- 如果选修过某门课程的学生全部毕业了，在删除这些学生元组的同时，相应教师开设该门课程的信息也同时丢掉了

– 数据冗余度大

- 虽然一个教师只教一门课，但每个选修该教师该门课程的学生元组都要记录这一信息

– 修改复杂

- 某个教师开设的某门课程改名后，所有选修了该教师该门课程的学生元组都要进行相应修改

■ 因此虽然 $STJ \in 3NF$ ，但它仍存在增删改等异常，还不是一个理想的关系模式



- 原因：
 - 主属性J依赖于T，即主属性J部分依赖于码(S, T)
- 解决方法：
 - 采用投影分解法，将STJ分解为两个关系模式：
 - ST(S, T)，码为(S, T)
 - TJ(T, J)，码为T
 - 注：该分解也可用后面的BCNFDecomp算法验证



- 在分解后的关系模式中没有任何属性对码的部分函数依赖和传递函数依赖，它解决了上述四个问题：
 1. TJ关系中可以存储所开课程尚未有学生选修的教师信息
 2. 选修过某门课程的学生全部毕业了，只是删除ST关系中的相应元组，不会影响TJ关系中相应教师开设该门课程的信息
 3. 关于每个教师开设课程的信息只在TJ关系中存储一次
 4. 某个教师开设的某门课程改名后，只需修改TJ关系中的一个相应元组即可



■ BCNF的定义6.8:

- 设关系模式 $R\langle U, F \rangle \in 1NF$, 若 $X \rightarrow Y$ 且 $Y \not\subseteq X$ 时 X 必含有码, 则 $R\langle U, F \rangle \in BCNF$
 - 即, 在关系模式 $R\langle U, F \rangle$ 中, 如果每一个决定属性集都包含候选码, 则 $R \in BCNF$

■ 例子:

- $STJ(S, T, J) \in 3NF$, $SJ(S, J) \in BCNF$, $TJ(T, J) \in BCNF$

■ BCNF的关系模式具有的性质:

- 所有非主属性都完全函数依赖于每个候选码
- 所有主属性都完全函数依赖于每个不包含它的候选码
- 没有任何属性完全函数依赖于非码的任何一组属性



- 关于BCNF, 3NF, 2NF, 1NF之间关系的结论:
 - $R \in \text{BCNF} \Rightarrow R \in \text{3NF} \Rightarrow R \in \text{2NF} \Rightarrow R \in \text{1NF}$, 反之则不然 (试给出反例)
- 采用投影分解法将一个3NF的关系分解为多个BCNF的关系, 可以进一步解决原3NF关系中存在的插入异常、删除异常、数据冗余度大、修改复杂等问题
- 如果一个关系数据库中的所有关系模式都属于BCNF, 那么在函数依赖范畴内, 它已实现了模式的彻底分解, 达到了最高的规范化程度, 消除了插入异常和删除异常



[例6.5] 分析关系模式**Course(Cno, Cname, Ccredit, Cpno)**的最高范式

- 只有唯一的码：**Cno**
- 没有任何属性对**Cno**部分依赖或传递依赖，所以**Course \in 3NF**
- **Course**中**Cno**是唯一的决定因素，所以**Course \in BCNF**

[例6.6] 分析关系模式**Student(Sno, Sname, Ssex, Sbirthdate, Smajor)**的最高范式

- 假设**Sname**唯一，**Student**有两个候选码，因都是单属性构成，所以彼此不相交。其它属性不存在对码的传递与部分函数依赖，所以**Student \in 3NF**。同时**Student**中除**Sno, Sname**外没有其它决定因素，所以**Student \in BCNF**

[例6.7] 在**SJP(S,J,P)**中，**S**是学生，**J**表示课程，**P**表示名次。每一个学生选修每门课程的成绩有一定的名次，每门课程中每一名次只有一个学生(即没有并列名次)

- 由语义可得到函数依赖：**(S, J) \rightarrow P, (J, P) \rightarrow S**
- **(S, J)**与**(J, P)**都可以作为候选码
- 关系模式中**没有非主属性**对码传递依赖或部分依赖，所以**SJP \in 3NF**
- 除**(S,J)**与**(J,P)**以外没有其它决定因素，所以**SJP \in BCNF**



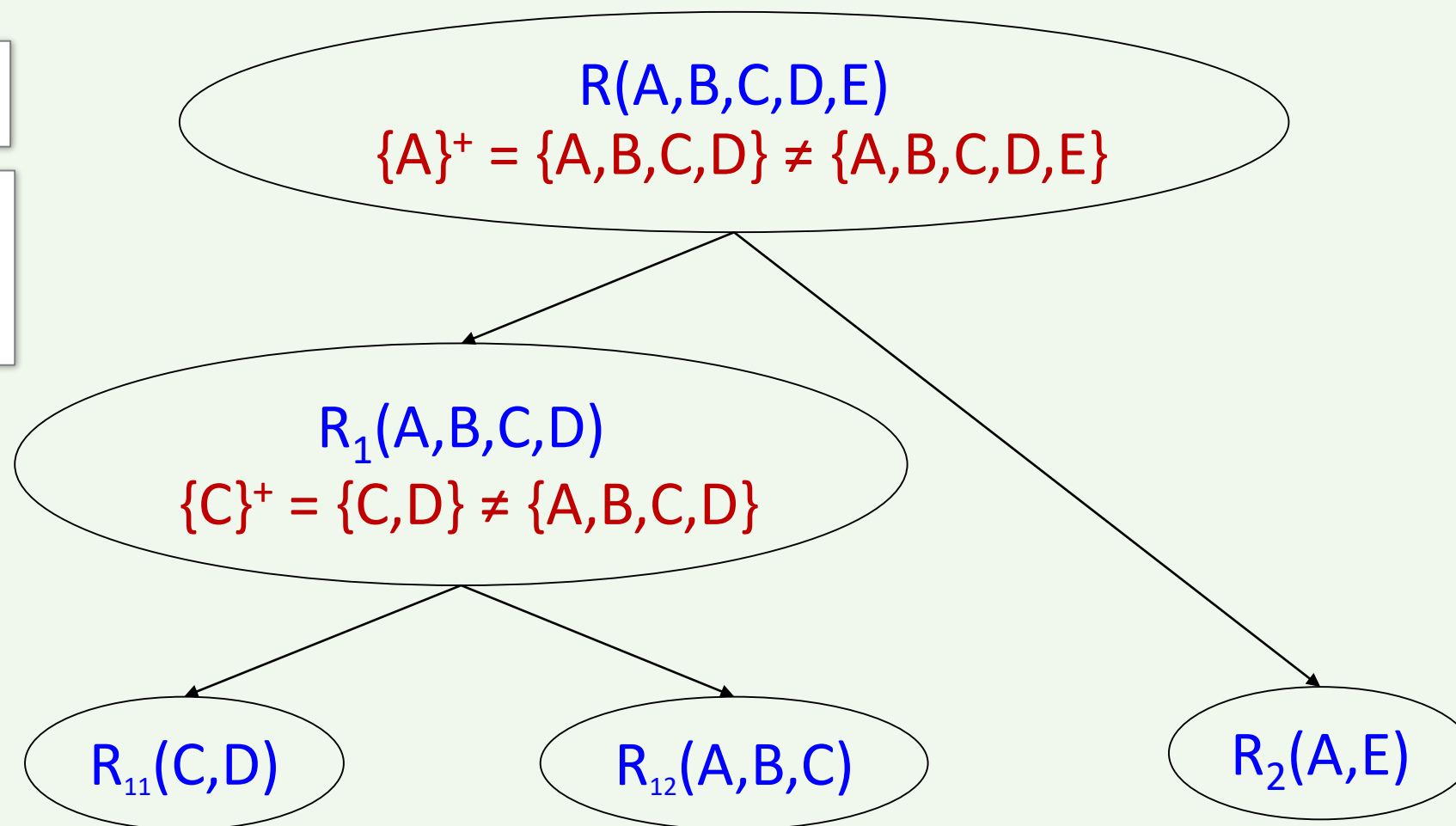
- 分解成BCNF的一种算法：
 - 假设关系模式 $R\langle U, F \rangle$ ，分解算法BCNFDecomp(R):
 - 找到属性(组) X ，满足 $X^+ \neq X$ ， $X^+ \neq U$
 - 如果没找到，则返回 R
 - 否则，令 $Y = X^+ - X$ ， $Z = (X^+)^c$ ，则 R 分解为 $R_1(XUY)$ ， $R_2(XUZ)$
- 此算法能够保持无损连接性
 - [例6.8]的关系模式STJ分解成BCNF的结果也可用该算法得到



【例题】：设有关系模式 $R(A, B, C, D, E)$, $F=\{A \rightarrow (B, C), C \rightarrow D\}$, 试判定 R 的最高范式。如果 R 不是BCNF, 则将之分解到BCNF.

