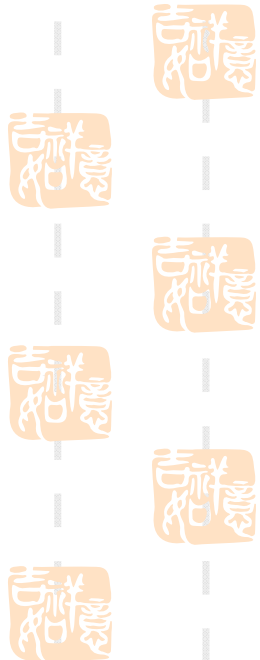




# 数据库系统概论

## An Introduction to Database System

### 第六章 关系数据理论



# 第六章 关系数据理论



## 6.1 问题的提出

## 6.2 规范化

## 6.3 数据依赖的公理系统

## \*6.4 模式的分解

## 6.5 小结



# 6.1 问题的提出



关系数据库逻辑设计

➤ 针对具体问题，如何构造一个适合于它的数据

模式

➤ 数据库逻辑设计的工具——关系数据库的规范

化理论



# 关系模式设计中的数据语义问题

- 考虑关系模式:

*Lending-schema* = (*branch-name*, *branch-city*, *assets*,  
*customer-name*, *loan-number*, *amount*)

<i>branch-name</i>	<i>branch-city</i>	<i>assets</i>	<i>customer-name</i>	<i>loan-number</i>	<i>amount</i>
Downtown	Brooklyn	9000000	Jones	L-17	1000
Redwood	Palo Alto	2100000	Smith	L-23	2000
Perryridge	Horseneck	1700000	Hayes	L-15	1500
Downtown	Brooklyn	90			

(Downtown Brooklyn 9000000 tom L-16 100)

- 假设要加入一笔新的贷款记录:

- 则 *branch-name*, *branch-city*, *assets* 数据项要重复一次, 重复信息浪费了空间。
- 更新操作复杂, 容易产生数据信息的不一致, 即更新异常。如更改*assets*必须更新同一支行所有的*assets*数据。
- Null值处理麻烦, 不允许null则若无贷款信息, 则相关的银行信息也必须删掉, 造成删除异常; 若一笔贷款都没有, 则不允许加入银行信息, 存在插入异常。

# 上述关系模式带来的问题

- 冗余
- 对表的操作可能会产生更新异常问题
  - **修改异常**：只修改一个元组，不修改另外元组中的相同信息
  - **插入异常**：假定每个元组必须要包含顾客名，则对于新成立的银行，必须要等到有顾客出现，该银行的信息才能插入数据库中
  - **删除异常**：当所有的顾客都删除了，则银行的信息也没有了

# 问题的根源

- 一个关系内，同时存在有多个实体的信息  
(模式中存在的某些数据依赖引起的)

- 银行的信息：名称、地点、资产
- 借贷户的信息：借贷户、账号、金额

■ 解决办法：把两类信息分开—分解(通过分解关系模式来消除其中不合适的数据依赖)

- 一个表中只有一个实体的信息（语义单一化）

# 规范化



规范化理论正是用来改造关系模式，通过分解关系模式来消除其中不合适的数据依赖，以解决插入异常、删除异常、更新异常和数据冗余问题。



# 函数依赖 (Functional Dependencies)

- 在关系数据库中，指一个或一组属性的值可以决定其他属性的值。例如，一个学生的学号可以决定一个学生的姓名。
- 一般地讲，设 $X$ ， $Y$ 是关系的两个不同的属性组，如果 $X$ 决定 $Y$ ，则其依赖关系可表示为 $X \rightarrow Y$ ，这种依赖关系是一种函数关系。
- 函数依赖存在与否，完全决定于数据的语义。如学号 $\rightarrow$ 姓名，就是根据数据语义而非数据值决定的。




# 函数依赖 (FD)

- 定义6.1
- 假设 **A** 和 **B** 是表 **T** 的两个属性

$$A \subseteq \text{Head}(T) \text{ and } B \subseteq \text{Head}(T)$$

我们说  $A \rightarrow B$  (读做“**A**函数决定**B**”或者“**B**函数依赖**A**”) 当且仅当对于表 **T** 的任何可能的内容, **T** 中的两行不能在 **A** 上取相同的值而在 **B** 上取不同的值。即给定 **T** 的两行 **r1** 和 **r2**, 如果  $r1(A)=r2(A)$ , 那么  $r1(B)=r2(B)$

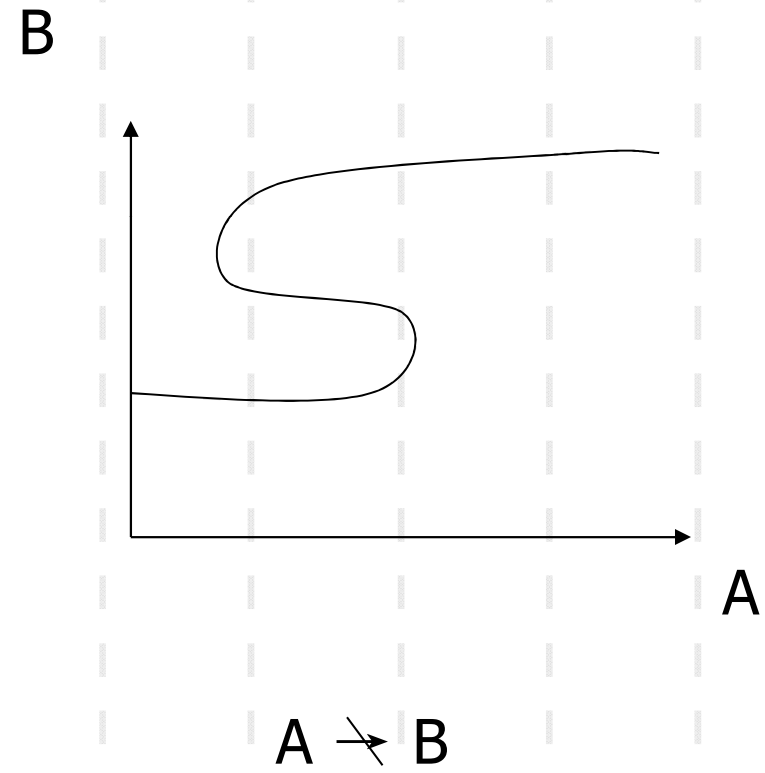
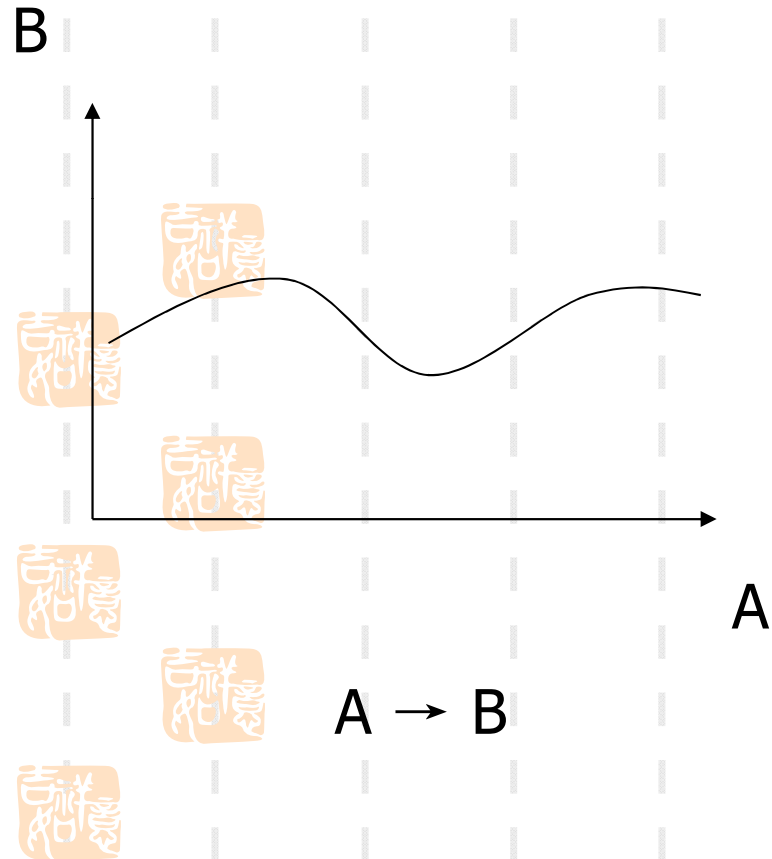
- Example: Consider  $r(A,B)$  with the following instance of  $r$ .



