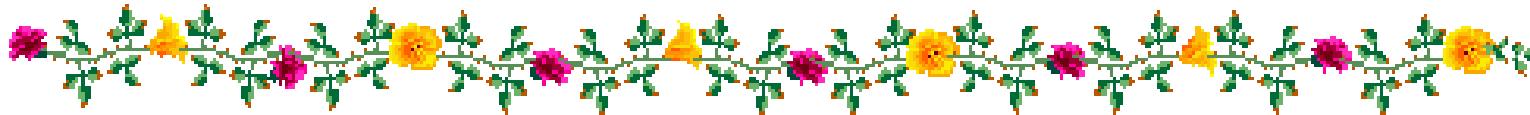
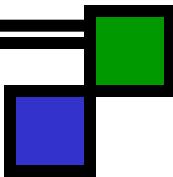


# 第六章 关系数据理论





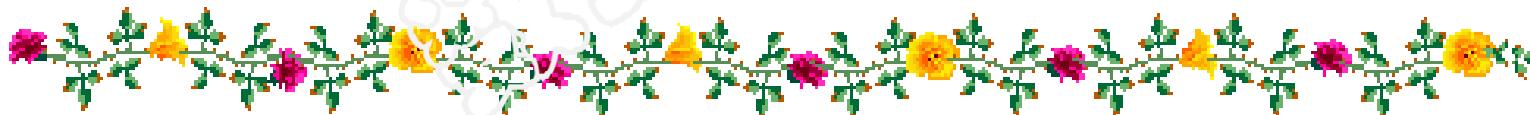
## 6.1 问题的提出

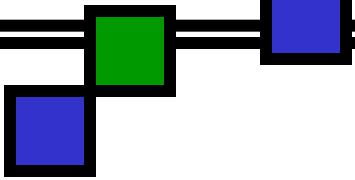
## 6.2 规范化

## 6.3 数据依赖的公理系统

## \*6.4 模式的分解

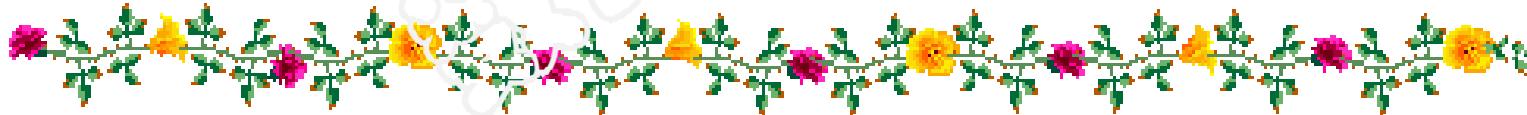
## 6.5 小结





## 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



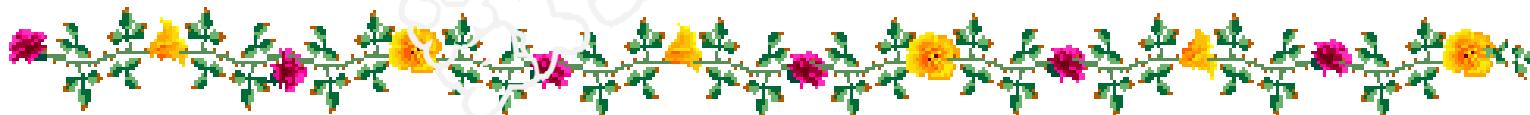
一、概念回顾

二、关系模式的形式化定义

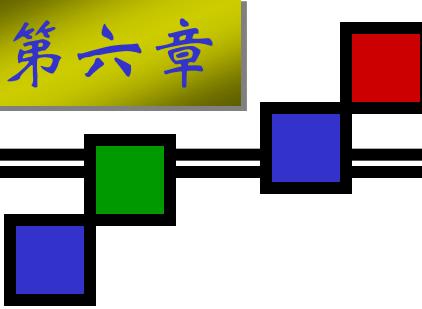
三、什么是数据依赖

四、关系模式的简化定义

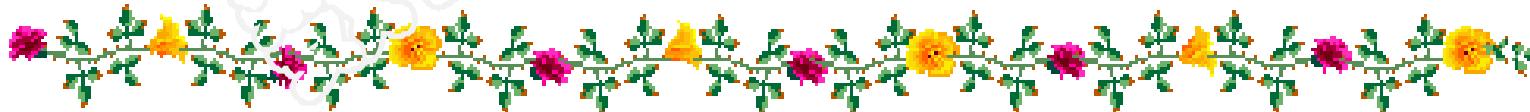
五、数据依赖对关系模式影响



## 一、概念回顾



- ❖ **关系:**若干元组的集合，说白了就是指数据库表
- ❖ **关系模式:**对关系的描述称为关系模式
- ❖ **关系数据库:**是建立在关系数据库模型基础上的数据库
- ❖ **关系数据库的模式:**三级模式结构：外模式、模式和内模式



## 二、关系模式的形式化定义

关系模式由五部分组成，即它是一个**五元组**：

**R(U, D, DOM, F)**

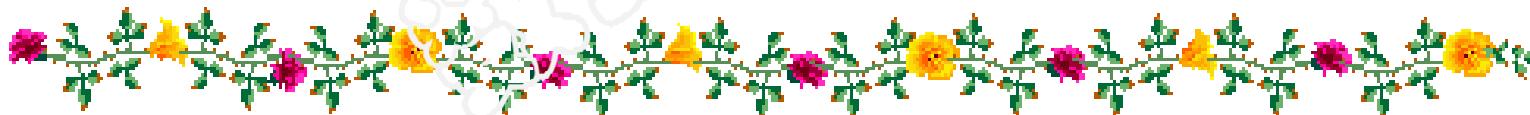
**R:** 关系名

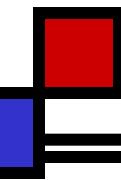
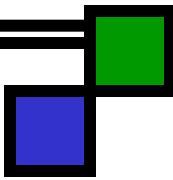
**U:** 组成该关系的属性名集合

**D:** 属性组**U**中属性所来自的域

**DOM:** 属性向域的映象集合

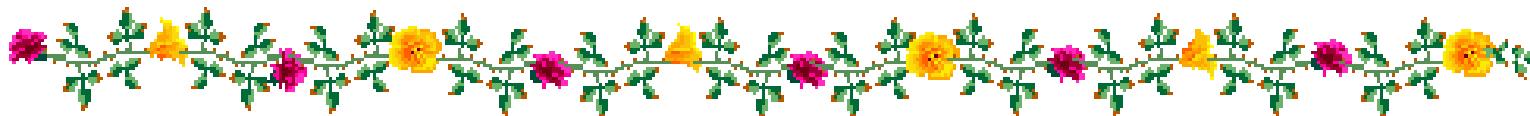
**F:** 属性间数据的依赖关系集合

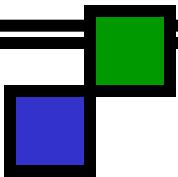




#### 1. 完整性约束的表现形式

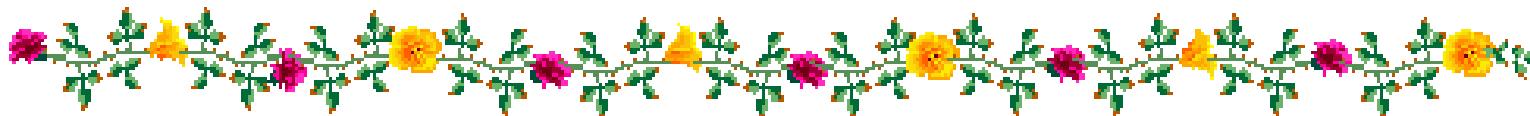
- ❖ 限定属性取值范围：例如学生成绩必须在**0-100**之间
- ❖ 定义属性**值**间的相互关连（主要体现于值的**相等与否**），这就是**数据依赖**，它是数据库模式设计的关键





## 2. 数据依赖

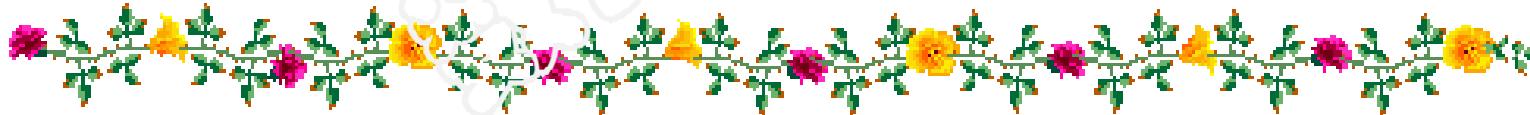
- ❖ 一个关系内部属性与属性之间的约束关系
- ❖ 现实世界属性间相互联系的抽象
- ❖ 数据内在的性质
- ❖ 语义的体现

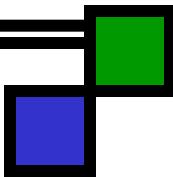




### 3. 数据依赖的类型

- ❖ **函数依赖 (Functional Dependency, 简记为FD)**
- ❖ **多值依赖 (Multivalued Dependency, 简记为MVD)**
- ❖ **其他**