- **【解】**：通过分析可知, **AE**为 R 的主码(唯一候选码), 且**存在非主属性 (B,C) 对码AE的部分函数依赖**, 所以 R 的最高范式为**1NF**.
 - 根据上面的分析, 可以从**C**处断开, 变成 **$R_1(AE)$, $R_2(ABC)$, $R_3(CD)$** 。由定义知, R_1, R_2, R_3 都达到3NF, 进一步分析, 它们也都属于BCNF.
- **第二种方法**：利用BCNF分解算法, 可得到 R 如下的一种分解结果.



$R(A,B,C,D,E)$ $\{A\} \rightarrow \{B,C\}$ $\{C\} \rightarrow \{D\}$ 

■ 传统方法(Bottom-top)

1. 找出F;
2. 验证R是否符合2NF定义。若否, $R \in 1NF$; 若是, 执行步骤3;
3. 验证R是否符合3NF定义。若否, $R \in 2NF$; 若是, 转向执行步骤4;
4. 验证R是否符合BCNF定义。若否, $R \in 3NF$; 若是, $R \in BCNF$

■ 新方法(Top-bottom)

- 依据: $1NF \supset 2NF \supset 3NF \supset BCNF$
- 找出F; 先验证R是否符合BCNF定义。若是, 则 $R \in BCNF$; 否则验证R是否符合3NF定义, 若是, 则 $R \in 3NF$; 否则验证R是否符合2NF定义, 若是, 则 $R \in 2NF$; 否则 $R \in 1NF$ 。

- 说明: 为简化写法, 上面的 $R \in ?NF$ 是指R达到的最高范式, 区别于通常的含义



- 两种方法的前提都需要先找出R的所有候选码
 - 求出R的所有候选码方法: **Armstrong公理**(后续章节)
 - **原因**: 只单纯从语义出发得出的**F未必是完全没有遗漏的**
- **BCNF的确定**
 - 求出R的所有候选码
 - 列出F的所有函数依赖
 - 如果F中所有的函数依赖中的决定因素都包含码, 则 $R \in \text{BCNF}$; 否则, $R \notin \text{BCNF}$, R最高只能是3NF
 - 要将一个非**BCNF**分解到**BCNF**, 只需分别验证**非主属性**对码是否存在部分、传递函数依赖以及**主属性**对不包含它的候选码存在部分、传递函数依赖
- **[例] TEACH(STUDENT, COURSE, INSTRUCTOR)**
 - $F = \{(\text{STUDENT}, \text{COURSE}) \rightarrow \text{INSTRUCTOR}, \text{INSTRUCTOR} \rightarrow \text{COURSE}\}$
 - 判定**TEACH**的最高范式, 如果**TEACH** $\notin \text{BCNF}$, 则将**TEACH**分解到**BCNF**

R1(STUDENT, INSTRUCTOR)和R2(COURSE, INSTRUCTOR)



例[6.9] 设学校中某一门课程由多个教师讲授，他们使用相同的一套参考书。每个教员可以讲授多门课程，每种参考书可以供多门课程使用。
用**Teaching(C,T,B)**表示上述关系模式。

表6.3 非规范化关系示例

课程 C	教员 T	参考书 B
物理	{李勇 王军}	{普通物理学 光学原理 物理习题集}
数学	{李勇 张平}	{数学分析 微分方程 高等代数}
计算数学	{张平 周峰}	{数学分析 ...}
...



表6.4 规范化的二维表Teaching

课程 C	教员 T	参考书 B
物理	李勇	普通物理学
物理	李勇	光学原理
物理	李勇	物理习题集
物理	王军	普通物理学
物理	王军	光学原理
物理	王军	物理习题集
数学	李勇	普通物理学
数学	李勇	光学原理
数学	李勇	物理习题集
数学	张平	普通物理学
数学	张平	光学原理
数学	张平	物理习题集
...



■ Teaching模式分析

– 候选码：唯一， $(C,T,B) \Rightarrow$ 全码

– **Teaching** \in BCNF

– 但**Teaching**仍然存在冗余度大，更新异常的问题

原因：存在多值依赖

课程 C	教员 T	参考书 B
物理	李勇	普通物理学
物理	李勇	光学原理
物理	李勇	物理习题集
物理	王军	普通物理学
物理	王军	光学原理
物理	王军	物理习题集
数学	李勇	普通物理学
数学	李勇	光学原理
数学	李勇	物理习题集
数学	张平	普通物理学
数学	张平	光学原理
数学	张平	物理习题集
...

- 冗余度大：有多少名任课教师，参考书就要存储多少次
- 插入操作复杂：当某一课程增加一名任课教师时，该课程有多少本参照书，就必须插入多少个元组
- 删除操作复杂：某一门课要去掉一本参考书，该课程有多少名教师，就必须删除多少个元组
- 修改操作复杂：某一门课要修改一本参考书，该课程有多少名教师，就必须修改多少个元组



■ 多值依赖定义6.9

- 给定关系模式 $R(U, F)$, X, Y, Z 是 U 的子集, 并且 $Z=U-X-Y$ 。关系模式 $R(U)$ 中多值依赖 $X \twoheadrightarrow Y$ 成立, 当且仅当对 $R(U, F)$ 的任一关系 r 给定的一对 (x, z) 值, 有一组 Y 的值, 这组值仅仅决定于 x 值而与 z 值无关。

[例] Teaching(C, T, B)

- 对于 C 的每一个值, T 有一组值与之对应, 而不论 B 取何值。因此 T 多值依赖于 C , 即 $C \twoheadrightarrow T$ 。

■ 多值依赖的另一个等价的定义:

- 在 $R(U)$ 的任一关系 r 中, 如果存在元组 t, s 使得 $t[X]=s[X]$, 那么就必然存在元组 $w, v \in r$, (w, v 可以与 s, t 相同), 使得 $w[X]=v[X]=t[X]$, 而 $w[Y]=t[Y], w[Z]=s[Z], v[Y]=s[Y], v[Z]=t[Z]$ (即交换 s, t 元组的 Y 值所得的两个新元组必在 r 中), 则 Y 多值依赖于 X , 记为 $X \twoheadrightarrow Y$ 。这里 X, Y 是 U 的子集, $Z=U-X-Y$ 。



■ 平凡多值依赖和非平凡的多值依赖

- 若 $X \twoheadrightarrow Y$ ，而 $Z = \Phi$ ，即 Z 为空，则称 $X \twoheadrightarrow Y$ 为平凡的多值依赖。
 - 即，对于 $R(X, Y)$ ，如果有 $X \twoheadrightarrow Y$ 成立，则 $X \twoheadrightarrow Y$ 为平凡的多值依赖。
- 否则称 $X \twoheadrightarrow Y$ 为非平凡的多值依赖。