1	4
1	5
3	7

- 这里,  $A \rightarrow B$  不成立, 但是  $B \rightarrow A$  成立.
- 函数依赖可以在关系中表示一组属性组的值可唯一确定另一组属性组的值的约束要求。

# 例子



# 函数依赖的例子

有关系模式:

student(sname, sno, sex, age, dormitory)

可能存在的FD:

{sno} → {sname, sex, age, dormitory}

{sname, dormitory} → {sno, age}

{dormitory} → {sex}

假定一个宿舍  
中无同名的学生

要会找出一个模式中的所有函数依赖

# 一些概念和记号

- 平凡的依赖

- 若  $Y \subseteq X$ , 则  $X \rightarrow Y$  成立, 则称为平凡的依赖

- 如  $\{\text{sex}, \text{sname}\} \rightarrow \{\text{sname}\}$

- 简单地讲,  $\text{FD}: A \rightarrow B$  中 存在 “包含” 关系。

- 平凡依赖是必然成立的。

- 非平凡的依赖, 如  $\{\text{sex}, \text{sname}\} \rightarrow \{\text{sname}, \text{age}\}$  和  $\{\text{sex}, \text{sname}\} \rightarrow \{\text{age}\}$

- 不函数依赖  $X \nrightarrow Y$

- 若  $X \rightarrow Y$  和  $Y \rightarrow X$  成立, 则  $X$  和  $Y$  一一对应, 记为  $X \leftrightarrow Y$

- $\{\text{sno}\} \leftrightarrow \{\text{dormitory}, \text{sname}\}$

f

p

# 完全函数依赖与部分函数依赖

❖ 定义6.2 在 $R(U)$ 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 $X \xrightarrow{P} Y$

❖ 部分依赖的例子:  $\{\text{sno}, \text{sex}\} \rightarrow \{\text{dormitory}, \text{sname}\}$

# 完全函数依赖与部分函数依赖

[例1] 中  $(Sno, Cno) \xrightarrow{F} Grade$  是完全函数依赖,

$(Sno, Cno) \xrightarrow{P} Sdept$  是部分函数依赖

因为  $Sno \rightarrow Sdept$  成立, 且  $Sno$  是  $(Sno,$

$Cno)$  的真子集

# 传递函数依赖

定义**6.3** 在 $R(U)$ 中, 如果 $X \rightarrow Y$ ,  $(Y \subsetneq X)$ ,  $Y \nrightarrow X$   
 $Y \rightarrow Z$ , 则称 $Z$ 对 $X$ 传递函数依赖。

记为:  $X \xrightarrow{\text{传递}} Z$

注: 如果 $Y \rightarrow X$ , 即 $X \longleftrightarrow Y$ , 则 $Z$ 直接依赖于 $X$ 。

例: 在关系 $Std(Sno, Sdept, Mname)$ 中, 有:

$Sno \rightarrow Sdept$ ,  $Sdept \rightarrow Mname$

$Mname$ 传递函数依赖于 $Sno$

## 6.3 数据依赖的公理系统

❖ 定义6.11 对于满足一组函数依赖 $F$ 的关系模式  $R \langle U, F \rangle$ ，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 $F$ 逻辑蕴涵 $X \rightarrow Y$ ，记为 $F \models X \rightarrow Y$ 。

❖ 例：  $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$



# 数据依赖的公理系统



## ❖ Armstrong公理系统

■ 一套推理规则，是模式分解算法的理论基础

■ 用途

● 求给定关系模式的码

● 从一组函数依赖求得蕴涵的函数依赖



# 函数依赖的推理规则

Armstrong公理:

假设 $U$ 是 $R$ 的所有属性集合

1. 自反律

如果 $Y \subseteq X \subseteq U$ , 则 $X \rightarrow Y$ 成立 (这是平凡依赖)

2. 扩展律

如果 $X \rightarrow Y$ 成立,  $Z \subseteq U$ , 则 $ZX \rightarrow ZY$ 成立

3. 传递律

如果 $X \rightarrow Y$ ,  $Y \rightarrow Z$ 成立, 则 $X \rightarrow Z$ 成立。

三条推理规则:

1. 合并规则  $\{X \rightarrow Y, X \rightarrow Z\}$  蕴含  $X \rightarrow YZ$

2. 伪传递规则  $\{X \rightarrow Y, WY \rightarrow Z\}$  蕴含  $XW \rightarrow Z$

3. 分解规则 如果 $X \rightarrow YZ$ , 则 $X \rightarrow Z$ ,  $X \rightarrow Y$ 成立。

## 一些概念和记号(c)

- 函数依赖集合**F**的闭包(定义**6.12**)
  - 函数依赖集合**F**所逻辑蕴涵的函数依赖的全体。记作**F<sup>+</sup>**。例： 如果  $A \rightarrow B$ ,  $B \rightarrow C$ ,  $C \rightarrow D$ ,  $D \rightarrow E$ ,  $E \rightarrow F$ ,  $F \rightarrow G$ ,  $G \rightarrow H$  则  $F^+ = A \rightarrow (\text{any non-empty subset of } A, B, C, D, E, F, G), B \rightarrow (\text{any non-empty subset of } B, C, D, E, F, G), \dots$