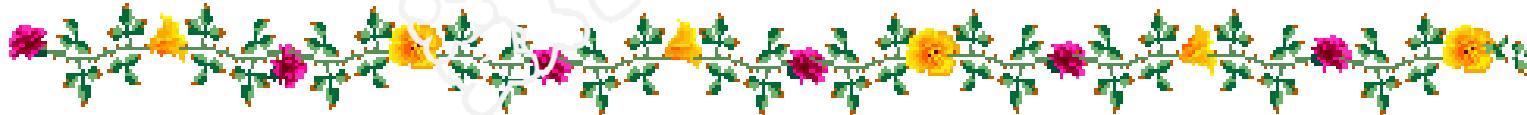


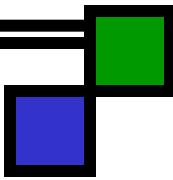


- ◆ 关系模式  $R (U, D, DOM, F)$  简化为一个三元组：

$R (U, F)$

- ◆ 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



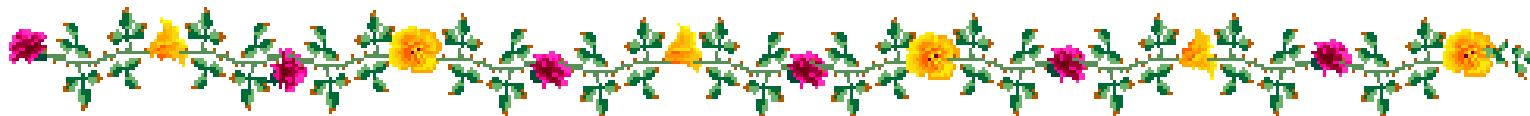


[例1]建立一个描述学校教务的数据库：学生的学号（**Sno**）、所在系（**Sdept**） 系主任姓名（**Mname**）、课程名（**Cname**） 成绩（**Grade**）

单一的关系模式

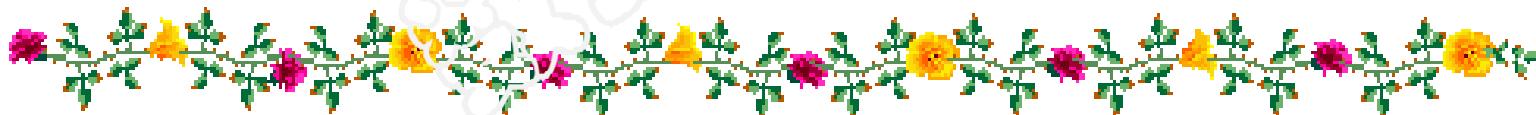
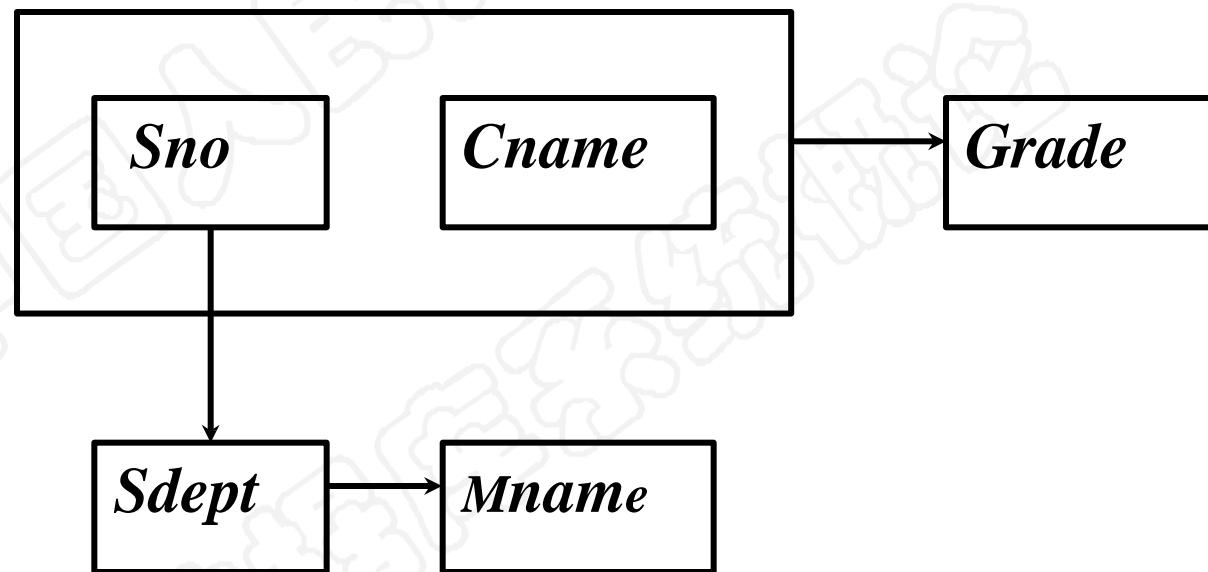
**Student <U、F>**

**U = { Sno, Sdept, Mname, Cname, Grade }**



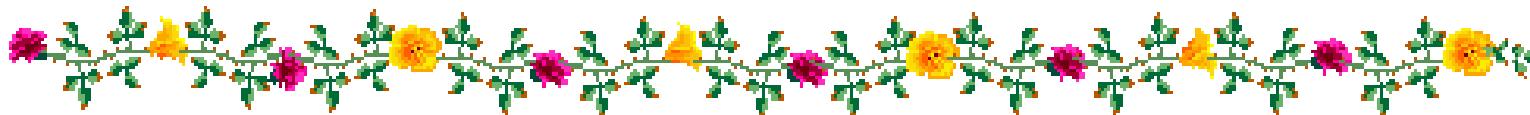
属性组U上的一组函数依赖F:

$$F = \{ Sno \rightarrow Sdept, \quad Sdept \rightarrow Mname, \\ (Sno, Cname) \rightarrow Grade \}$$



## 关系模式**Student**<U, F>中存在的问题

1. 数据冗余太大
2. 更新异常 (Update Anomalies)
3. 插入异常 (Insertion Anomalies)
4. 删除异常 (Deletion Anomalies)

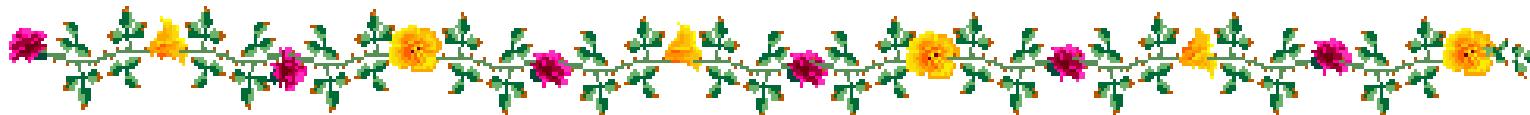


## (1) 存储冗余

- ◆ 一个学生肯定要学几十门课，那么该学生的姓名、系编号、系主任等信息就要重复存储，其存储冗余问题是相当严重的。

## (2) 插入异常

- ◆ 对于刚成立的系，如果还没有学生，由于**Snum**是**主属性**，不能为空值，因此该系主任等信息就无法加入到该关系中，这是极不合理的。即存在插入异常问题。

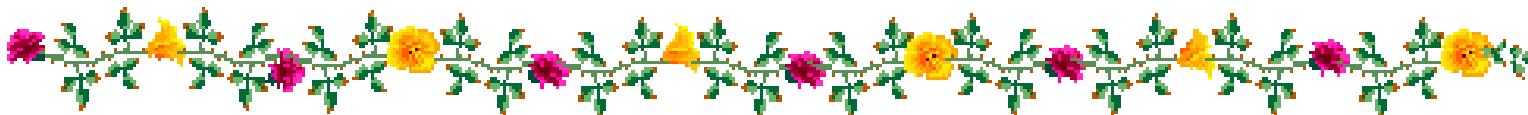


## (3) 删除异常

- ◆ 若某学生因病下学期未选课程，则需删除该学生所对应所有元组，结果该学生的学号、姓名、性别等信息也同时删去了，即删去了一些不该删除的信息。这样在该关系中就找不到该学生的姓名、性别等信息了。这也是极不合理的。

## (4) 修改异常

- ◆ 如果更换了某个系的主任，那么该系学生所有对应的元组的系主任等信息都要修改，修改量很大，潜在着严重的数据不一致问题，有可能会出现同一个系有不同主任的情况。这种不一致性是由于数据的存储冗余产生的。