[例6.10] WSC(W,S,C): W-仓库, S-保管员, C-商品

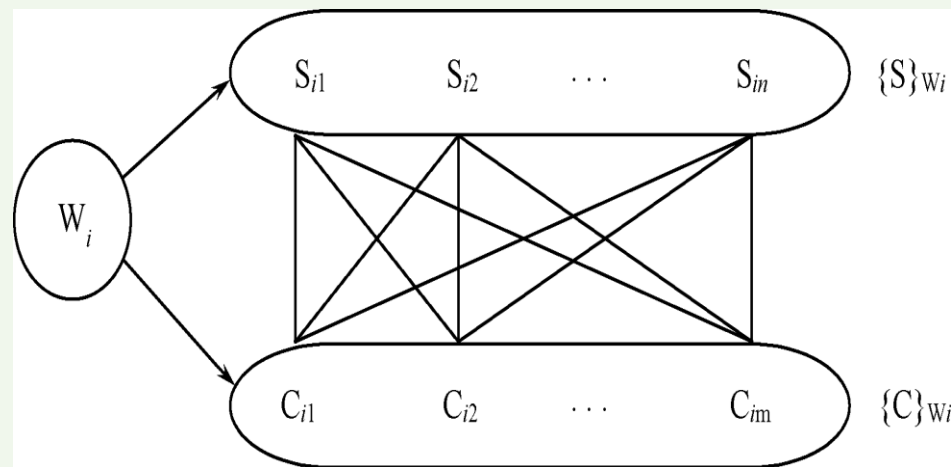
- 假设每个仓库有若干个保管员，有若干种商品
- 每个保管员保管所在仓库的所有商品
- 每种商品被所有保管员保管

W	S	C
W1	S1	C1
W1	S1	C2
W1	S1	C3
W1	S2	C1
W1	S2	C2
W1	S2	C3
W2	S3	C4
W2	S3	C5
W2	S4	C4
W2	S4	C5



[例6.10]的多值依赖分析

- 对于 W 的每一个值 W_i , S 有一个完整的集合与之对应而不问 C 取何值。所以 $W \twoheadrightarrow S$
- C 与 S 的完全对称性, 必然有 $W \twoheadrightarrow C$ 成立



$\{S\}_{W_i}$ $\{S\}_{W_i}$: 对应 W_i 的所有 S 值, 在 W_i 工作的全部保管员

$\{C\}_{W_i}$ $\{C\}_{W_i}$: 对应 W_i 的所有 C 值, 存放在 W_i 的所有商品

图6.7 $W \twoheadrightarrow S$ 且 $W \twoheadrightarrow C$

■ 多值依赖的性质:

1. 对称性

- 即若 $X \twoheadrightarrow Y$, 则 $X \twoheadrightarrow Z$, 其中 $Z = U - X - Y$
- 可用完全二分图直观地表示出来

2. 传递性

- 即如果 $XYZ=U$, $X \twoheadrightarrow Y$, $Y \twoheadrightarrow Z$, 则 $X \twoheadrightarrow Z-Y$

3. 函数依赖是多值依赖的特殊情况

- 即若 $X \rightarrow Y$, 则 $X \twoheadrightarrow Y$

4. 若 $X \twoheadrightarrow Y$, $X \twoheadrightarrow Z$, 则 $X \twoheadrightarrow YZ$

5. 若 $X \twoheadrightarrow Y$, $X \twoheadrightarrow Z$, 则 $X \twoheadrightarrow Y \cap Z$

6. 若 $X \twoheadrightarrow Y$, $X \twoheadrightarrow Z$, 则 $X \twoheadrightarrow Y-Z$, $X \twoheadrightarrow Z-Y$



■ 多值依赖与函数依赖的区别

1. 多值依赖的有效性与属性集的范围有关

- 若 $X \twoheadrightarrow Y$ 在 U 上成立, 则在 $W(XY \subseteq W \subseteq U)$ 上一定成立; 反之则不然, 即 $X \twoheadrightarrow Y$ 在 $W(W \subset U)$ 上成立, 在 U 上并不一定成立.
 - 这是因为多值依赖的定义中不仅涉及 U 的子集 X 和 Y , 而且涉及 U 中其余的子集 Z
- 一般地, 在 $R(U, F)$ 上若有 $X \twoheadrightarrow Y$ 在 $W(W \subset U)$ 上成立, 则称 $X \twoheadrightarrow Y$ 为 $R(U, F)$ 的嵌入型多值依赖.
- 在关系模式 $R(U, F)$ 中, 函数依赖 $X \rightarrow Y$ 的有效性仅决定于 X 、 Y 的值.
 - 只要在 $R(U)$ 的任何一个关系 r 中, 元组在 X 和 Y 上的值满足定义6.1, 则函数依赖 $X \rightarrow Y$ 在任何属性集 $W(XY \subseteq W \subseteq U)$ 上成立

2. 若函数依赖 $X \rightarrow Y$ 在 $R(U, F)$ 上成立, 则对于任何 $Y' \subset Y$ 均有 $X \rightarrow Y'$ 成立。而多值依赖 $X \twoheadrightarrow Y$ 若在 $R(U, F)$ 上成立, 不能断言对于任何 $Y' \subset Y$ 有 $X \twoheadrightarrow Y'$ 成立



■ 多值依赖与函数依赖的区别

- 例如，关系 $R(A, B, C, D)$ ， $A \twoheadrightarrow BC$ 成立，当然也有 $A \twoheadrightarrow D$ 成立。有 R 的一个关系实例，在此实例上 $A \twoheadrightarrow B$ 是不成立的。

表6.6 R 的一个实例

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_1	d_2
a_1	b_2	c_2	d_1
a_1	b_2	c_2	d_2



■ 多值依赖的判定方法

– 以上例为例进行说明：

1. $A \twoheadrightarrow BC$ 成立：

- 先选取第一条元组 r ，得出 $r(A)$ 的值，再往下从第二条元组开始，依次检查后续元组 r' 在 A 上的分量是否等于 $r(A)$ 。若相等，则调换两条元组在 BC 分量并检查调换后得到新的两个元组是否还在 R 中？若在，则继续上述比较过程，若不在，则表明 $A \nrightarrow BC$ ，即不满足多值依赖的条件。
- 示例操作：因为第1, 2条元组在属性 A 上的值相同，交换 (a_1, b_1, c_1, d_1) 与 (a_1, b_1, c_1, d_2) 在 BC 上的值后仍然在 R 中。穷尽实例上的所有元组，可得最后结果。



2. $A \twoheadrightarrow B$ 不成立

- 因为交换 (a_1, b_1, c_1, d_1) 与 (a_1, b_2, c_2, d_1) 在 B 上的分量后变为： (a_1, b_2, c_1, d_1) 和 (a_1, b_1, c_2, d_1) ，这两条新元组都不在 R 中，所以 $A \nrightarrow B$ 。



1. 以下是关系模式R的一个实例，问：
该实例满足以下哪个多值依赖？

- $B \twoheadrightarrow A$
- $D \twoheadrightarrow BC$
- $A \twoheadrightarrow A$
- $D \twoheadrightarrow A$

A	B	C	D
1	2	3	4
1	3	3	3
1	3	3	4
1	2	3	3
2	2	4	4
2	4	2	4
2	4	4	4
2	2	2	4

2. 假设关系模式R(A, B, C, D, E)有如下多值依赖： $A \twoheadrightarrow B$, $B \twoheadrightarrow D$
现在R包含两个元组(0,1,2,3,4)和(0,5,6,7,8)，请问下列哪些元组必须也包含在R中？

- (0,5,2,7,4)
- (0,5,6,7,4)
- (0,1,2,3,8)
- (0,5,2,7,8)



■ 定义6.10

– 关系模式 $R(U, F) \in 1NF$ ，如果对于 R 的每个非平凡多值依赖 $X \twoheadrightarrow Y$ ($Y \not\subseteq X$)， X 都含有码，则 $R(U, F) \in 4NF$ 。

■ 4NF就是限制关系模式的属性之间不允许有非平凡且非函数依赖的多值依赖

– 对于每一个非平凡的多值依赖 $X \twoheadrightarrow Y$ ， X 都含有候选码，于是 $X \rightarrow Y$ ，所以 4NF所允许的非平凡多值依赖实际上是函数依赖

■ $R \in 4NF \Rightarrow R \in BCNF$



■ 将BCNF的R分解成4NF的方法

- 假设 $R(A,B,C) \in \text{BCNF}$ 满足 $A \twoheadrightarrow B$, $A \twoheadrightarrow C$, 则分解R可分解为 $R_1(A, B)$ 和 $R_2(A, C)$, 要求 $R_1(A, B)$ 和 $R_2(A, C)$ 都是平凡的多值依赖
- 原则是分解后的所有关系模式必须都是平凡的多值依赖