## F的闭包

$F = \{X \rightarrow Y, Y \rightarrow Z\}$

$F^+ = \{$

$X \rightarrow X,$	$Y \rightarrow Y,$	$Z \rightarrow Z,$	$XY \rightarrow X,$	$XZ \rightarrow X,$	$YZ \rightarrow Y,$	$XYZ \rightarrow X,$
$X \rightarrow Y,$	$Y \rightarrow Z,$		$XY \rightarrow Y,$	$XZ \rightarrow Y,$	$YZ \rightarrow Z,$	$XYZ \rightarrow Y,$
$X \rightarrow Z,$	$Y \rightarrow YZ,$		$XY \rightarrow Z,$	$XZ \rightarrow Z,$	$YZ \rightarrow YZ,$	$XYZ \rightarrow Z,$
$X \rightarrow XY,$			$XY \rightarrow XY,$	$XZ \rightarrow XY,$		$XYZ \rightarrow XY,$
$X \rightarrow XZ,$			$XY \rightarrow YZ,$	$XZ \rightarrow XZ,$		$XYZ \rightarrow YZ,$
$X \rightarrow YZ,$			$XY \rightarrow XZ,$	$XZ \rightarrow XY,$		$XYZ \rightarrow XZ,$
$X \rightarrow ZYZ,$			$XY \rightarrow XYZ,$	$XZ \rightarrow XYZ,$		$XYZ \rightarrow XYZ\}$

$F = \{X \rightarrow A_1, \dots, X \rightarrow A_n\}$ 的闭包 $F^+$ 计算是一个NP完全问题

# 闭包



- $R = (A, B, C, G, H, I)$

- $F = \{A \rightarrow B$   
 $A \rightarrow C$   
 $CG \rightarrow H$   
 $CG \rightarrow I$   
 $B \rightarrow H\}$



- some members of  $F^+$



- $A \rightarrow H$

- $AG \rightarrow I$

- $CG \rightarrow HI$



# 属性集的闭包

定义6.13:

给定表T中的一组属性A和函数依赖F，属性集A的闭包为A  
可以函数决定的最大属性集

算法(6.1):

输入: X, F

输出: result:=X<sup>+</sup>

方法:

result:=X;

while (result 发生变化) do

for each  $v \rightarrow w$  in F do

begin

if  $v \subseteq \text{result}$  then result:=result  $\cup$  w;

end

该算法的特点:穷举搜索了所有可能。

# 计算属性集的闭包

■ 例：  $F = \{AB \rightarrow C, D \rightarrow EG, C \rightarrow A, BE \rightarrow C, BC \rightarrow D, CG \rightarrow BD, ACD \rightarrow B, CE \rightarrow AG\}$

计算  $(BD)^+$

1.  $Result = BD$

2、找左部为result 的函数依赖，

由于  $D \rightarrow EG$ ,  $result = result \cup EG = BDEG$

3、循环继续

由于  $BE \rightarrow C$ ,  $result = BCDEG$

由于  $C \rightarrow A$ ,  $BC \rightarrow D$ ,  $CG \rightarrow BD$ ,  $CE \rightarrow AG$ ,  $result = ABCDEG$

$(BD)^+ = ABCDEG$

# 属性集闭包算法的几种用途

- 1. 判断超键:
  - To test if  $\alpha$  is a superkey, we compute  $\alpha^+$ , and check if  $\alpha^+$  contains all attributes of  $R$ .
- 2. 测试函数依赖
  - To check if a functional dependency  $\alpha \rightarrow \beta$  holds (or, in other words, is in  $F^+$ ), just check if  $\beta \subseteq \alpha^+$ .
  - Is a simple and cheap test, and very useful
- 3. 计算F的闭包
  - For each  $\gamma \subseteq R$ , we find the closure  $\gamma^+$ , and for each  $S \subseteq \gamma^+$ , we output a functional dependency  $\gamma \rightarrow S$ .



# 属性闭包的应用

- 要判断 $X \rightarrow Y$ 是否是一个关系模式的函数依赖，可以判 $X \rightarrow Y$ 是否属于 $F^+$ ，也可以判 $Y$ 是否是 $X^+$ 的子集。
- 练习：存在关系模式 $R(ABCDE)$ 和函数依赖集  $F = \{AC \rightarrow B, C \rightarrow D, AD \rightarrow E, E \rightarrow A\}$ ，问 $AC \rightarrow E$ 是否是 $R$ 的一个函数依赖？

要点：

👉 找 $(AC)^+ = (ABCDE)$ ， $AC$ 关于 $F$ 的闭包。

👉 判断， $E$ 是否为其子集，即： $E \subseteq (AC)^+$

👉 则 $AC \rightarrow E$ 是其函数依赖。

## 练习:

- 已知:  $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B$   
 $A \rightarrow C$   
 $CG \rightarrow H$   
 $CG \rightarrow I$   
 $B \rightarrow H\}$
- 利用属性闭包算法判断AG是否是R的候选键?

思路: 求 $(AG)^+$

1.  $result = AG$
2.  $result = ABCG$  ( $A \rightarrow C$  and  $A \rightarrow B$ )
3.  $result = ABCGH$  ( $CG \rightarrow H$  and  $CG \subseteq AGBC$ )
4.  $result = ABCGHI$  ( $CG \rightarrow I$  and  $CG \subseteq AGBCH$ )

■ Is AG a candidate key?

1. Is AG a super key?

1. Does  $AG \rightarrow R$ ? == Is  $(AG)^+ \supseteq R$

2. Is any subset of AG a superkey?

1. Does  $A \rightarrow R$ ? == Is  $(A)^+ \supseteq R$

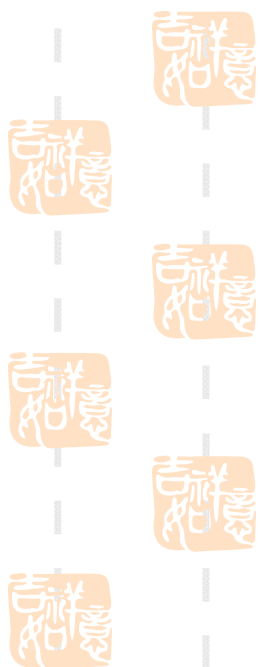
2. Does  $G \rightarrow R$ ? == Is  $(G)^+ \supseteq R$

👉 To test if  $\alpha$  is a superkey, we compute  $\alpha^+$ , and check if  $\alpha^+$  contains all attributes of  $R$ .