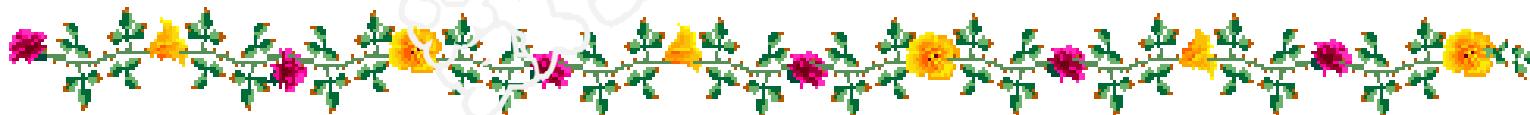


结论：

- **Student**关系模式不是一个好的模式。
- “好”的模式： 不会发生插入异常、删除异常、更新异常，  
数据冗余应尽可能少

原因：由存在于模式中的某些数据依赖引起的

解决方法：通过分解关系模式来消除其中不合适的数据依赖

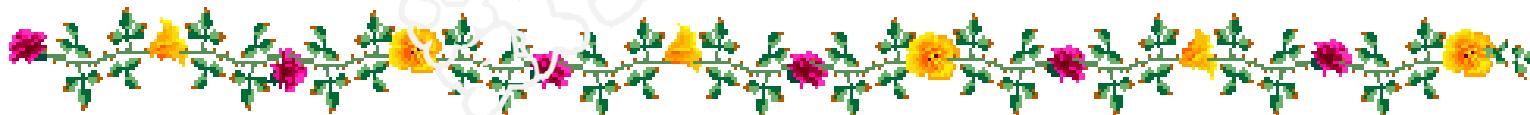


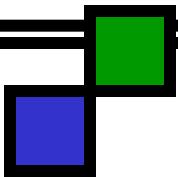
- ❖ 把这个单一模式分成3个关系模式：

**S (Sno, Sdept, Sno → Sdept) ;**

**SC (Sno, Cno, Grade, (Sno, Cno) → Grade) ;**

**DEPT (Sdept, Mname, Sdept→ Mname)**





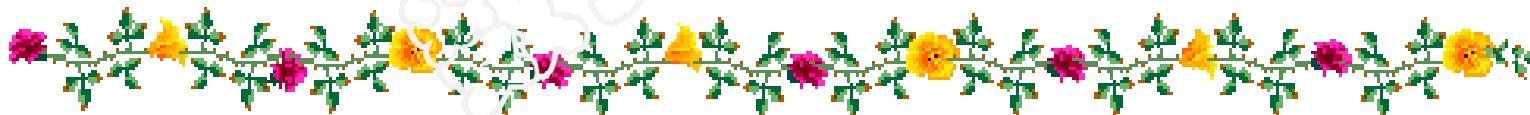
## 6.1 问题的提出

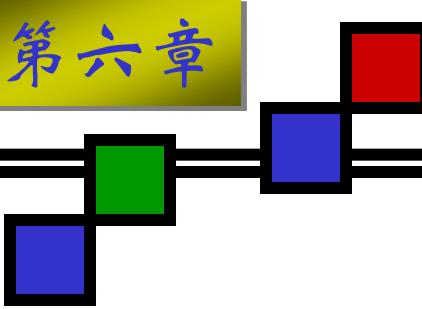
## 6.2 规范化

## 6.3 数据依赖的公理系统

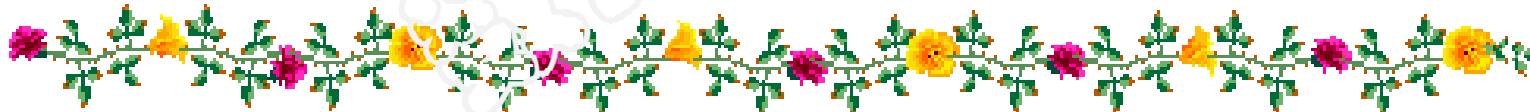
## \*6.4 模式的分解

## 6.5 小结

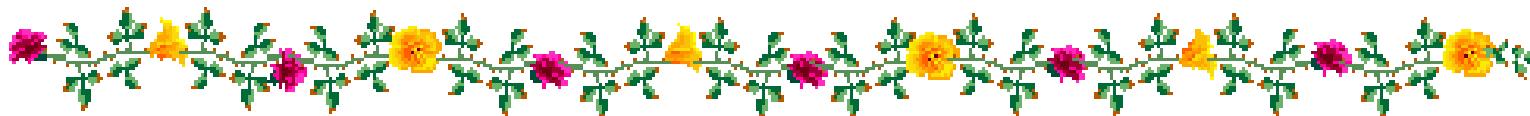




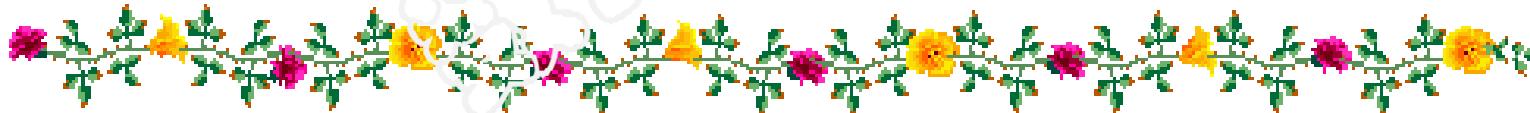
- **规范化理论**正是用来改造关系模式，通过分解关系模式来消除其中不合适的数据依赖，以解决插入异常、删除异常、更新异常和数据冗余问题。

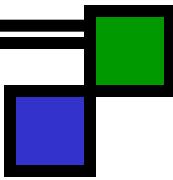


1. 函数依赖
2. 码
3. 范式
4. **2NF**
5. **3NF**
6. **BCNF**
7. 多值依赖
8. **4NF**
9. 规范化小结



- ❖ 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

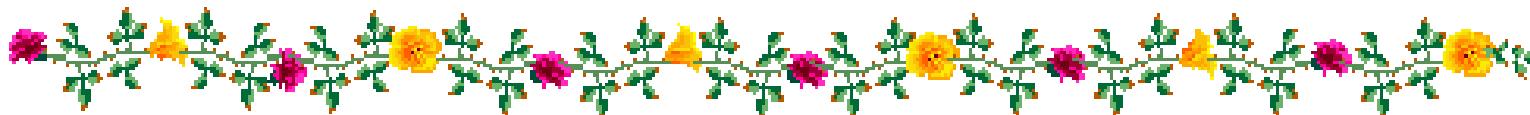


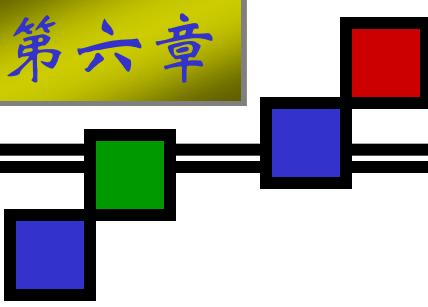


**定义6.1** 设 $R(U)$ 是一个属性集 $U$ 上的关系模式， $X$ 和 $Y$ 是 $U$ 的子集。

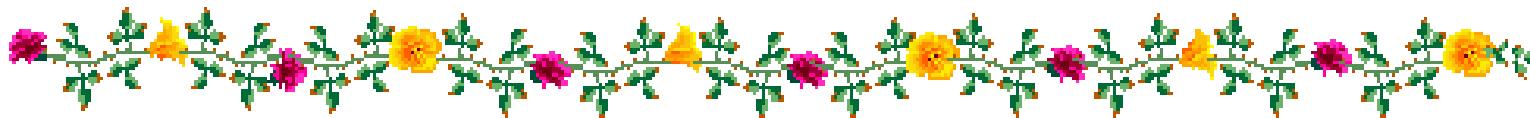
若对于 $R(U)$ 的任意一个可能的关系 $r$ ， $r$ 中不可能存在两个元组在 $X$ 上的属性值相等，而在 $Y$ 上的属性值不等，则称“ $X$  函数确定 $Y$ ”或“ $Y$ 函数依赖于 $X$ ”，记作 $X \rightarrow Y$ 。

函数依赖简单点说就是：某个属性集决定另一个属性集时，称另一属性集依赖于该属性集





1. 所有关系实例均要满足
2. 语义范畴的概念
3. 数据库设计者可以对现实世界作强制的规定





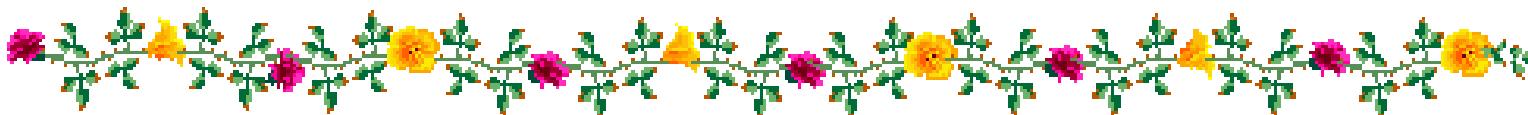
在关系模式  $R(U)$  中，对于  $U$  的子集  $X$  和  $Y$ ，如果  $X \rightarrow Y$ ，但  $Y \not\subseteq X$ ，  
则称  $X \rightarrow Y$  是非平凡的函数依赖  
若  $X \rightarrow Y$ ，但  $Y \subseteq X$ ，则称  $X \rightarrow Y$  是平凡的函数依赖

❖ 例：在关系  $SC(Sno, Cno, Grade)$  中，

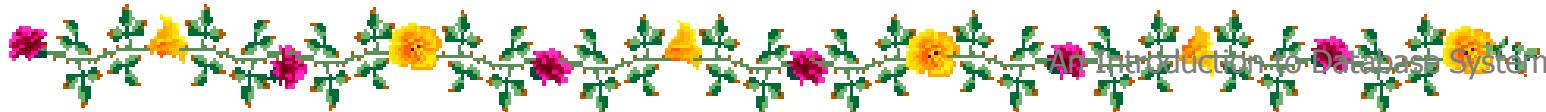
非平凡函数依赖：  $(Sno, Cno) \rightarrow Grade$

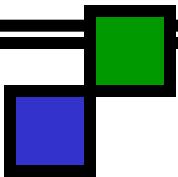
平凡函数依赖：  $(Sno, Cno) \rightarrow Sno$

$(Sno, Cno) \rightarrow Cno$



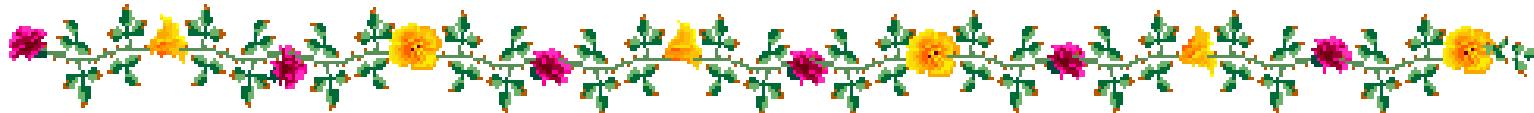
- 若  $X \rightarrow Y$ , 则  $X$  称为这个函数依赖的 **决定属性组**, 也称为 **决定因素** (**Determinant**)。
- 若  $X \rightarrow Y$ ,  $Y \rightarrow X$ , 则记作  $X \longleftrightarrow Y$ 。
- 若  $Y$  不函数依赖于  $X$ , 则记作  $X \not\rightarrow Y$ 。