■ 将[例6.10]中的WSC分解成4NF

- 经分析, $WSC \in \text{BCNF}$, 且 $WSC \notin 4NF$
- 因 $W \twoheadrightarrow S$, $W \twoheadrightarrow C$, 可分解WSC为 $WS(W,S)$ 和 $WC(W,C)$, $WS(W,S)$ 和 $WC(W,C)$ 都是平凡的多值依赖, 且 $WS \in 4NF$, $WC \in 4NF$

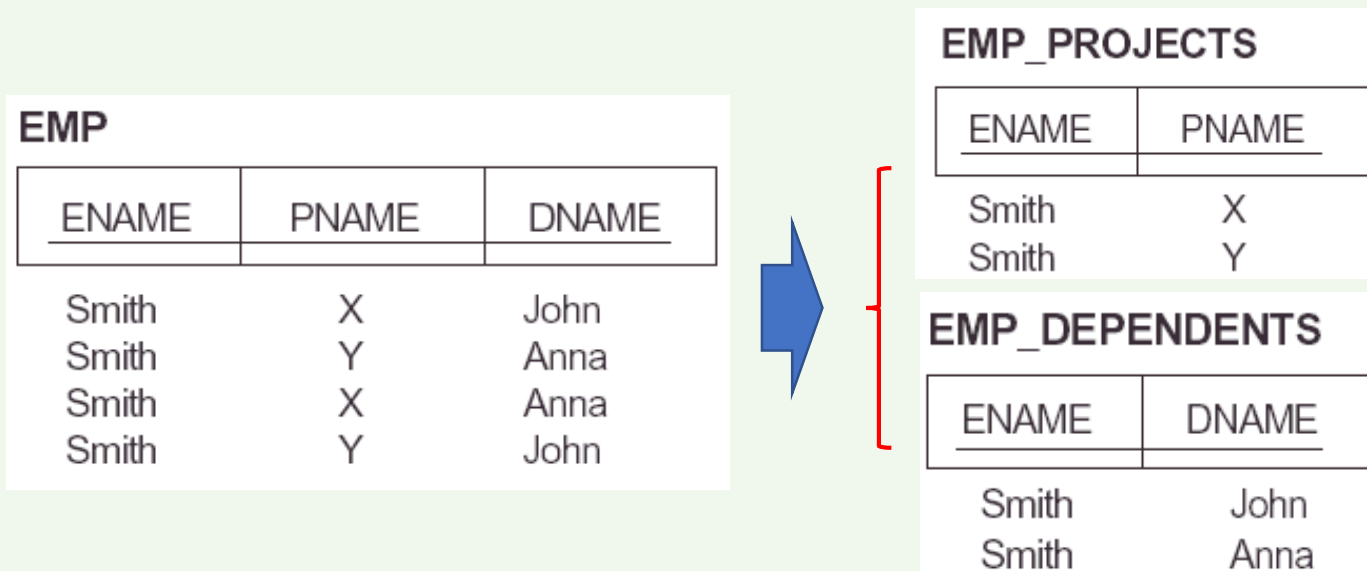


[例] 假设EMP (Ename, Pname, Dname)(注：全码)具有多值依赖：

$ENAME \twoheadrightarrow PNAME, ENAME \twoheadrightarrow DNAME$

请判断EMP \in 4NF是否成立？若不成立，请将EMP分解为4NF范式。

- **[解]** EMP为全码，且多值依赖的左部不含有码，故EMP \notin 4NF。将EMP分解为两个平凡的多值依赖：EMP_PROJECTS, EMP_DEPENDENTS



- 下列关于函数依赖的叙述中，哪一条是不正确的？
 - A. 若 $X \rightarrow Y$, $Y \rightarrow Z$, 则 $X \rightarrow Z$
 - B. 若 $X \rightarrow Y$, $Y \rightarrow Y'$, 则 $X \rightarrow Y'$
 - C. 若 $X \rightarrow Y$, $X' \rightarrow X$, 则 $X' \rightarrow Y$
 - D. 若 $X' \rightarrow X$, 则 $X \rightarrow X'$
- 关系数据库规范化是为解决关系数据库中的_____问题而引入的
 - A. 操作异常和数据冗余
 - B. 提高查询速度
 - C. 减少数据操作的复杂性
 - D. 保证数据的安全性和完整性
- 假设关系模式 $R(A,B)$ 的最高范式为3NF，则下列说法_____是正确的
 - A. 它一定消除了插入和删除异常
 - B. 仍存在一定的插入或删除异常
 - C. 一定属于BCNF
 - D. B和C均是正确的



- 在关系数据库中，任意一个二元关系模式R至少可以达到?NF
- 如果一个关系模式R中的属性全部是主属性，则R至少可以达到?NF
- 如果一个关系模式R的主码是全码，则R至少可以达到?NF
- 上述三问能帮助我们更加精确快速地判断一个关系模式的最高范式



- 一个关系只要其分量都是不可分的数据项，它就是规范化的关系，但这只是最基本的规范化
- 规范化程度过低的关系不一定能够很好地描述现实世界
 - 可能存在插入异常、删除异常、修改复杂、数据冗余等问题
 - 解决方法就是对其投影分解，转换成高级范式
- 一个低一级范式的关系模式，通过模式分解可以转换为若干个高一级范式的关系模式集合，这种过程就叫关系模式的规范化
- 函数依赖和多值依赖是两种最重要的数据依赖
 - BCNF为规范化程度最高的函数依赖
 - 4NF为规范化程度最高的多值依赖
- 关系数据库的规范化理论是数据库逻辑设计的工具



■ 规范化的基本思想

- 是逐步消除数据依赖中不合适的部分，使模式中的各关系模式达到某种程度的“分离”
- “一事一地”的模式设计原则
 - 让一个关系描述一个概念、一个实体或者实体间的一种联系
 - 若多于一个概念就把它“分离”出去
- 规范化实质上是概念的单一化

■ 不能说规范化程度越高的关系模式就越好

- 必须对现实世界的实际情况和用户应用需求作进一步分析，确定一个合适的、能够反映现实世界的模式
- 上面的规范化步骤可以在其中任何一步终止



关系模式规范化的基本步骤



- 数据依赖的公理系统是模式分解算法的理论基础.
- Armstrong公理系统
 - 函数依赖的一个有效而完备的公理系统
 - 一套推理规则, 是模式分解算法的理论基础
 - 1974年由Armstrong提出
 - 用途
 - 求给定关系模式的码
 - 从一组函数依赖求得蕴含的函数依赖
- 逻辑蕴含定义6.11
 - 给定关系模式 $R(U, F)$, 其任何一个关系 r , 若函数依赖 $X \rightarrow Y$ 都成立(即 r 中任意两元组 t, s , 若 $t[X]=s[X]$, 则 $t[Y]=s[Y]$), 则称 F 逻辑蕴含 $X \rightarrow Y$.