👉 若这个超键是最小的, 则为候选键。

# 如何找出关系模式的候选键？

从单个属性开始检查，其关于  $F$  的闭包是否为全属性集。



# 一组概念和结论



- 函数依赖集  $F$  覆盖  $G$ ，是指  $F^+ \supseteq G$ 。
- 函数依赖集  $F$  和  $G$  等价，是指  $F^+ = G^+$ ，也称为互相覆盖。
- 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。

■ 结论：  $F^+ = G^+$  的必要条件是  $F \subseteq G^+$  和  $G \subseteq F^+$ 。

➤ 如何证明  $F \subseteq G^+$ ？

■ 将  $F$  中每个函数依赖逐个取出，判断  $X \rightarrow Y \subseteq G^+$

■ 结论：任何函数依赖集  $F$  总可以为一些右部都是单属性的函数依赖集所覆盖

➤ 单属性：只有一个字段（属性）

# 一组概念和结论 (C)

- 定义6.15
- $F$ 是一个极小函数依赖集，如果：
  - (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。
  - (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。
- 定理 任何一个函数依赖 $F$ 都和一个最小函数依赖集 $F'$ 等价， $F'$ 称为 $F$ 的最小覆盖

# 最小依赖集



[例2] 关系模式  $S\langle U, F \rangle$ , 其中:

$U = \{ \text{Sno}, \text{Sdept}, \text{Mname}, \text{Cno}, \text{Grade} \},$

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

设  $F' = \{ \text{Sno} \rightarrow \text{Sdept}, \text{Sno} \rightarrow \text{Mname}, \text{Sdept} \rightarrow \text{Mname},$

$(\text{Sno}, \text{Cno}) \rightarrow \text{Grade}, (\text{Sno}, \text{Sdept}) \rightarrow \text{Sdept} \}$

$F$  是最小覆盖, 而  $F'$  不是。

因为:  $F' - \{ \text{Sno} \rightarrow \text{Mname} \}$  与  $F'$  等价

$F' - \{ (\text{Sno}, \text{Sdept}) \rightarrow \text{Sdept} \}$  也与  $F'$  等价

# 极小化过程

(1)逐一检查 $F$ 中各函数依赖 $FD_i: X \rightarrow Y$ , 若 $Y=A_1A_2 \dots A_k$ ,  $k > 2$ , 则用  $\{X \rightarrow A_j | j=1, 2, \dots, k\}$  来取代 $X \rightarrow Y$ 。

(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。

(3)逐一取出 $F$ 中各函数依赖 $FD_i: X \rightarrow A$ , 设 $X=B_1B_2 \dots B_m$ , 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 $X$ 。

# 求解极小函数依赖集

- $F = \{ABD \rightarrow AC, C \rightarrow BE, AD \rightarrow BF, B \rightarrow E\}$
- Step 1. Create an equivalent set  $H$  of FDs, with only one attributes on the right side.
  - $H = \{ABD \rightarrow A, ABD \rightarrow C, C \rightarrow B, C \rightarrow E, AD \rightarrow B, AD \rightarrow F, B \rightarrow E\}$
- Step 2. Remove individual FDs that are inessential in  $H$ 
  - $ABD \rightarrow A$  is trivial
  - $H = \{ABD \rightarrow C, C \rightarrow B, C \rightarrow E, AD \rightarrow B, AD \rightarrow F, B \rightarrow E\}$
- 1. Remove  $ABD \rightarrow C$ ?
  - $ABD^+ = ABDFE$

No



➤  $H = \{ABD \rightarrow C, C \rightarrow B, C \rightarrow E, AD \rightarrow B, AD \rightarrow F, B \rightarrow E\}$

2. Remove  $C \rightarrow B$ ?

No

$C^+ = CE$

3. Remove  $C \rightarrow E$ ?

Yes

$C^+ = CBE$

$H = \{ABD \rightarrow C, C \rightarrow B, AD \rightarrow B, AD \rightarrow F, B \rightarrow E\}$

4. Remove  $AD \rightarrow B$ ?

No

$AD^+ = ADF$

5. Remove  $AD \rightarrow F$

No

$AD^+ = ADBEC$

$$H = \{ABD \rightarrow C, C \rightarrow B, AD \rightarrow B, AD \rightarrow F, B \rightarrow E\}$$

6. Remove  $B \rightarrow E$ ?

No

$$B^+ = B$$

**Step 3 : Replace individual FDs with FDs that have a smaller number of attributes**

1. Replace  $ABD \rightarrow C$  by  $BD \rightarrow C$  ?

No

$$BD^+ = BDE$$

2. Replace  $ABD \rightarrow C$  by  $AD \rightarrow C$  ?

Yes

$$AD^+ = ADBFEC$$

$$H = \{AD \rightarrow C, C \rightarrow B, AD \rightarrow B, AD \rightarrow F, B \rightarrow E\}$$

3. Replace  $AD \rightarrow C$  by  $A \rightarrow C$  ?

No

$$A^+ = A$$

Rewrite  $H = \{AD \rightarrow C, C \rightarrow B, AD \rightarrow B, AD \rightarrow F, B \rightarrow E\}$

4. Replace  $AD \rightarrow B$  by  $D \rightarrow B$  ?

No

$$D^+ = D$$

5. Replace  $AD \rightarrow B$  by  $A \rightarrow B$  ?

No

$$A^+ = A$$

6. Replace  $AD \rightarrow F$  by  $D \rightarrow F$  ?

No

$$D^+ = D$$

7. Replace  $AD \rightarrow F$  by  $A \rightarrow F$  ?

No

$$A^+ = A$$

Rewrite  $H = \{AD \rightarrow C, C \rightarrow B, AD \rightarrow B, AD \rightarrow F, B \rightarrow E\}$ .

Repeat step 2:

1. Remove  $AD \rightarrow C$  ?

No

$AD^+ = ADBEF$

2. Remove  $C \rightarrow B$ ?

No

$C^+ = C$

3. Remove  $AD \rightarrow B$ ?

Yes

$AD^+ = ADCBEF$

$H = \{AD \rightarrow C, C \rightarrow B, AD \rightarrow F, B \rightarrow E\}$ .

4. Remove  $AD \rightarrow F$ ?

No

$AD^+ = ADCBE$

$H = \{AD \rightarrow C, C \rightarrow B, AD \rightarrow F, B \rightarrow E\}$ .

Rewrite  $H = \{AD \rightarrow C, C \rightarrow B, AD \rightarrow F, B \rightarrow E\}$ .