**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 $X' \not\rightarrow Y$ ，则称Y对X**完全函数依赖**，记作  $X^F \rightarrow Y$ 。

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 $X^P \rightarrow Y$ 。



[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

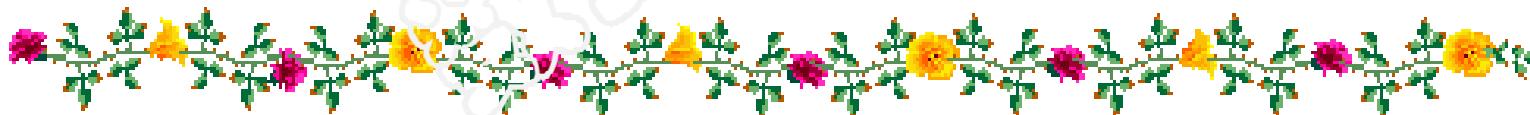


定义6.3 在R(U)中，如果 $X \rightarrow Y$ , ( $Y \not\subseteq X$ ),  $Y \nrightarrow X$   $Y \rightarrow Z$ , 则称Z对X传递函数依赖。

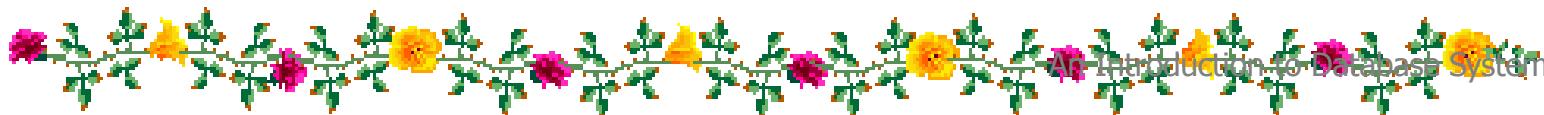
记为:  $X \xrightarrow{\text{传递}} Z$

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

例: 在关系**Std(Sno, Sdept, Mname)**中, 有: **Sno → Sdept**, **Sdept → Mname** **Mname**传递函数依赖于**Sno**



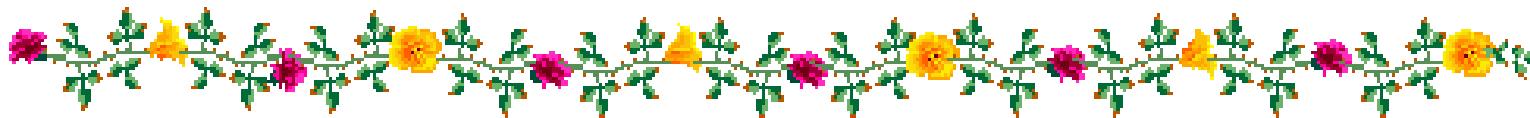
1. 函数依赖
2. 码
3. 范式
4. **2NF**
5. **3NF**
6. **BCNF**
7. 多值依赖
8. **4NF**
9. 规范化小结



## 6.2.2 码

定义6.4 设 $K$ 为 $R<U,F>$ 中的属性或属性组合。若 $K \xrightarrow{F} U$ ，则 $K$ 称为 $R$ 的**候选码**（Candidate Key）。

若候选码多于一个，则选定其中的一个做为**主码**（Primary Key）。

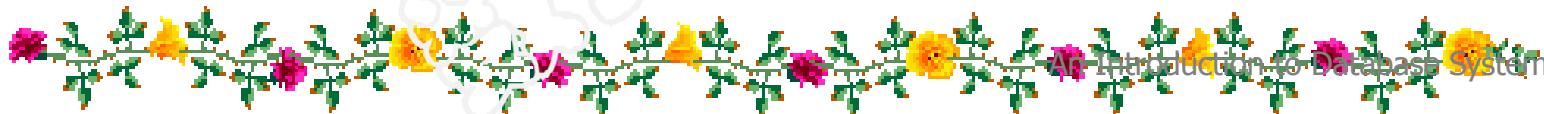


❖ 主属性与非主属性

- 包含在任何一个候选码中的属性，称为**主属性** (**Prime attribute**)
- 不包含在任何码中的属性称为**非主属性** (**Nonprime attribute**) 或**非码属性** (**Non-key attribute**)

❖ 全码

- 整个属性组是码，称为**全码** (**All-key**)

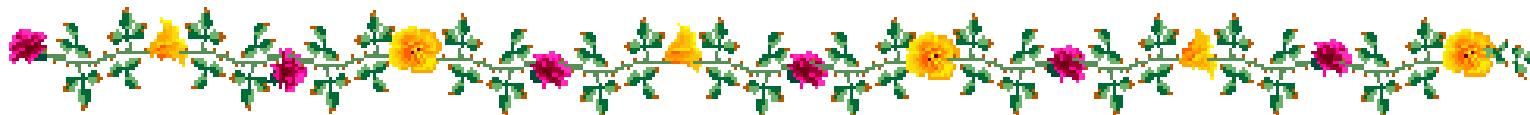


## [例2]

关系模式 **S(Sno, Sdept, Sage)**, 单个属性 **Sno** 是码,  
**SC (Sno, Cno, Grade)** 中, **(Sno, Cno)** 是码

## [例3]

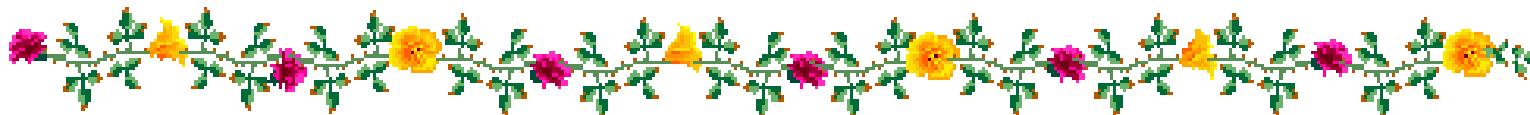
关系模式 **R (P, W, A)** P: 演奏者 W: 作品 A: 听众  
一个演奏者可以演奏多个作品, 某一作品可被多个演奏者演奏  
听众可以欣赏不同演奏者的不同作品, 码为 **(P, W, A)**, 即 All-Key

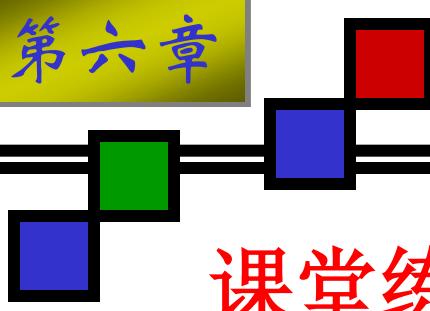


**定义6.5:**关系模式  $R$  中属性或属性组  $X$  并非  $R$  的码，但  $X$  是另一个关系模式的码，则称  $X$  是  $R$  的**外部码**

(**Foreign key**) 也称外码

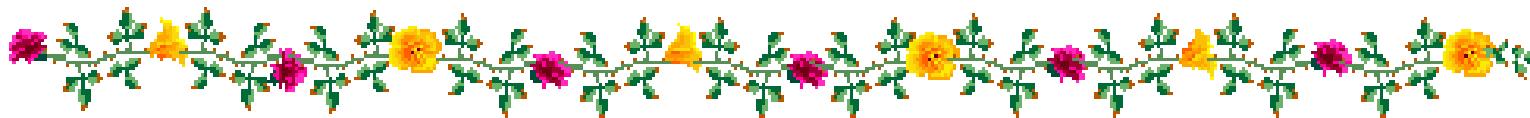
- ❖ 如在  $SC$  (**Sno**, **Cno**, **Grade**) 中，**Sno** 不是码，但 **Sno** 是关系模式  $S$  (**Sno**, **Sdept**, **Sage**) 的码，则 **Sno** 是关系模式  $SC$  的外部码
- ❖ 主码与外部码一起提供了表示关系间联系的手段





## 课堂练习

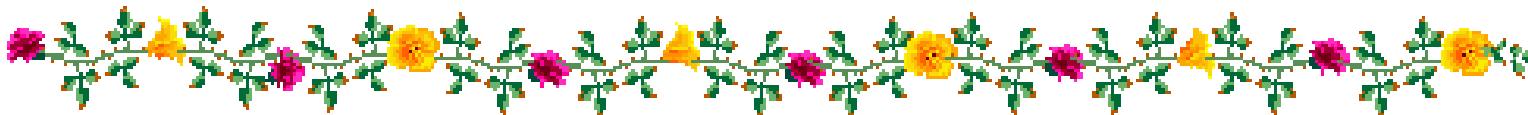
完成教材习题第二题（不需要做极小依赖集）



- 学生（学号，姓名，出生年月，系名，班号，宿舍区）
- 班级（班号，专业名，系名，人数，入校年份）
- 系（系名，洗好，系办公室地点，人数）
- 学会（学会名，成立年份，地点，人数）