■ Armstrong公理系统

- 设 U 为属性组全集, F 是 U 上的一组函数依赖, 于是有关系模式 $R(U, F)$ 。对 $R(U, F)$ 来说有以下的推理规则:

A1.自反律(Reflexivity):

- 若 $Y \subseteq X \subseteq U$, 则 $X \rightarrow Y$ 为 F 所蕴涵

A2.增广律(Augmentation):

- 若 $X \rightarrow Y$ 为 F 所蕴涵, 且 $Z \subseteq U$, 则 $XZ \rightarrow YZ$ 为 F 所蕴涵。 --- $XZ = XU \cup Z, YZ = Y \cup Z$

A3.传递律(Transitivity):

- 若 $X \rightarrow Y$ 及 $Y \rightarrow Z$ 为 F 所蕴含, 则 $X \rightarrow Z$ 为 F 所蕴含

- 注意: 由自反律所得到的函数依赖均是平凡的函数依赖, 自反律的使用并不依赖于 F

■ 定理6.1 Armstrong推理规则是正确的。(证明略)



- 根据A1, A2, A3三条推理规则可以得到三条有用推理规则:
 - 合并规则: 由 $X \rightarrow Y$, $X \rightarrow Z$, 有 $X \rightarrow YZ$
 - (A2, A3) $X \rightarrow YX$, $XY \rightarrow ZY$
 - 伪传递规则: 由 $X \rightarrow Y$, $WY \rightarrow Z$, 有 $XW \rightarrow Z$
 - (A2, A3) $XW \rightarrow YW$
 - 分解规则: 由 $X \rightarrow Y$ 及 $Z \subseteq Y$, 有 $X \rightarrow Z$
 - (A1, A3) $Z \subseteq Y$, $Y \rightarrow Z$
- 引理6.1 $X \rightarrow A_1 A_2 \dots A_k$ 成立的充分必要条件是 $X \rightarrow A_i$ 成立($i=1, 2, \dots, k$)
 - 必要性: 使用分解规则
 - 充分性: 使用合并规则



- 闭包

- **定义6.12** 在关系模式 $R(U, F)$ 中为**F**所逻辑蕴含的函数依赖的全体叫作**F**的闭包，记为 **F^+** 。

- **Armstrong**是有效的

- 指**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

- **Armstrong**是完备的

- **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

- **定义6.13** 设**F**为属性集**U**上的一组函数依赖， $X \subseteq U$ ， $X_F^+ = \{A \mid X \rightarrow A \text{ 能由 } F \text{ 根据 Armstrong 公理导出}\}$ ，
 X_F^+ 称为**X**关于函数依赖集**F**的闭包

- **引理6.2** 设**F**为属性集**U**上的一组函数依赖， $X, Y \subseteq U$ ， $X \rightarrow Y$ 能由**F**根据**Armstrong**公理导出的充分必要条件是 **$Y \subseteq X_F^+$**

- **作用：**判定 $X \rightarrow Y$ 是否能由**F**根据**Armstrong**公理导出的问题转化为先求出 **X_F^+** 然后再判定**Y**是否为 **X_F^+** 的子集问题



- **Armstrong**公理的完备性及有效性说明了“导出”与“蕴涵”是完全等价的概念。 F^+ 也可以说成是由**F**出发借助**Armstrong**公理导出的函数依赖的集合。
 - F^+ 为**F**所逻辑**蕴含**的函数依赖的全体 (定义6.12)
 - F^+ 也可以说成由**F**出发借助**Armstrong**公理**导出**的函数依赖的集合
- 设 $F = \{ X \rightarrow Y, Y \rightarrow Z \}$,
 - $F^+ = \{ X \rightarrow \varphi, Y \rightarrow \varphi, Z \rightarrow \varphi, XY \rightarrow \varphi, XZ \rightarrow \varphi, YZ \rightarrow \varphi, XYZ \rightarrow \varphi, X \rightarrow X, Y \rightarrow Y, Z \rightarrow Z, XY \rightarrow X, XZ \rightarrow X, YZ \rightarrow Y, XYZ \rightarrow X, X \rightarrow Y, Y \rightarrow Z, XY \rightarrow Y, XZ \rightarrow Y, YZ \rightarrow Z, XYZ \rightarrow Y, X \rightarrow Z, Y \rightarrow YZ, XY \rightarrow Z, XZ \rightarrow Z, YZ \rightarrow YZ, XYZ \rightarrow Z, X \rightarrow XY, XY \rightarrow XY, XZ \rightarrow XY, XYZ \rightarrow XY, X \rightarrow XZ, XY \rightarrow YZ, XZ \rightarrow XZ, XYZ \rightarrow YZ, X \rightarrow YZ, XY \rightarrow XZ, XZ \rightarrow XY, XYZ \rightarrow XZ, XY \rightarrow XYZ, XZ \rightarrow XYZ, XYZ \rightarrow XYZ \}$



算法 6.1 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^+

计算步骤:

1. 令 $X^{(0)} = X$, $i=0$
 2. 求 B , 这里 $B = \{ A \mid (\exists V)(\exists W)(V \rightarrow W \in F \wedge V \subseteq X^{(i)} \wedge A \in W) \}$
 3. $X^{(i+1)} = B \cup X^{(i)}$
 4. 判断 $X^{(i+1)} = X^{(i)}$
 5. 若 $X^{(i+1)}$ 与 $X^{(i)}$ 相等或 $X^{(i)} = U$, 则 $X^{(i)}$ 就是 X_F^+ , 算法终止
 6. 若否, 则 $i = i+1$, 返回第2步
-



[例6.11] 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 $(AB)^+_F$ ，判断 AB 是否为候选码。

[解] 迭代使用算法6.1得到 $(AB)^+_F = ABCDE = U$ ，所以 AB 为候选码

课堂练习：

- 设有关系模式 $R(A, B, C, D)$ 及其函数依赖集 $F=\{D \rightarrow B, B \rightarrow D, AD \rightarrow B, AC \rightarrow D\}$ ，求 $(AC)^+_F$ 和 $(AB)^+_F$ 并判断 AC 和 AB 是否为候选码。



- 给定关系模式的任何属性都有如下性质：
 - 只出现在**箭头右侧**的属性一定不会是码的组成部分，因为它**不可能**是决定因素
 - 只出现在**箭头左侧**的属性一定是码的组成部分，因为它**只能**是决定因素
 - 既出现在**箭头左侧**又出现在**箭头右侧**的属性需要进行验证是否为码的组成部分
- 由上述性质+引理6.2可较快速求出关系模式的所有候选码

课堂练习：

- 利用上述方法重新考察例6.11
- 设 $U=\{A, B, C, D\}$, $F=\{A \rightarrow B, C \rightarrow D\}$ ，试求此关系的候选码。
- 设有关系模式 $R(A, B, C, D, E, P)$ 及其函数依赖集 $F=\{A \rightarrow D, E \rightarrow D, D \rightarrow B, BC \rightarrow D, DC \rightarrow A\}$ ，求 R 的所有候选码。
- 关系模式 $R(U, F)$, $U=\{A, B, C, D, E, P\}$, $F=\{A \rightarrow B, C \rightarrow D, E \rightarrow A, CE \rightarrow D\}$ ，试求此关系的候选码。



【例】 关系模式 $R(U, F)$ ，请针对以下不同情形的 U 和 F ，试确定 R 的最高范式。若 R 不是BCNF则将其分解到BCNF。

1. $U=\{A, B, C, D\}$, $F=\{B \rightarrow D, AB \rightarrow C\}$;
2. $U=\{A, B, C, D, E\}$, $F=\{AB \rightarrow CE, E \rightarrow AB, C \rightarrow D\}$;
3. $U=\{A, B, C\}$, $F=\{A \rightarrow B, B \rightarrow A, A \rightarrow C\}$

【解】

1. 首先确定 R 只有一个候选码：**AB**，因此**AB**也是主码。因非主属性**D**部分函数依赖于**AB**，所以**R**的最高范式为**1NF**。分解为BCNFs： **$R_1(A, B, C, AB \rightarrow C)$ 和 $R_2(B, D, B \rightarrow D)$**
2. 易得 R 有两个候选码：**AB**、**E**。因为 $R \notin \text{BCNF}$ ，所以**R最高范式至多为3NF**。因 **$AB \rightarrow CE$** ，故 **$AB \rightarrow CE$** ，又 **$C \rightarrow D$** ，所以存在非主属性**D**对码**AB**的传递函数依赖，所以 $R \notin 3\text{NF}$ 。又**F**中不存在部分依赖，所以**R的最高范式为2NF**。分解为BCNFs： **$R_1(A, B, C, E, \{AB \rightarrow CE, E \rightarrow AB\})$ 和 $R_2(C, D, C \rightarrow D)$**
3. 作为课堂练习，请自行完成



■ 函数依赖集等价定义

– 定义6.14:

- 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 与 G 等价。
- 即，两个函数依赖集等价是指它们的闭包等价