5. Remove  $B \rightarrow E$  ?

No

$$B^+ = B$$

Step 4: Use union rule to gather all FDs with equal left-hand sides

➤ Union  $AD \rightarrow C$  and  $AD \rightarrow F$  by  $AD \rightarrow CF$

■ The minimal cover is:

➤  $H = \{AD \rightarrow CF, C \rightarrow B, B \rightarrow E\}$

# 求解极小函数依赖集

- 例：  $F = \{AB \rightarrow C, C \rightarrow A, BC \rightarrow D, ACD \rightarrow B, D \rightarrow EG, BE \rightarrow C, CG \rightarrow BD, CE \rightarrow AG\}$ 
  - 1. 将右属性分解为单属性
    - $F = \{AB \rightarrow C, C \rightarrow A, BC \rightarrow D, ACD \rightarrow B, D \rightarrow E, D \rightarrow G, BE \rightarrow C, CG \rightarrow B, CG \rightarrow D, CE \rightarrow A, CE \rightarrow G\}$
  - 2、去掉冗余的函数依赖。
    - 方法：让  $F' = F - \{X \rightarrow A\}$ ，依次检测每个  $X \rightarrow A \subseteq F'^+$ ，即A是否属于X关于F'闭包。
      - 可检测到  **$CG \rightarrow B, CE \rightarrow A$**  是冗余的，可去掉。
    - $F = \{AB \rightarrow C, C \rightarrow A, BC \rightarrow D, ACD \rightarrow B, D \rightarrow E, D \rightarrow G, BE \rightarrow C, CG \rightarrow D, CE \rightarrow G\}$
  - 3、去掉左边多余的属性。( $F^+ = (F - \{X \rightarrow A\} \cup \{X - B_i \rightarrow A\})^+$ )
    - 方法：逐个去除每个  $X \rightarrow A$ （若X是多属性）中X的每个属性  $B_i$ ，检查剩下的函数依赖  $X - B_i \rightarrow A$  是否属于F的函数依赖，即  $A \in (X - B_i)^+_F$
    - 可检测到  **$ACD \rightarrow B$**  中A属性是多余的。则可去掉属性A。
      - $F = \{AB \rightarrow C, C \rightarrow A, BC \rightarrow D, CD \rightarrow B, D \rightarrow E, D \rightarrow G, BE \rightarrow C, CG \rightarrow D, CE \rightarrow G\}$
  - 以上 **F** 为极小函数依赖集。

## 练习

### 1. 求极小函数依赖集

■  $\{A \rightarrow B, B \rightarrow C, A \rightarrow CD\}$

$\{A \rightarrow B, B \rightarrow C, A \rightarrow D\}$

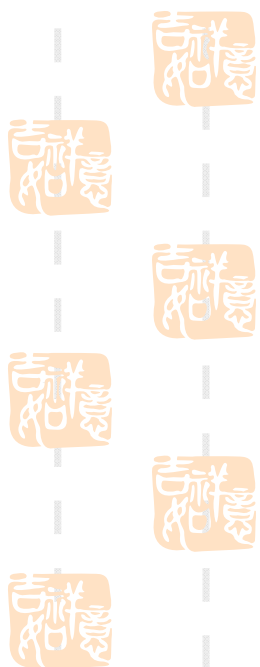
■  $\{A \rightarrow B, B \rightarrow C, AC \rightarrow D\}$

$\{A \rightarrow B, B \rightarrow C, A \rightarrow D\}$

2.  $F = \{AB \rightarrow E, AC \rightarrow F, AD \rightarrow BF, B \rightarrow C, C \rightarrow D\}$ , 证明  $AC \rightarrow F$  是冗余的.

## 数据依赖的公理系统（续）

- $F$  的最小依赖集  $F_m$  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。

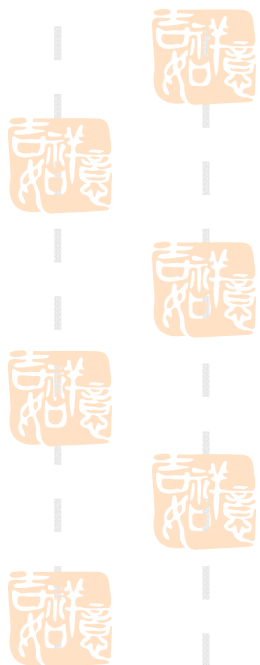




# 无损分解

- 当一张表  $T$  被分解为表  $T_1$  ,  $T_2$  , ...  $T_k$  时, 如果对于表  $T$  任何可能的元组,

$T = T_1 \bowtie T_2 \bowtie \dots \bowtie T_k$  成立, 则称这种分解为无损分解



## Example

- Head( $T$ ) = (A, B, C)

A	B	C
a1	100	c1
a2	200	c2
a3	300	c3
a4	200	c4

- Head( $T_1$ ) = (A, B) Head( $T_2$ ) = (B, C)

A	B
a1	100
a2	200
a3	300
a4	200

B	C
100	c1
200	c2
300	c3
200	c4

- AB JOIN BC

A	B	C
a1	100	c1
a2	200	c2
a2	200	c4
a3	300	c3
a4	200	c2
a4	200	c4

*We lost  
information!*

**Lossy-join decompositions  
result in information loss.**

**Note:** Loss. refers not to the loss of tuples, but to the loss of information Or, the ability to distinguish different original relations

# 模式的分解（续）

例：S-L (Sno, Sdept, Sloc)

$F = \{ Sno \rightarrow Sdept, Sdept \rightarrow Sloc \}$

sno	sdept	sloc
s1	cs	c10
s2	cs	c10
s3	ma	c8
s4	ma	c8
s5	ph	c8

不具有无损连接性，  
也未保持函数依赖

不保持函数依赖，不  
具有无损连接性

保持无损连接性，但  
未保持函数依赖

具有无损连接性，  
又保持了函数依赖

分解方法可以有多种：

1. S-L分解为三个关系模式：SN(Sno)  
SD(Sdept)  
SO(Sloc)

2. SL分解为下面二个关系模式：NL(Sno, Sloc),  
L(Sdept, Sloc)

3. SL分解为下面二个关系模式：ND(Sno, Sdept),  
NL(Sno, Sloc)

4. SL分解为下面二个关系模式：ND(Sno, Sdept),  
NL(Sdept, Sloc)

# 模式的分解



- 如果一个分解具有无损连接性，则它能够保证不丢失信息
- 如果一个分解保持了函数依赖，则它可以减轻或解决各种异常情况
- 分解具有无损连接性和分解保持函数依赖是两个互相独立的标准。

具有无损连接性的分解不一定能够保持函数依赖；同样，保持函数依赖的分解也不一定具有无损连接性。



# 分解的准则

- 1 满足无损
- 2 且保持依赖关系

无损和保持依赖地分解一个关系，可以避免用户的操作出现更新异常，同时能够保持原有的模式数据中实际存在的约束关系。

# 判断无损分解的算法(算法6.2)

- 对于 $R(A_1, A_2, \dots, A_n)$ ,  $R$ 的函数依赖集是 $F$ ,  $R$ 的分解 $\rho = \{R_1, R_2, \dots, R_k\}$ , 要判断 $\rho$ 是 $R$ 的一个无损分解
- 判断方法: 建立如下的矩阵