### ➤ 函数依赖：

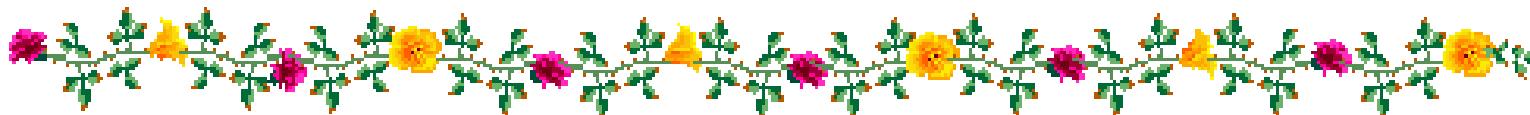
- 学号→姓名，学号→出生年月，学号→系名，学号→班号，学号→宿舍区。
- 班号→专业名，班号→系名，班号→人数，班号→入校年份。
- 系名→系号，系号→系名，系名→办公地点，系名→人数。
- 学会名→成立年份，学会名→地点，学会名→人数。
- 专业名→系名，（专业名，入校年份）→班号，
- 系名→宿舍区，（学号，学会名）→入会年份



➤ 学生关系模式的函数依赖集为：

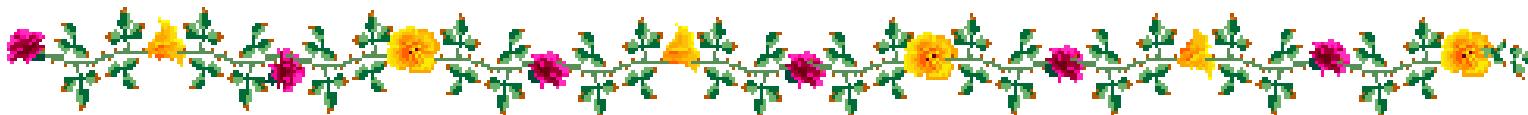
- 学号→班号， 班号→系名， 存在学号→系名的传递函数依赖。
- 学号→系名， 系名→宿舍区， 存在学号→宿舍区的传递函数依赖。
- 班号→系名， 系名→宿舍区， 存在班号→宿舍区的传递函数依赖。

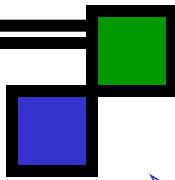
➤ 候选码：学号， 外部码：班号， 系名。



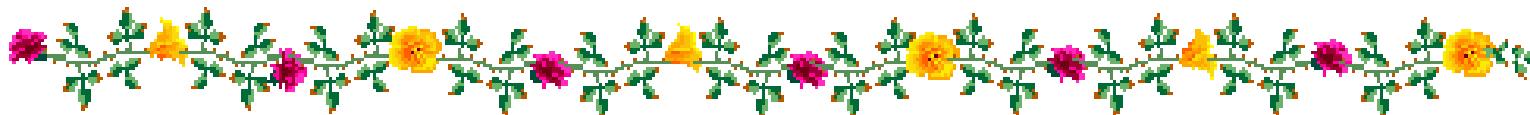
➤ 班级关系模式的函数依赖为：

- 班号→专业名，班号→系名，班号→人数，班号→入校年份，专业名→系名，（专业名，入校年份）→班号。
  - 班号→专业名，专业名→系名，存在班号→系名的传递函数依赖。
- 候选码：班号，（专业名，入校年份），外部码：系名。

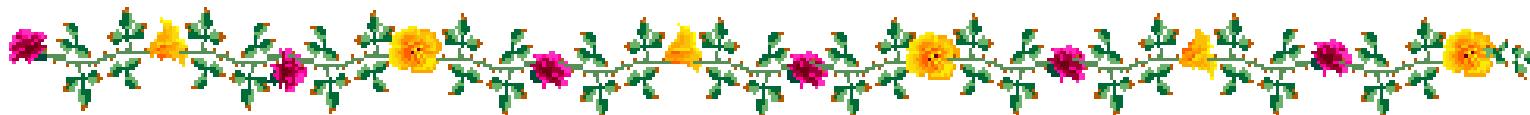




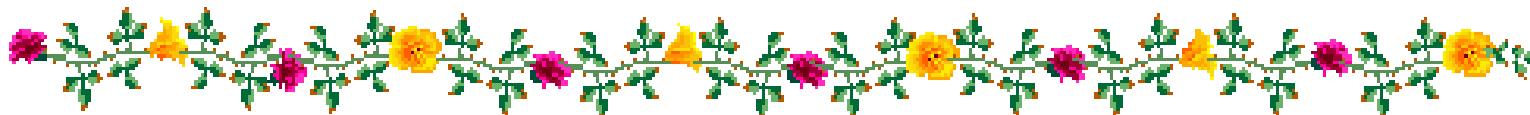
- 系关系模式的函数依赖为:
  - 系名→系号, 系号→系名, 系名→办公地点, 系名→人数。
  - 不存在传递函数依赖。
- 候选码: **系名, 系号**, 无外部码。
- 学会关系模式的函数依赖为:
  - 学会名→成立年份, 学会名→地点, 学会名→人数。
  - 不存在传递函数依赖。
- 候选码: **学会名**, 无外部码。



1. 函数依赖
2. 码
3. 范式
4. **2NF**
5. **3NF**
6. **BCNF**
7. 多值依赖
8. **4NF**
9. 规范化小结



- ❖ 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：
  - ① 第一范式(1NF)
  - ② 第二范式(2NF)
  - ③ 第三范式(3NF)
  - ④ BC范式(BCNF)
  - ⑤ 第四范式(4NF)
  - ⑥ 第五范式(5NF)



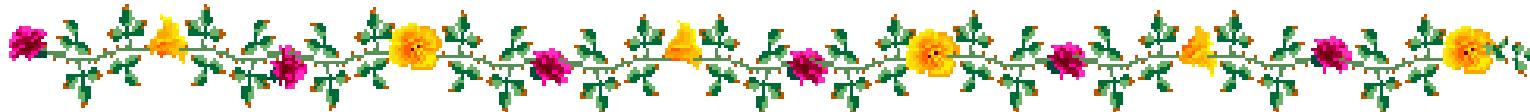
- ❖ 各种范式之间存在联系：

**1NF ⊇ 2NF ⊇ 3NF ⊇ BCNF ⊇ 4NF ⊇ 5NF**

- ❖ 某一关系模式R为第n范式，可简记为R∈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

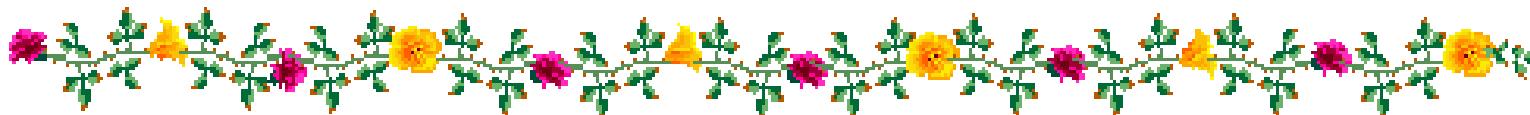
## 6.2 规范化

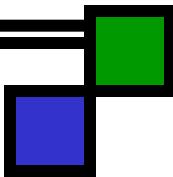
1. 函数依赖
2. 码
3. 范式
4. **2NF**
5. **3NF**
6. **BCNF**
7. 多值依赖
8. **4NF**
9. 规范化小结