■ 引理6.3: $F^+ = G^+$ 的充分必要条件是 $F \subseteq G^+$ 和 $G \subseteq F^+$

– 证: 必要性显然，只证充分性。

(1) 若 $F \subseteq G^+$ ，则 $X_F^+ \subseteq X_G^+$

(2) 任取 $X \rightarrow Y \in F^+$ ，则有 $Y \subseteq X_F^+ \subseteq X_G^+$ 。所以 $X \rightarrow Y \in (G^+)^+ = G^+$ 。即 $F^+ \subseteq G^+$

(3) 同理可证 $G^+ \subseteq F^+$ ，所以 $F^+ = G^+$



- 引理6.3给出了判断两个函数依赖集等价的可行算法
- 如何判定 $F \subseteq G^+$?
 - 只须逐一对 F 中的函数依赖 $X \rightarrow Y$, 考察 Y 是否属于 X_{G^+}
- 例: $F = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$, $G = \{A \rightarrow CD, E \rightarrow AH\}$, 证明 F 与 G 等价.
 - 除了使用引理6.3证明之外, 也可直接使用前面介绍的A1, A2, A3三条准则证明
- 在 $R(U, F)$ 中可以用与 F 等价的依赖集 G 来取代 F
 - 原因: 两个关系模式 $R_1(U, F)$, $R_2(U, G)$, 如果 F 与 G 等价, 那么 R_1 的关系一定是 R_2 的关系。
反过来, R_2 的关系也一定是 R_1 的关系。



■ 极小函数依赖集

– 如果函数依赖集F满足下列条件，则称F为一个极小函数依赖集，或称极小依赖集、最小覆盖(minimal cover)

- ① F中任一函数依赖的右部仅含有一个属性
- ② F中不存在这样的函数依赖 $X \rightarrow A$ ，使得F与 $F - \{X \rightarrow A\}$ 等价 //说明不存在传递函数依赖
- ③ F中不存在这样的函数依赖 $X \rightarrow A$ ，X有真子集Z使得 $\{F - \{X \rightarrow A\}\} \cup \{Z \rightarrow A\}$ 与F等价
//说明F不存在部分函数依赖，换言之，若存在则说明F不是最小覆盖

[例6.12] 对于6.1节中的关系模式Student(U, F)，其中：

$U = \{ Sno, School, Mname, Cno, Grade \},$

$F = \{ Sno \rightarrow School, School \rightarrow Mname, (Sno, Cno) \rightarrow Grade \}$

设 $F' = \{ Sno \rightarrow School, Sno \rightarrow Mname, School \rightarrow Mname,$
 $(Sno, Cno) \rightarrow Grade, (Sno, School) \rightarrow School \}$

- 可以验证，F为最小覆盖；但F'不是，因为 $F' - \{Sno \rightarrow Mname\}$ 与F'等价



- **定理6.3:** 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。
- 构造性证明:
 - 分三步对 F 进行“极小化处理”, 找出 F 的一个最小依赖集.
 - 1. 逐一检查 F 中各函数依赖 $FD_i: X \rightarrow Y$, 若 $Y = A_1 A_2 \dots A_k$, $k \geq 2$, 则用 $\{X \rightarrow A_j \mid j=1, 2, \dots, k\}$ 来取代 $X \rightarrow Y$ (引理6.1保证了该变换的等价性)
 - 2. 逐一检查 F 中各函数依赖 $FD_i: X \rightarrow A$, 令 $G = F - \{X \rightarrow A\}$, 若 $A \in X_G^+$, 则从 F 中去掉此函数依赖 (因为 F 与 G 等价的充要条件是 $A \in X_G^+$, 保证了变换的等价性)
 - 3. 逐一取出 F 中各函数依赖 $FD_i: X \rightarrow A$, 设 $X = B_1 B_2 \dots B_m$, $m \geq 2$, 逐一考查 B_i ($i=1, 2, \dots, m$), 若 $A \in (X - B_i)_F^+$, 则以 $X - B_i$ 取代 X (因为 F 与 $F - \{X \rightarrow A\} \cup \{Z \rightarrow A\}$ 等价的充要条件是 $A \in Z_F^+$, 其中 $Z = X - B_i$, 这保证了变换的等价性)



- 定理6.3的证明过程就是求F极小依赖集的过程，也是检验F是否为最小依赖集的一个算法。
- F的极小依赖集 F_m 不一定是唯一的，它与对各函数依赖 FD_i 及 $X \rightarrow A$ 中X各属性的处置顺序有关。

[例6.13] $F = \{A \rightarrow B, B \rightarrow A, B \rightarrow C, A \rightarrow C, C \rightarrow A\}$,

- $F_{m1} = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$, --取消 $B \rightarrow A$ 或 $A \rightarrow C$ 得到
- $F_{m2} = \{A \rightarrow B, B \rightarrow A, A \rightarrow C, C \rightarrow A\}$ --取消 $B \rightarrow C$ 得到
- 可以验证, F_{m1} 和 F_{m2} 是F的两个不同的最小依赖集



[例] 设有依赖集： $F=\{AB\rightarrow C, C\rightarrow A, BC\rightarrow D, ACD\rightarrow B, D\rightarrow EG, BE\rightarrow C, CG\rightarrow BD, CE\rightarrow AG\}$ ，求与F等价的最小依赖集。

[解]：

1.将依赖右边属性**单一化**

– $F_1 = \{AB\rightarrow C, C\rightarrow A, BC\rightarrow D, ACD\rightarrow B, D\rightarrow E, D\rightarrow G, BE\rightarrow C, CG\rightarrow B, CG\rightarrow D, CE\rightarrow A, CE\rightarrow G\}$

2.在F1中去掉依赖**左部多余**的属性。对于 $CE\rightarrow A$ ，由于 $C\rightarrow A$ ，故E是多余的；对于 $ACD\rightarrow B$ ，由于 $(CD)^+=ABCDEG$ ，故A多余。删除依赖左部多余的依赖后：

– $F_2 = \{AB\rightarrow C, C\rightarrow A, BC\rightarrow D, CD\rightarrow B, D\rightarrow E, D\rightarrow G, BE\rightarrow C, CG\rightarrow B, CG\rightarrow D, CE\rightarrow G\}$

3.在F2中去掉**多余的依赖**。对 $CG\rightarrow B$ ，由于 $(CG)^+=ABCDEG$ ，故 $CG\rightarrow B$ 是多余的。删除依赖左部多余的依赖后：

– $F_3 = \{AB\rightarrow C, C\rightarrow A, BC\rightarrow D, CD\rightarrow B, D\rightarrow E, D\rightarrow G, BE\rightarrow C, CG\rightarrow D, CE\rightarrow G\}$

注： $CG\rightarrow B$ 与 $CD\rightarrow B$ 不能同时存在，但去掉任何一个都可以，说明最小依赖集不唯一



[练习1] 设 $R(U, F)$, $U=ABCD$, $F=\{A \rightarrow BD, AB \rightarrow C, C \rightarrow D\}$, 求其最小依赖集。

– 步骤:

- 1.将 F 中的所有依赖右边化为单一元素
- 2.去掉 F 中的所有依赖左边的冗余属性
- 3.去掉 F 中所有冗余依赖关系

[练习2] 已知 $R(U, F)$, $U=ABCDEFGH IJ$, $F=\{ABC \rightarrow E, AB \rightarrow G, B \rightarrow F, C \rightarrow J, CJ \rightarrow I, G \rightarrow H\}$, 求 $R(U, F)$ 的最小依赖集。

▪ 参考答案: $F'=\{ABC \rightarrow E, AB \rightarrow G, B \rightarrow F, C \rightarrow J, C \rightarrow I, G \rightarrow H\}$



- 关系模式的规范化过程是通过对关系模式的分解来实现的
 - 把低一级的关系模式分解为若干个高一级的关系模式的方法并不是唯一的
 - 在这些分解方法中，只有能够保证分解后的关系模式与原关系模式等价的方法才有意义