	$A_1$	$A_2$	...	$A_j$	...	$A_n$
$R_1$						
$R_2$						
...						
$R_i$				$M[i, j]$		
...						
$R_k$						

其中 $M[i, j] = \begin{cases} a_j & \text{如果 } A_j \in R_i \\ b_{ij} & \text{如果 } A_j \notin R_i \end{cases}$

# 判断无损分解的算法 (C)

	A1	A2	...	Aj	...	An
R1						
R2						
...						
Ri	a1	a2	...	aj	....	an
Rk						

对F中的每一个函数依赖 $X \rightarrow Y$ ，找出矩阵中在X字段上相等的行，把他们Y字段上的值变成一样：如果有一字段是 $a_j$ ，则其它行的同一字段的值也改为 $a_j$ ；如果是 $b_{ij}$ ，就全部改为某 $b_{ij}$ 值，反复施加这样的修改，直到矩阵不再改变。

如果结果矩阵中有一行是 $a_1, a_2, \dots, a_n$ ，则 $\rho$ 是无损分解

# 判断无损连接分解实例1

例1:  $R\langle U, F \rangle$ ,  $U = \{A, B, C, D, E\}$ ,  
 $F = \{AB \rightarrow C, C \rightarrow D, D \rightarrow E\}$ ,  $R$ 的一个分解  
为  $R_1(A, B, C)$ ,  $R_2(C, D)$ ,  $R_3(D, E)$ 。

解：我们来看算法的动画演示。



# 例1 第

$A \in \{A, B, C\}$   
因此此处填 $a_1$

D不属于 $\{A, B, C\}$   
因此此处填 $b_{14}$

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

$U_i$

# 例1

对于属性组 {A,B,C} 有任一

对于C上, (a3) 又因为

对于D→E来说, 在D属性组上, 第1, 2, 3行的值相等, 因此要修改E的值, 又因为在E属性上存在a5, 因此把b15, b25修改为a5

	A	B	C	D	E
{A,B,C}	a <sub>1</sub>	a <sub>2</sub>	a <sub>3</sub>	a <sub>4</sub>	a <sub>5</sub>
{C,D}	b <sub>21</sub>	b <sub>22</sub>	a <sub>3</sub>	a <sub>4</sub>	a <sub>5</sub>
{D,E}	b <sub>31</sub>	b <sub>32</sub>	b <sub>33</sub>	a <sub>4</sub>	a <sub>5</sub>

## 判断无损连接分解实例2

- $R(ABCDE), F = \{A \rightarrow C, B \rightarrow C, C \rightarrow D, DE \rightarrow C, CE \rightarrow A\}$ ,  $R$  分解为  $R_1(AD), R_2(AB), R_3(BE), R_4(CDE), R_5(AE)$  是否是无损分解

# 初始

	A	B	C	D	E
R1(AD)	a1	b12	b13	a4	b15
R2(AB)	a1	a2	b23	b24	b25
R3(BE)	b31	a2	b33	b34	a5
R4(CDE)	b41	b42	a3	a4	a5
R5(AE)	a1	b52	b53	b54	a5

$A \rightarrow C$

	A	B	C	D	E
R1(AD)	a1	b12	b13	a4	b15
R2(AB)	a1	a2	b13	b24	b25
R3(BE)	b31	a2	b33	b34	a5
R4(CDE)	b41	b42	a3	a4	a5
R5(AE)	a1	b52	b13	b54	a5

$B \rightarrow C$

	A	B	C	D	E
R1(AD)	a1	b12	b13	a4	b15
R2(AB)	a1	a2	b13	b24	b25
R3(BE)	b31	a2	b13	b34	a5
R4(CDE)	b41	b42	a3	a4	a5
R5(AE)	a1	b52	b13	b54	a5

C → D

	A	B	C	D	E
R1(AD)	a1	b12	b13	a4	b15
R2(AB)	a1	a2	b13	a4	b25
R3(BE)	b31	a2	b13	a4	a5
R4(CDE)	b41	b42	a3	a4	a5
R5(AE)	a1	b52	b13	a4	a5

DE→C

	A	B	C	D	E
R1(AD)	a1	b12	b13	a4	b15
R2(AB)	a1	a2	b13	a4	b25
R3(BE)	b31	a2	a3	a4	a5
R4(CDE)	b41	b42	a3	a4	a5
R5(AE)	a1	b52	a3	a4	a5



CE→A之后m不再变化，停止，判断为无损分解

	A	B	C	D	E
R1(AD)	a1	b12	b13	a4	b15
R2(AB)	a1	a2	b13	a4	b25
R3(BE)	<u>a1</u>	<u>a2</u>	<u>a3</u>	a4	<u>a5</u>
R4(CDE)	a1	b42	a3	a4	a5
R5(AE)	a1	b52	a3	a4	a5

# 判断无损分解的一个简单算法

## 定理 6.5

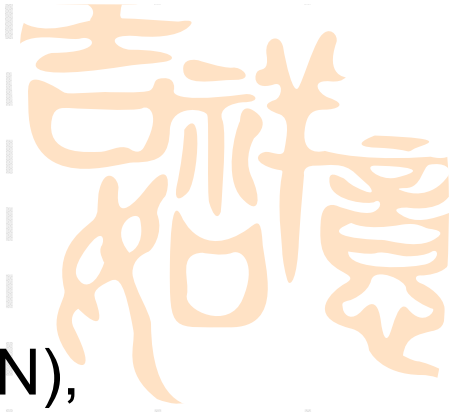
设 $\rho=\{R_1(U_1), R_2(U_2)\}$ 是 $R(U)$ 的一个分解，则 $\rho$ 为无损分解的充分必要条件为

$$(U_1 \cap U_2) \rightarrow (U_1 - U_2) \text{ 或 } (U_1 \cap U_2) \rightarrow (U_2 - U_1)$$

注意：仅当一个关系被分解为两个子模式时用

练习：

- 有关系模式： $R(CN, TN, D)$ ，函数依赖集 $F=\{CN \rightarrow TN, TN \rightarrow D\}$ ，分解
- $R_1=(CN, TN), R_2(TN, D)$
- 试证： $P=(R_1, R_2)$ 是 $R$ 的无损分解。



解:  $U1 = (CN, TN)$ ,  $U2 = (TN, D)$ ,  $U1 \cap U2 = (TN)$ ,  
 $U1 - U2 = (CN)$   
 $U2 - U1 = (D)$

因为:  $TN \rightarrow D$ , 故  $U1 \cap U2 \rightarrow U2 - U1$  成立, 根据定理6.5, 可知:  **$P = (R1, R2)$**  分解是无损的