## 6.2.4 2NF

- ❖ **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





[例4] 关系模式 **S-L-C(Sno, Sdept, Sloc, Cno, Grade)**

**Sloc**为学生住处，假设每个系的学生住在同一个地方

❖ 函数依赖包括：

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

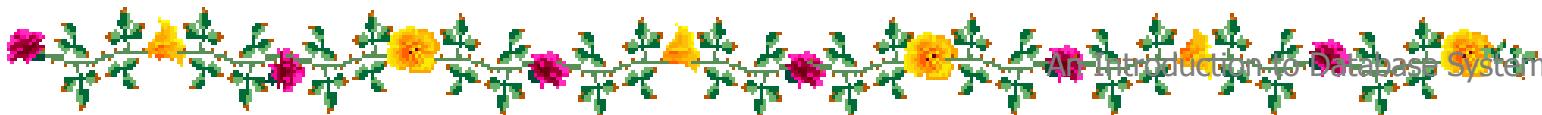
$Sno \rightarrow Sdept$

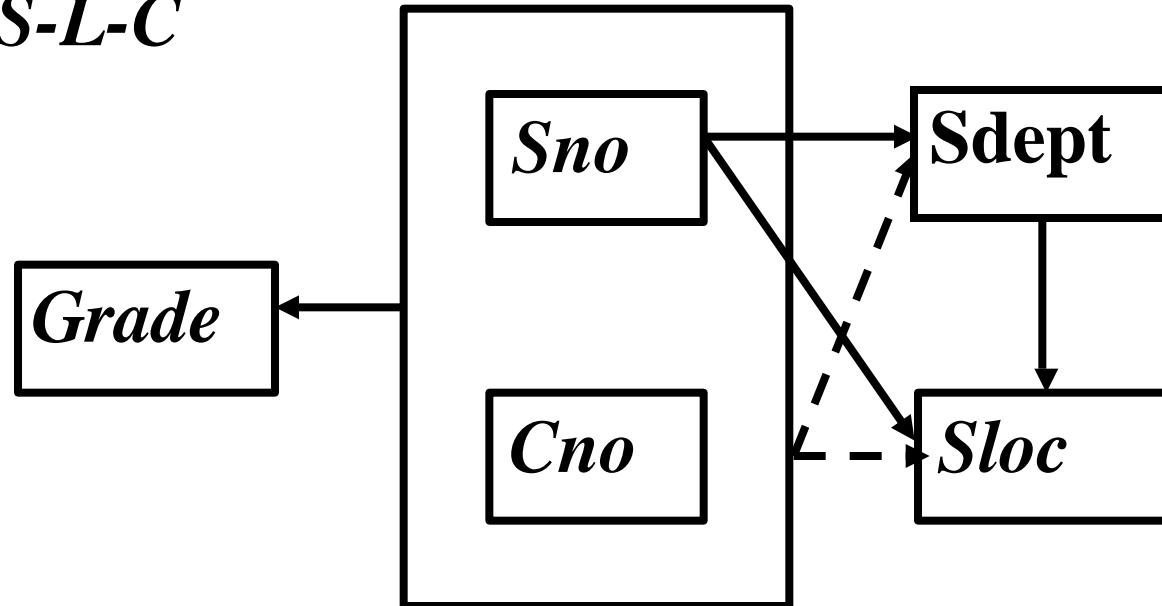
$(Sno, Cno) \xrightarrow{P} Sdept$

$Sno \rightarrow Sloc$

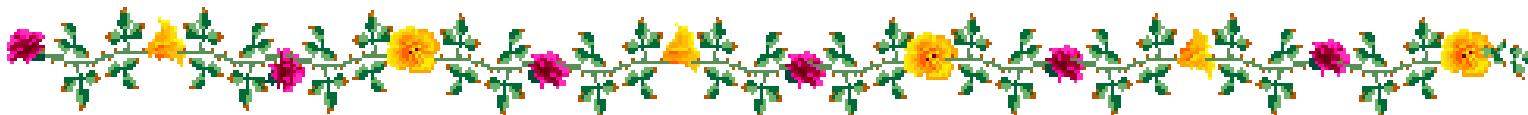
$(Sno, Cno) \xrightarrow{P} Sloc$

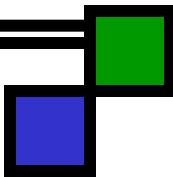
$Sdept \rightarrow Sloc$



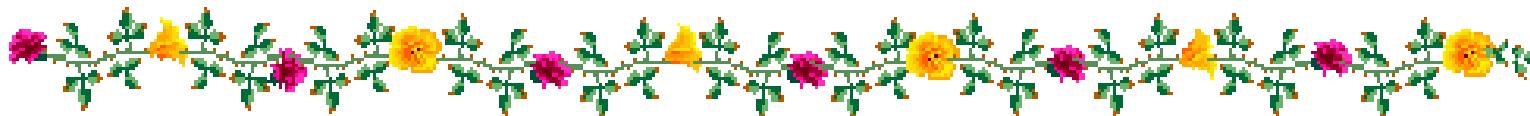
*S-L-C*

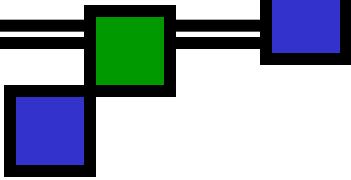
- ❖ *S-L-C*的码为(*Sno*, *Cno*)
- ❖ *S-L-C*满足第一范式。
- ❖ 非主属性*Sdept*和*Sloc*部分函数依赖于码(*Sno*, *Cno*)





- (1) 插入异常
- (2) 删除异常
- (3) 数据冗余度大
- (4) 修改复杂





❖ 原因

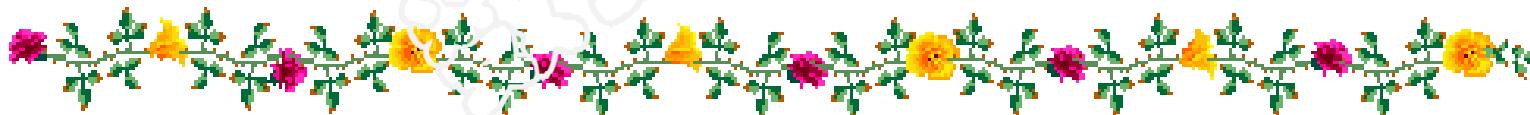
**Sdept**、**Sloc**部分函数依赖于码。

❖ 解决方法

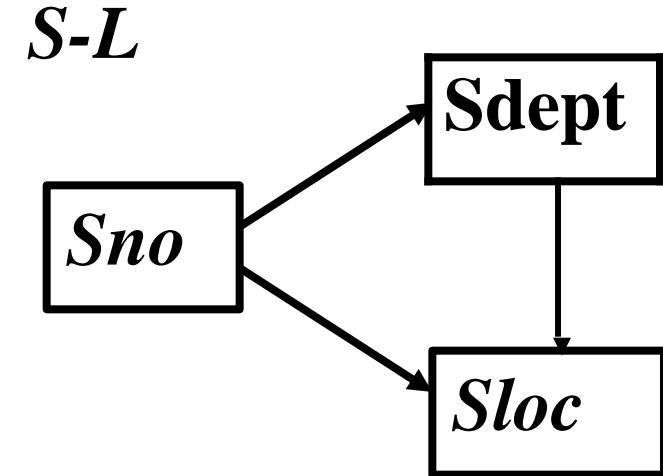
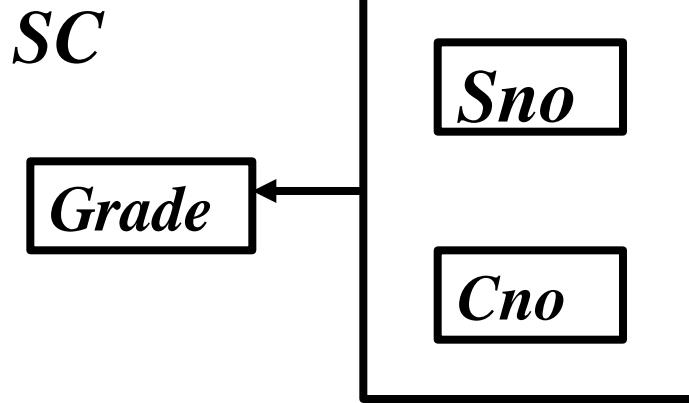
**S-L-C**分解为两个关系模式，以消除这些部分函数依赖

**SC** (**Sno**, **Cno**, **Grade**)

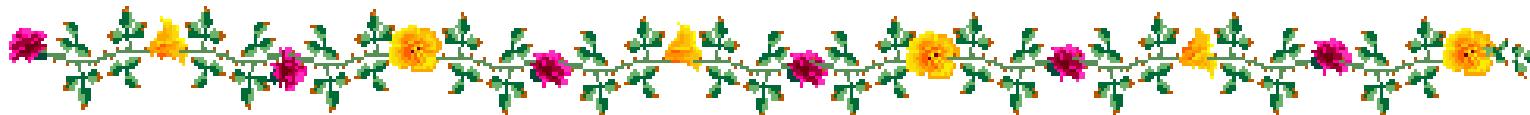
**S-L** (**Sno**, **Sdept**, **Sloc**)

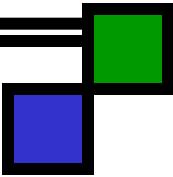


函数依赖图：



- ❖ 关系模式SC的码为 (Sno, Cno)
- ❖ 关系模式S-L的码为Sno
- ❖ 这样非主属性对码都是完全函数依赖





## ◆ 2NF的定义

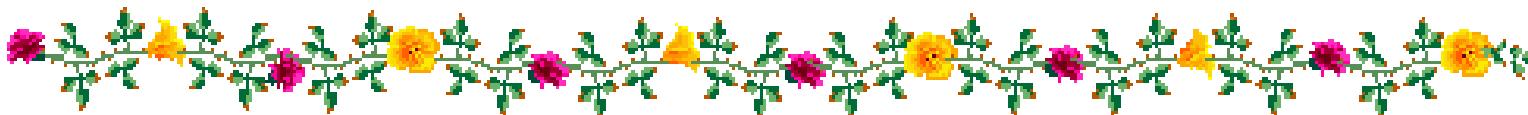
定义6.6:若 $R \in 1NF$ , 且每一个非主属性完全函数依赖于码, 则 $R \in 2NF$ 。

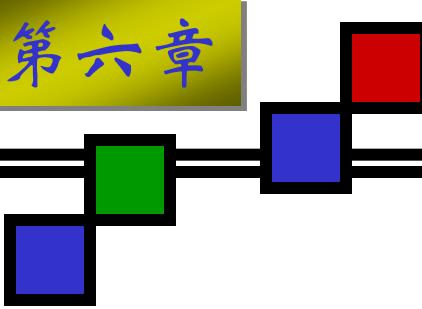
例:  $S-L-C(Sno, Sdept, Sloc, Cno, Grade) \in 1NF$

$S-L-C(Sno, Sdept, Sloc, Cno, Grade) \in 2NF$

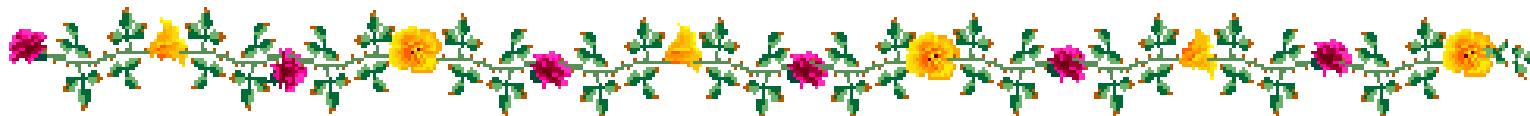
$SC(Sno, Cno, Grade) \in 2NF$

$S-L(Sno, Sdept, Sloc) \in 2NF$

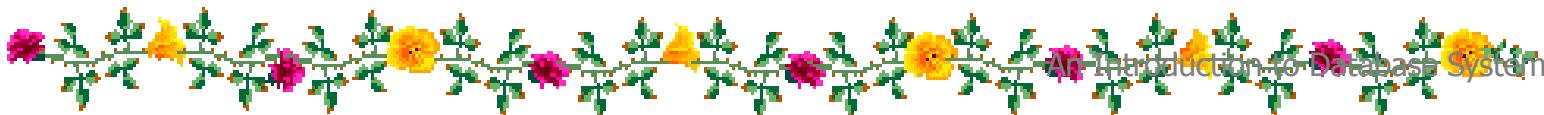


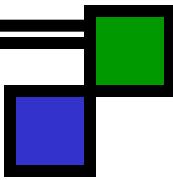


- ❖ 采用**投影分解法**将一个**1NF**的关系分解为多个**2NF**的关系，可以在一定程度上**减轻**原**1NF**关系中存在的插入异常、删除异常、数据冗余度大、修改复杂等问题。
- ❖ 将一个**1NF**关系分解为多个**2NF**的关系，**并不能完全消除**关系模式中的各种异常情况和数据冗余。