- 定义6.16** 关系模式 $R(U, F)$ 的一个分解是指

$$\rho = \{R_1(U_1, F_1), R_2(U_2, F_2), \dots, R_n(U_n, F_n)\}$$

其中, $U = \bigcup_{i=1}^n U_i$, 并且没有 $U_i \subseteq U_j, 1 \leq i \neq j \leq n$, F_i 是 F 在 U_i 上的投影, 即

$$F_i = \{X \rightarrow Y | X \rightarrow Y \in F^+ \wedge XY \subseteq U_i\}.$$

- 说明: 若只要求 R 分解后的各关系模式所含属性的“并”等于 U , 这个限定是很不够的, 下例可以说明



[例6.14] 已知关系模式 $R(U, F)$ ，其中

- $U = \{Sno, School, Mname\}$, $F = \{Sno \rightarrow School, School \rightarrow Mname\}$
- 语义是学生 **Sno**正在一个学院**School**学习，其院长姓名是**Mname**
- 并且一个学生(**Sno**)只在一个学院学习，一个学院只有一名院长

■ **存在的问题：**

- 由于 R 中存在传递函数依赖 $Sno \xrightarrow{T} Mname$ ，它会发生更新异常
- 例如，如果 **S4** 毕业，删除该条记录，则**D3**学院的院长王一的信息也就丢掉了
- 反过来，如果一个学院 **D5** 刚刚成立尚无在校学生，那么这个学院院长赵某的信息也无法存入

■ **现在考虑对 R 进行如下分解：**

- $\rho = \{R_1(\{Sno, School\}, \{Sno \rightarrow School\}), R_2(\{Sno, Mname\}, \{Sno \rightarrow Mname\})\}$
- 表 R 被分解为两个新的表 R_1 和 R_2 ，每个新表有4个元组
- 但是，这样的分解仍然存在**插入和删除异常**，因为 R 中原来的函数依赖 $School \rightarrow Mname$ 在 R_1 和 R_2 中都不再存在了，即这个**函数依赖被丢掉了**。因此一个合理的要求就是，分解应“保持函数依赖”。分解后的模式应该保持与原模式的“等价”，在此，我们采用**保持函数依赖作为模式等价的一种准则**

表6.7 R 的一个关系示例

Sno	School	Mname
S1	D1	张五
S2	D1	张五
S3	D2	李四
S4	D3	王一

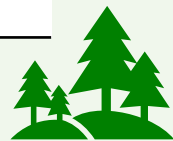


- **定义6.17** $\rho = \{R_1(U_1, F_1), R_2(U_2, F_2), \dots, R_k(U_k, F_k)\}$ 是关系模式 $R(U, F)$ 的一个保持函数依赖的分解, 如果 $F^+ = (\bigcup_{i=1}^k F_i)^+$ 。
 - 容易验证, 上例中 $School \rightarrow Mname \notin (F_1 \cup F_2)^+$, 即该分解不保持函数依赖
- 因此, 保持函数依赖是模式分解等价的一个准则。
- 6.3节引理6.3给出了判断两个函数依赖集等价的可行算法, 因此引理6.3也可用于判别 R 的分解 ρ 是否保持函数依赖。



算法6.2(合成法) 转换为3NF的保持函数依赖的分解

- 1.对 $R(U, F)$ 中的函数依赖集 F 进行极小化处理(处理后得到的依赖集仍记为 F)。
- 2.找出所有不在 F 中出现的属性(记为 U_0), 构成一个关系模式 $R_0(U_0, F_0)$ 。把这些属性从 U 中去掉, 剩余的属性仍记为 U 。
- 3.若有 $X \rightarrow A \in F$, 且 $XA=U$, 则 $\rho = \{R\}$, 算法终止。
- 4.否则, 对 F 按具有相同左部的原则分组(假定分为 k 组), 每一组函数依赖所涉及的全部属性形成一个属性集 U_i 。若 $U_i \subseteq U_j (i \neq j)$ 就去掉 U_i 。由于经过了步骤2, 故 $U = \bigcup_{i=1}^k U_i$, 于是 $\rho = \{R_1(U_1, F_1), R_2(U_2, F_2), \dots, R_k(U_k, F_k)\} \cup R_0(U_0, F_0)$ 构成 $R(U, F)$ 的一个保持函数依赖的分解, 且每个 $R_i(U_i, F_i)$ 均属于3NF。这里 F_i 是 F 在 U_i 上的投影。 F 只是经过了极小化处理和按照相同左部原则进行分组, 因此分解 ρ 保持函数依赖是显然的。



[证明] 以下证明每一 $R_i(U_i, F_i)$ 一定属于**3NF**

- 设 $F_i = \{X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_k\}$, $U_i = \{X, A_1, A_2, \dots, A_k\}$
- 1. $R_i(U_i, F_i)$ 一定以X为码。
- 2.若 $R_i(U_i, F_i)$ 不属于3NF, 则必存在非主属性 $A_m (1 \leq m \leq k)$ 及属性组合Y, $A_m \notin Y$, 使得 $X \rightarrow Y, Y \rightarrow A_m \in F_i^+$, 而 $Y \rightarrow X \notin F_i^+$ 。
- 3.若 $Y \subset X$, 则与 $X \rightarrow A_m$ 属于最小依赖集F相矛盾, 因而 $Y \not\subseteq X$ 。不妨设 $Y \cap X = X_1$, $Y - X = \{A_1, A_1, \dots, A_p\}$, 令 $G = F - \{X \rightarrow A_m\}$, 显然 $Y \subseteq X_G^+$, 即 $X \rightarrow Y \in G^+$ 。
- 可以断言 $Y \rightarrow A_m$ 也属于 G^+ 。因为 $Y \rightarrow A_m \in F_i^+$, 所以 $A_m \in Y_F^+$ 。若 $Y \rightarrow A_m$ 不属于 G^+ , 则在求 Y_F^+ 的算法中, 只有使用 $X \rightarrow A_m$ 才能将 A_m 引入。于是按算法6.1必有j, 使得 $X \subseteq Y^{(j)}$, $Y \rightarrow X$ 成立, 与 $Y \rightarrow X \notin F_i^+$ 矛盾。
- 于是 $X \rightarrow A_m$ 属于 G^+ , 与F是最小依赖集相矛盾, 所以 $R_i(U_i, F_i)$ 一定属于**3NF**。
- 至此, 我们已经有一个算法, 可以在保持函数依赖的情况下, 自动化地将一个关系模式分解为一组子关系模式, 而且这些子关系模式都满足**3NF**范式



- 本节讨论的问题：
 - 什么是无损连接？如何判断一个分解是否为无损连接？
 - 无损连接的模式分解与保持函数依赖的模式分解之间的关系
 - 实现无损连接模式分解的算法是什么？分解后关系模式所能达成的规范化程度如何？
- 本节仅给出重要结论并以例子加深理解，详细严格的证明请参考教材。



7

7.1无损连接的模式分解定义

- 无损连接(**lossless join**)的模式分解是指分解后的关系通过自然连接可以“恢复”原始关系。
- 算法6.3 判别一个分解的无损连接性

$\rho = \{R_1(U_1, F_1), R_2(U_2, F_2), \dots, R_k(U_k, F_k)\}$ 是关系模式 $R(U, F)$ 的一个的分解, $U = \{A_1, \dots, A_n\}$, $F = \{FD_1, FD_2, \dots, FD_\rho\}$, 不妨设 F 为最小依赖集, 记 FD_i 为 $X_i \rightarrow A_{li}$.