。





例:  $R = (CN, TN, D)$

$F = \{CN \rightarrow TN, TN \rightarrow D\}$

如果分解为  $R_1 = (CN, TN)$ 、 $R_2 = (CN, D)$ ，则无损，但不保持依赖(丢失了  $TN \rightarrow D$  的函数依赖)

如果分解为  $R_1 = (CN, TN)$ 、 $R_2 = (TN, D)$ ，则无损，也保持依赖

## 6.2.3 范式

- 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:



- 第一范式(1NF)
- 第二范式(2NF)
- 第三范式(3NF)
- BC范式(BCNF)
- 第四范式(4NF)
- 第五范式(5NF)

## 6.2.3 范式

- 各种范式之间存在联系：

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

- 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

# 1NF



- 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库

- 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



# 1NF的例子

- 关系的属性值都是不可再分的原子值
  - Emp\_info(emp\_id, emp\_name, emp\_phone, dept\_name, dept\_phone, dept\_mgrname, skill\_id, skill\_name, skill\_date, skill\_lvl)

➤ 非第一范式的数据模式的例子如下：

学号	姓名	家庭成员
----	----	------

父亲，母亲，  
...

Set-value





### **emp\_info**

<b>emp_id</b>	<b>emp_name</b>	<b>...</b>	<b>skill_id</b>	<b>skill_name</b>	<b>skill_date</b>	<b>skill_lvl</b>
09112	Jones	...	44	librarian	03-15-99	12
09112	Jones	...	26	PC-admin	06-30-98	10
09112	Jones	...	89	word-proc	01-15-00	12
12231	Smith	...	26	PC-admin	04-15-99	5
12231	Smith	...	39	bookkeeping	07-30-97	7
13597	Brown	...	27	statistics	09-15-99	6
14131	Blake	...	26	PC-admin	05-30-98	9
14131	Blake	...	89	word-proc	09-30-99	10
...	...	...	...	...	...	...



## 第二范式 (2NF)

- 一个关系模式中，如果所有的非主属性都完全依赖于一个候选键，则称为第二范式

➤ Emp\_info(emp\_id, emp\_name, emp\_phone, dept\_name, dept\_phone, dept\_mgrname, skill\_id, skill\_name, skill\_date, skill\_lvl)

➤ emp\_info中的函数依赖

➤ emp\_id → emp\_name emp\_phone dept\_name

➤ dept\_name → dept\_phone dept\_mgrname

➤ skill\_id → skill\_name

➤ emp\_id, skill\_id → skill\_date skill\_lvl

- Key: emp\_id, skill\_id

➤ emp\_name只依赖于键的一部分emp\_id (部分依赖)

- 产生更新异常

## 第二范式 (2NF) (续)

- 第二范式消除了非主属性对键的部分函数依赖
- 关系模式只满足2NF, 仍存在问题
- 例:
- Emps(emp\_id, emp\_name, emp\_phone, dept\_name, dept\_phone, dept\_mgrname)
  - Key: emp\_id
- Emp\_skills(emp\_id, skill\_id, skill\_date, skill\_lvl)
  - Key: emp\_id, skill\_id
- Skills(skill\_id, skill\_name)
  - Key: skill\_id
- $R \in 2NF$ , 但仍出现冗余度大和更新异常问题
- 根源是存在非主属性对主属性的传递函数依赖。

## 第三范式 (3NF)



- 满足第二范式2NF
- 如果关系模式R中的所有非主属性对任何候选关键字都不存在传递依赖，则称关系R是属于第三范式的。即消除传递依赖

■ 非第三范式例子：

➤ (书号，书名，作者名，作者工作单位)，这是一个第二范式，但存在作者名 → 作者工作单位，这是非主属性之间依赖关系。



# Employee Information Schema in 3NF

- Emps(emp\_id,emp\_name,emp\_phone,dept\_name)
  - Key: emp\_id
- Depts(dept\_name,dept\_phone,dept\_mgrname)
  - Key: dept\_name
- Emp\_skills(emp\_id,skill\_id,skill\_date,skill\_lvl)
  - Key: emp\_id,skill\_id
- Skills(skill\_id,skill\_name)
  - Key: skill\_id

## 第三范式 (3NF) (续)



- 关系模式只满足3NF，仍存在异常。
- 例如：(S, C, Z)，S代表街道，C代表城市，Z代表邮政编码。函数依赖集 $F=\{SC \rightarrow Z, Z \rightarrow C\}$ ，SC和ZS是键。属第三范式，但不属BCNF。
- 不能只插入邮政编码和城市，还必须插入街道。



# BC范式 (BCNF)

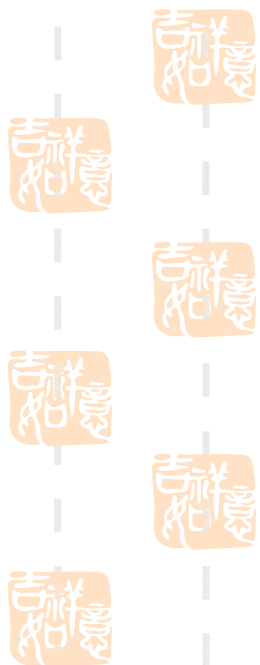


- 一个关系模式中，对任意的非平凡函数依赖 $X \rightarrow A$ ，必有 $X$ 是超键，则称为BC范式。
  - 和第三范式相比，要求 模式中的非主属性要完全依赖于一个键（所以BCNF都属于3NF），另外，主属性不能部分依赖或传递依赖于键
  - 属3NF，但不属BCNF的关系模式很少，有时统称为第三范式



## BC范式 (BCNF)

- 将上例 $R(S, C, Z)$  分解为 $R_1(Z, C)$ 和 $R_2(S, Z)$





# 总 结

- 判断一个关系模式的规范程度，关键是要找出所有的函数依赖关系，然后找出所有的候选键，并根据定义判断。
  - B C N F — 所有决定子是超键。
  - 3 N F — 消除了部分依赖和传递依赖。
  - 2 N F — 消除了部分依赖，存在传递依赖
  - 1 N F — 每个数据项都是原子的，允许有部分依赖

# 练习



- 判断下列关系属于第几范式
  - $R(ABCD), F = \{B \rightarrow D, AB \rightarrow C\}$ 。
  - $R(ABCDE), F = \{AB \rightarrow CE, E \rightarrow AB, C \rightarrow D\}$ 。



## BCNF (续)



[例5] 关系模式C (Cno, Cname, Pcnno)

■  $C \in 3NF$

■  $C \in BCNF$



# BCNF (续)



[例8]在关系模式STJ (S, T, J) 中, S表示学生, T表示教师, J表示课程。每个老师只教一门课,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。