1. 函数依赖
2. 码
3. 范式
4. **2NF**
5. **3NF**
6. **BCNF**
7. 多值依赖
8. **4NF**
9. 规范化小结

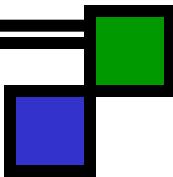




### ◆ 3NF的定义

**定义6.7:**关系模式 $R<U, F>$  中若不存在这样的码X、属性组Y及非主属性Z ( $Z \not\subseteq Y$ )，使得 $X \rightarrow Y$ ,  $Y \rightarrow Z$ 成立,  $Y \not\rightarrow X$ ，则称 $R<U, F> \in 3NF$ 。

- 若 $R \in 3NF$ ，则每一个非主属性既不部分依赖于码也不完全依赖于码。

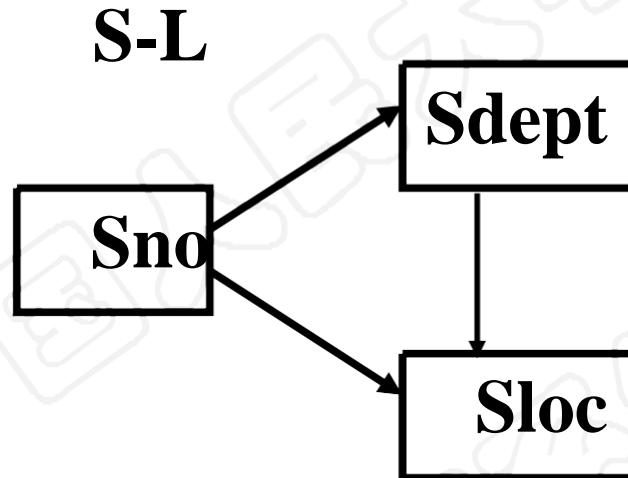


例：2NF关系模式S-L(Sno, Sdept, Sloc)中

■函数依赖： $Sno \rightarrow Sdept$   $Sdept \rightarrow Sloc$

可得： $Sno \xrightarrow{\text{传递}} Sloc$ ，即S-L中存在非主属性对码的传递函数依赖， $S-L \not\in 3NF$

函数依赖图：



### ❖ 解决方法

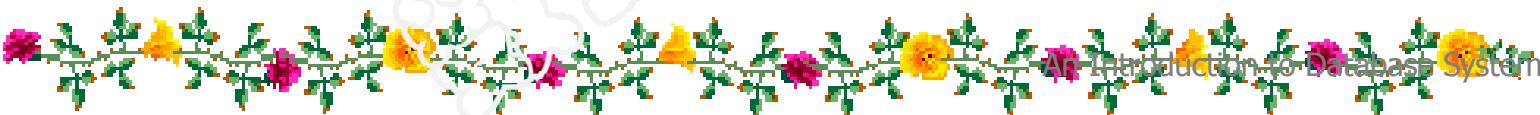
采用投影分解法，把S-L分解为两个关系模式，以消除传递函数依赖：

S-D (**Sno**, **Sdept**)

D-L (**Sdept**, **Sloc**)

S-D的码为**Sno**， D-L的码为**Sdept**。

- 分解后的关系模式**S-D**与**D-L**中不再存在传递依赖

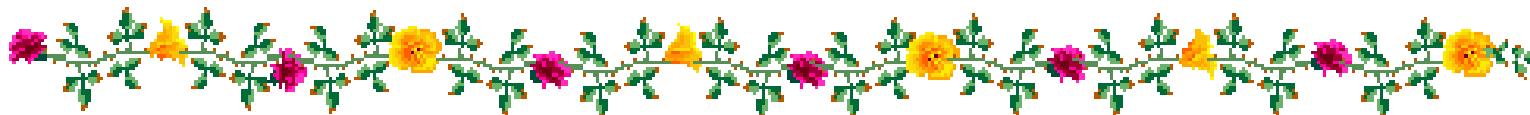




S-D的码为Sno, D-L的码为Sdept



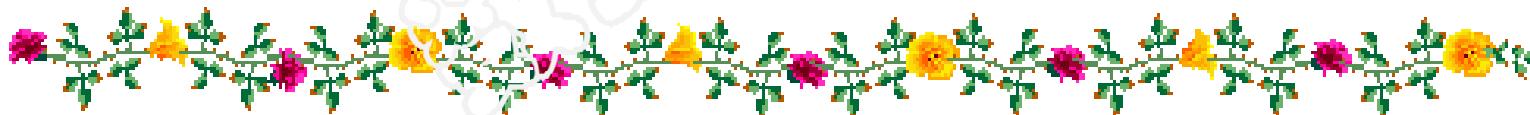
- ❖  $S-L(Sno, Sdept, Sloc) \in 2NF$
- ❖  $S-L(Sno, Sdept, Sloc) \in 3NF$
- ❖  $S-D(Sno, Sdept) \in 3NF$
- ❖  $D-L(Sdept, Sloc) \in 3NF$

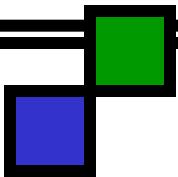


- 采用投影分解法将一个**2NF**的关系分解为多个**3NF**的关系，可以在一定程度上解决原**2NF**关系中存在的插入异常、删除异常、数据冗余度大、修改复杂等问题。
- ❖ 将一个**2NF**关系分解为多个**3NF**的关系后，仍然**不能完全消除**关系模式中的各种异常情况和数据冗余。

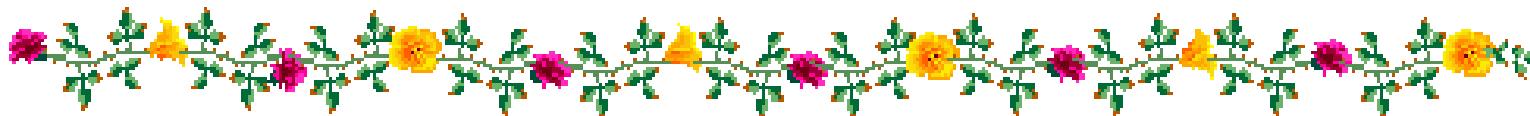
## 6.2 规范化

1. 函数依赖
2. 码
3. 范式
4. **2NF**
5. **3NF**
6. **BCNF**
7. 多值依赖
8. **4NF**
9. 规范化小结





- ❖ 定义6.8 关系模式  $R<U, F> \in 1NF$ , 若  $X \not\rightarrow Y$  且  $Y \subseteq X$  时  $X$  必含有码, 则  $R<U, F> \in BCNF$ .
- ❖ 等价于: 每一个决定属性因素都包含码

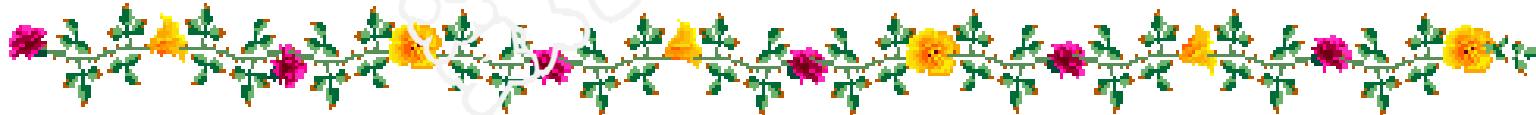


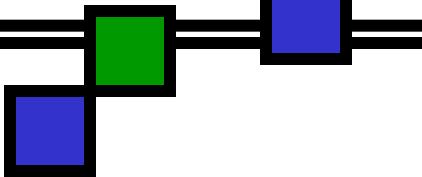


❖ 若  $R \in BCNF$

- 所有非主属性对每一个码都是完全函数依赖
- 所有的主属性对每一个不包含它的码，也是完全函数依赖
- 没有任何属性完全函数依赖于非码的任何一组属性

$$R \in BCNF \xrightleftharpoons[\text{不必要}]{\text{充分}} R \in 3NF$$



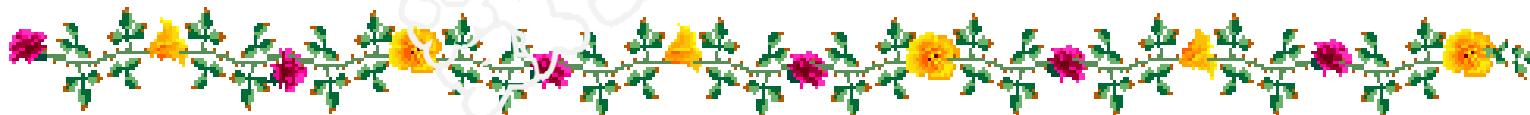


### [例5] 关系模式C (Cno, Cname, Pno)

- $C \in 3NF$
- $C \in BCNF$

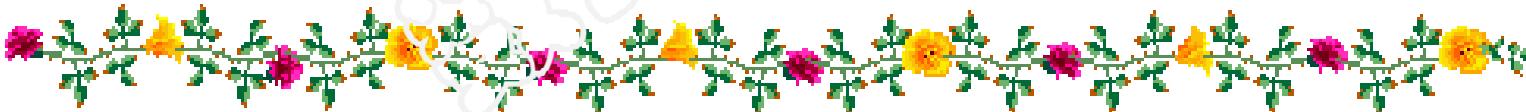
### [例6] 关系模式S (Sno, Sname, Sdept, Sage)