- ① 建立一张 n 列 k 行的表, 每一列对应一个属性, 每一行对应分解中的一个关系模式。若属性 A_j 属于 U_i , 则在 j 列 i 行交叉处填上 a_j , 否则填上 b_{ij} 。
- ② 对每一个 FD_i 做下列操作: 找到 X_i 所对应的列中具有相同符号的那些行, 考察这些行中 li 列的元素。若其中有 a_{li} , 则全部改为 a_{li} ; 否则全部改为 b_{mli} 。其中 m 是这些行的行号最小值。若某个 b_{tli} 被更改, 那么该表的 li 列中凡是 b_{tli} 的符号(不管它是否开始找到的那些行)均应做相应更改。如在某次更改之后, 有一行成为 a_1, a_2, \dots, a_n , 则算法终止, ρ 具有无损连接性, 否则 ρ 不具有无损连接性。对 F 中 ρ 个 FD 逐一进行一次这样的处理, 称为对 F 的一次扫描。
- ③ 比较扫描前后表有无变化。若有变化则返回第②步, 否则算法终止。

如果发生循环, 那么前次扫描至少应使该表减少一个符号, 表中符号有限, 因此循环必然终止。



- **定理6.4** 如果算法6.3终止时表中有一行为 a_1, a_2, \dots, a_n ，则 ρ 为无损连接分解。

[例6.15] 已知 $R(U, F)$, $U = \{A, B, C, D, E\}$, $F = \{AB \rightarrow C, C \rightarrow D, D \rightarrow E\}$, R 的一个分解为 $R_1(A, B, C), R_2(C, D), R_3(D, E)$. 判定该分解是否具有无损连接性。

A	B	C	D	F		A	B	C	D	E
a_1	a_2	a_3	b_{14}	b_{15}		a_1	a_2	a_3	a_4	a_5
b_{21}	b_{22}	a_3	a_4	b_{25}		b_{21}	b_{22}	a_3	a_4	a_5
b_{31}	b_{32}	b_{33}	a_4	a_5		b_{31}	b_{32}	b_{33}	a_4	a_5

(a) (b)

- (b)表中一行成为 a_1, a_2, \dots, a_5 ，所以此分解具有无损连接性。



- 当关系模式 R 分解为两个关系模式 R_1, R_2 时具有下面的判定准则。
- 定理6.5** 对于 $R(U, F)$ 的一个分解 $\rho = \{R_1(U_1, F_1), R_2(U_2, F_2)\}$, 如果 $U_1 \cap U_2 \rightarrow U_1 - U_2 \in F^+$ 或 $U_2 \cap U_1 \rightarrow U_2 - U_1 \in F^+$, 则 ρ 具有无损连接性。



- 分解具有无损连接性和分解保持函数依赖是**两个不同的概念**
 - 具有无损连接性的分解不一定能够保持函数依赖
 - 保持函数依赖的分解也不一定具有无损连接性
- 例子：
 - $R(\{\text{学号}, \text{姓名}, \text{课程号}, \text{课程名称}\}, \{\text{学号} \rightarrow \text{姓名}, \text{课程号} \rightarrow \text{课程名称}\})$, 以及一个可能的分解 $R_1(\{\text{学号}, \text{姓名}\}, \{\text{学号} \rightarrow \text{姓名}\})$ 和 $R_2(\{\text{课程号}, \text{课程名称}\}, \{\text{课程号} \rightarrow \text{课程名称}\})$ 。显然, 这个分解是保持函数依赖的。但是, 由于 R_1 和 R_2 之间不存在公共属性, 因此 R_1 与 R_2 的自然连接就退化为笛卡儿积。因此, 这个分解就不满足无损连接性。
- 可以看出, **保持函数依赖和无损连接是两个不同的概念**。也许, 既保持函数依赖又可以无损连接的分解才是最理想的分解。



算法6.4 转换为3NF既有无损连接性又保持函数依赖的分解

① 设 X 是 $R(U, F)$ 的码。 $R(U, F)$ 已由算法6.2分解为 $\rho =$

$$\{R_1(U_1, F_1), R_2(U_2, F_2), \dots, R_k(U_k, F_k)\} \cup R_0(U_0, F_0), \text{ 令 } \tau = \rho \cup \{R^*(X, F_x)\}.$$

② 有某个 U_i , 若 $X \subseteq U_i$, 则将 $R^*(X, F_x)$ 从 τ 中去掉; 若 $U_i \subseteq X$, 则将 $R_i(U_i, F_i)$ 从 τ 中去掉。

③ τ 就是所求的分解。

■ 算法6.4的正确性证明请参考教材。



- 如果只关注无损连接，那么可以进一步提高规范化程度。

算法6.5(分解法) 转化为BCNF的无损连接分解

- ① 令 $\rho = \{R(U, F)\}$ 。
- ② 检查 ρ 中各关系模式是否均属于 **BCNF**。若是，则算法终止。
- ③ 设 ρ 中 $R_i(U_i, F_i)$ 不属于 **BCNF**，那么必有 $X \rightarrow A \in F_i^+$ ($A \notin X$)，且 X 非 R_i 的码。因此， XA 是 U_i 的真子集。对 R_i 进行分解： $\sigma = \{S_1, S_2\}$ ， $U_{S_1} = XA$ ， $U_{S_2} = U_i - \{A\}$ ，以 σ 代替 $R_i(U_i, F_i)$ ，返回第②步。

由于 U 中属性有限，因而有限次循环后算法6.5一定会终止。

- 这是一个自顶向下的算法。它自然形成一棵对 $R(U, F)$ 的二叉分解树
 - 该分解树不一定是唯一的，这与步骤③中具体选定的 $X \rightarrow A$ 有关
- 使用算法6.5得到的是一个 **BCNF** 具有保持无损连接性的分解
 - 证明详见教材



- 6.2.8小节已经指出，一个关系模式中若存在多值依赖(指非平凡的非函数依赖的多值依赖)，则数据的冗余度大且存在增删改异常等问题。
- 为此要消除这种多值依赖使模式分离达到一个新的高度**4NF**。下面讨论达到**4NF**的具有无损连接性的分解。
- **定理6.6** 关系模式 $R(U, D)$ 中， D 为 R 中函数依赖和多值依赖的集合。则 $X \twoheadrightarrow Y$ 成立的充要条件是 R 的分解 $\rho = \{R_1(X, Y), R_2(X, Z)\}$ 具有无损连接性，其中 $Z = U - X - Y$ 。
- **算法6.6** 达到**4NF**的具有无损连接性的分解
 - 首先使用算法6.5得到 R 的一个达到**BCNF**的无损连接分解，然后对某一 $R_i(U_i, D_i)$ ，若不属于**4NF**，则可按定理6.6进行分解，直到每一个关系模式都属于**4NF**为止
 - 定理6.6和引理6.5保证了最后得到的分解的无损连接性



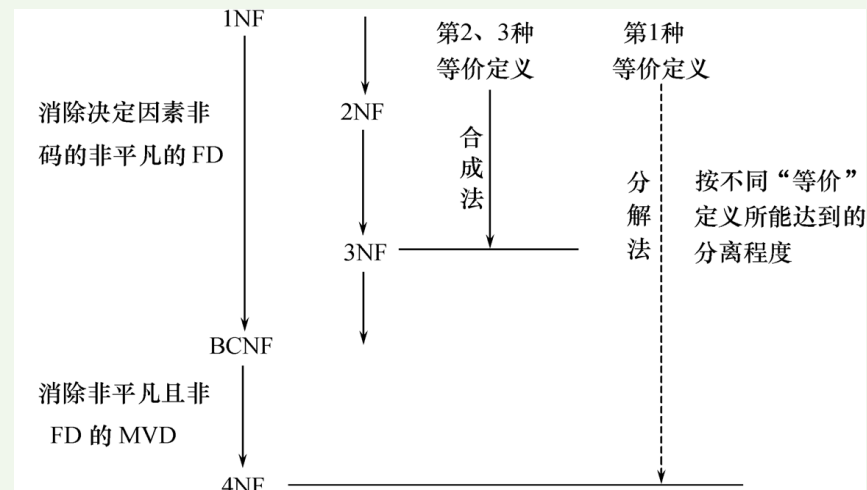
■ 函数依赖

- 函数依赖
- 平凡函数依赖与非平凡函数依赖
- 完全函数依赖与部分函数依赖
- 传递函数依赖
- 码

■ 多值依赖

- 多值依赖
- 平凡多值依赖和非平凡的多值依赖
- 多值依赖的性质
 - 对称性
 - 传递性

■ 关系模式规范化的基本步骤



■ Armstrong公理系统

- Armstrong公理
 - 自反律；增广律；传递律
 - 合并规则；伪传递规则；分解规则
- 求函数依赖闭包
- 求极小函数依赖集



- “从已知的函数依赖集 F 使用推理规则集推不出的函数依赖，必定不在 F^+ 中”，这句话是指推理规则的有效性还是完备性？



- 教材第六章习题2-8.
- 要求：作业在布置后一周内完成并提交到课程网站