➤ 函数依赖:



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



➤ (S, J)和(S, T)都是候选码



# BCNF (续)



■  $STJ \in 3NF$

➤ 没有任何非主属性对码传递依赖或部分依赖



■  $STJ \notin BCNF$



➤ T是决定因素，T不包含码



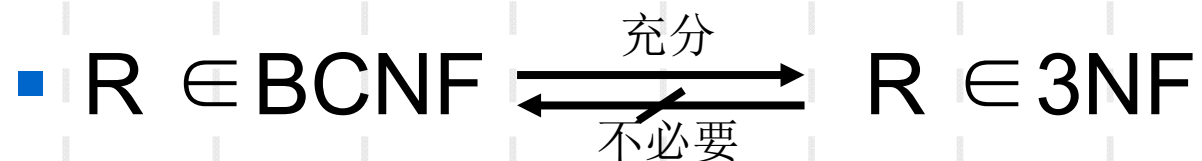
## BCNF (续)

- 解决方法：将STJ分解为二个关系模式：  
 $ST(S, T) \in BCNF, TJ(T, J) \in BCNF$

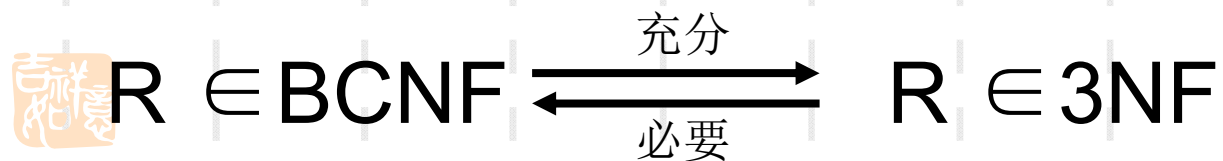
没有任何属性对码的部分函数依赖和传递函数依赖



## 3NF与BCNF的关系



■ 如果  $R \in 3\text{NF}$ ，且  $R$  只有一个候选码



# 分解为3NF的算法(算法6.3)

- 分解为3NF并保持依赖
- (1)  $F$ 的最小覆盖仍记为 $F$ ，找出不在 $F$ 中出现的 $R$ 中属性，组成一个关系模式，将其从 $R$ 中分离，其余属性仍组成的关系模式仍记为 $R$ ，属性集记为 $U$
- (2) 若最小覆盖 $F$ 中有一函数依赖含有 $R$ 中的所有属性，则把整个 $R$ 作为输出，算法结束，否则进入 (3)
- (3) 对于 $F$ 中所有函数依赖 $X \rightarrow A$ ，将 $XA$ 作为 $\sigma$ 中的一个关系模式输出，但如果 $F$ 中有函数依赖 $X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_n$ ，可将 $XA_1 \dots A_n$ 作为一个关系模式输出

- $R(ABCDE) \quad F=\{A \rightarrow B, C \rightarrow BD\}$

➤  $R_1(E)$

➤  $R_2(AB)$

➤  $R_3(BCD)$



# 分解为3NF的算法（算法6.4）

- 无损分解为3NF并保持依赖
- （1）首先用上述算法求出分解R为3NF并保持依赖的分解 $\sigma$ ，设 $\sigma = \{R_1, R_2, \dots, R_k\}$
- （2）设X为R的一个键，则  $\tau = \sigma \cup \{R(X)\}$ ，消除重复。

## 分解为3NF的算法（例）

- $R(M, N, P, C, S, Z)$
- $F = \{M \rightarrow N, M \rightarrow P, M \rightarrow C, M \rightarrow S, M \rightarrow Z, \{P, C, S\} \rightarrow Z, Z \rightarrow P, Z \rightarrow C\}$
- 求F的最小覆盖，得  $F = F - \{M \rightarrow Z\}$
- 根据算法求得  $\sigma = \{(M, N, P, C, S), (P, C, S, Z), (Z, P, C)\}$
- 因R的键为M，故可得
- $\tau = \{(M, N, P, C, S), (P, C, S, Z), (Z, P, C), (M)\}$
- 消除多余关系(M)及(Z, P, C)，得
- $\tau' = \{(M, N, P, C, S), (P, C, S, Z)\}$

# 分解为BCNF的算法(算法6.5)

- 无损分解为BCNF
- (1) 初始化 $\rho$ 为 $\{R\}$
- (2) 若 $S$ 为 $\rho$ 中的一个非BCNF关系模式，则 $S$ 必存在非平凡函数依赖 $X \rightarrow A$ ，其中 $X$ 不是 $S$ 的超键。将 $S$ 分解为 $S_1(XA)$ 和 $S_2(U_S - A)$ 。

➤ 分解是无损的。

- (3) 如此反复，直到 $\rho$ 中所有关系模式都是BCNF为止

- 注： $X \rightarrow A$ 是包含在 $F^+$ 中的，因此该算法涉及到求 $F^+$

- 此算法不能保证保持依赖

## 6.2.9 规范化小结

- 关系数据库的规范化理论是数据库逻辑设计的工具
- 目的：尽量消除插入、删除异常，修改复杂，数据冗余
- 基本思想：逐步消除数据依赖中不合适的部分
- 实质：概念的单一化

# 规范化小结（续）



## ■ 关系模式规范化的基本步骤



## 规范化小结（续）



- 不能说规范化程度越高的关系模式就越好
- 在设计数据库模式结构时，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



# 第六章 关系数据理论



## 6.1 问题的提出

## 6.2 规范化

## 6.3 数据依赖的公理系统



## \*6.4 模式的分解



## 6.5 小结



## 6.5 小结



- 规范化理论为数据库设计提供了理论的指南和工具

- 也仅仅是指南和工具



- 并不是规范化程度越高，模式就越好



- 必须结合应用环境和现实世界的具体情况合理地选择数据库模式





# 作业

2

补充题:

1. 设有关系R (ABCDEG), 其函数依赖集为:

$F = \{E \rightarrow D, C \rightarrow B, CE \rightarrow G, B \rightarrow A\}$

请回答下列问题:

(1) R最高属于第几范式?

(2) 分解R为3NF,

(3) 分解R为BCNF,

(4) 请验证 $\rho = \{R_1(DE), R_2(BC), R_3(CEG), R_4(AB)\}$ 是否是R的一个无损分解。

2. 试问下列关系模式最高属于第几范式,并解释原因

1)  $R(ABCD), F = \{B \rightarrow D, AB \rightarrow C\}$ 。

2)  $R(ABCDE), F = \{AB \rightarrow CE, E \rightarrow AB, C \rightarrow D\}$ 。

3)  $R(ABC), F = \{A \rightarrow B, B \rightarrow A, A \rightarrow C\}$ 。