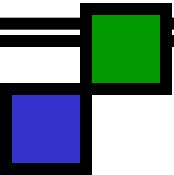
- 假定S有两个码Sno, Sname
- $S \in 3NF$ .
- $S \in BCNF$



### [例7] 关系模式SJP (S, J, P)

- 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, 属性相交
- $SJP \in 3NF$ ,
- $SJP \in BCNF$



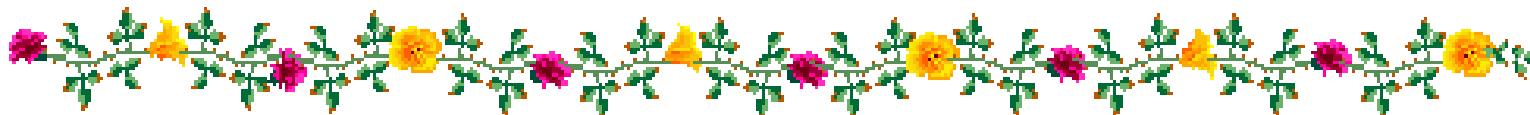


[例8]在关系模式STJ (S, T, J) 中, S表示学生, T表示教师, J表示课程。

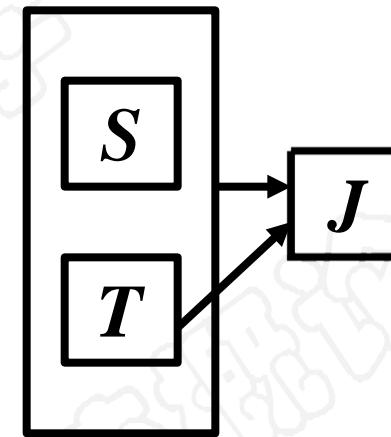
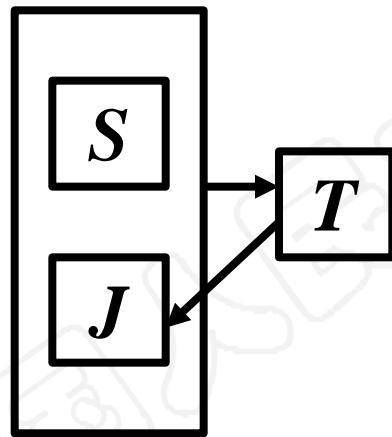
- 函数依赖:

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

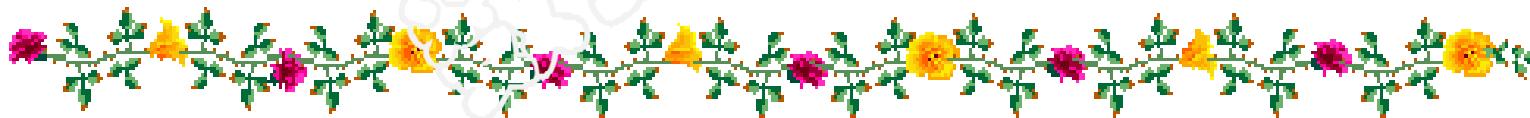
- **(S, J)和(S, T)都是候选码**



# BCNF (续)



STJ中的函数依赖

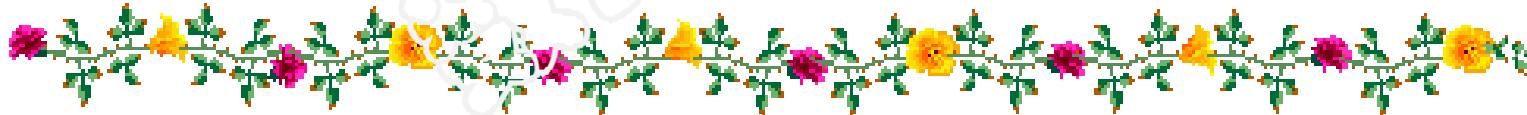


❖ **STJ ∈ 3NF**

- 没有任何非主属性对码传递依赖或部分依赖

❖ **STJ ∉ BCNF**

- T是决定因素，T不包含码



◆ 解决方法：将STJ分解为二个关系模式：

$ST(S, T) \in BCNF, TJ(T, J) \in BCNF$

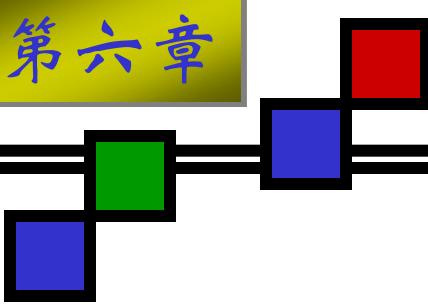


ST



TJ

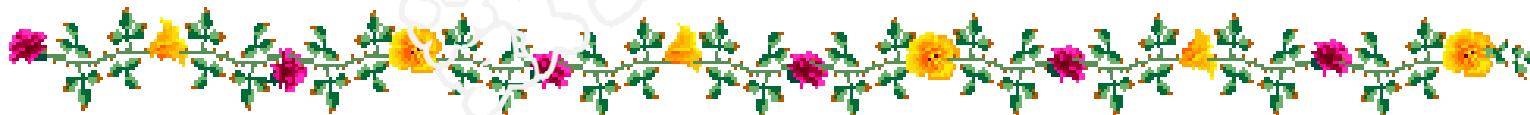
没有任何属性对码的部分函数依赖和传递函数依赖



◆  $R \in BCNF \xrightleftharpoons[\text{不必要}]{\text{充分}} R \in 3NF$

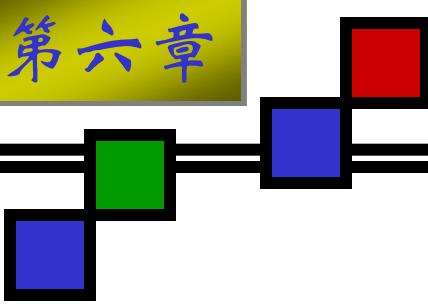
◆ 如果  $R \in 3NF$ , 且  $R$  只有一个候选码

$R \in BCNF \xrightleftharpoons[\text{必要}]{\text{充分}} R \in 3NF$

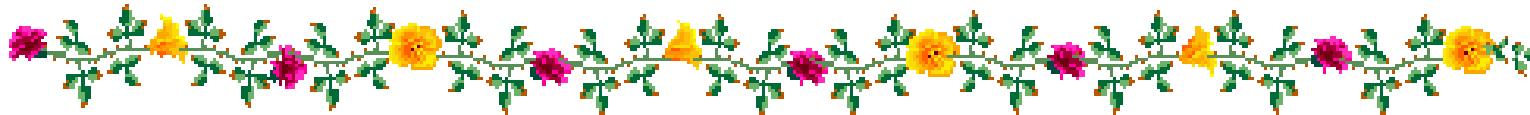


## 6.2 规范化

1. 函数依赖
2. 码
3. 范式
4. **2NF**
5. **3NF**
6. **BCNF**
7. 多值依赖
8. **4NF**
9. 规范化小结

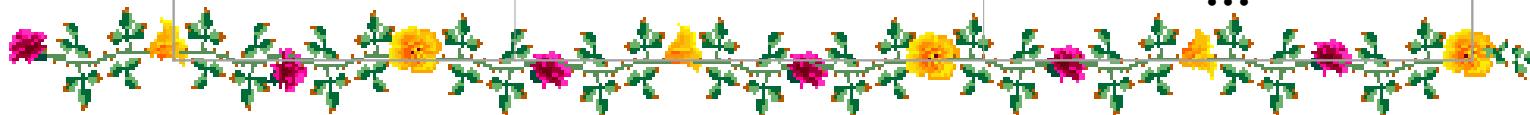


- [例9] 学校中某一门课程由多个教师讲授，他们使用相同的一套参考书。每个教员可以讲授多门课程，每种参考书可以供多门课程使用。



❖ 非规范化关系

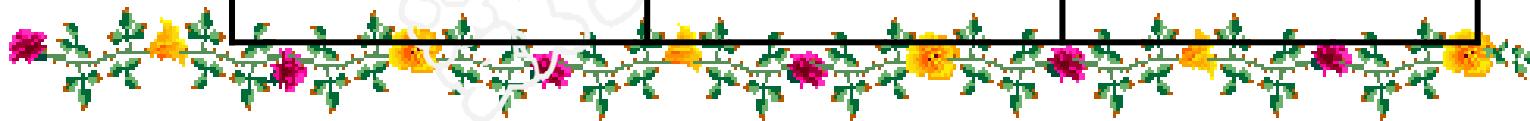
课 程 C	教 员 T	参 考 书 B
物理	李 勇 王 军	普通物理学 光学原理 物理习题集
数学	李 勇 张 平	数学分析 微分方程
计算数学	张 平 周 峰	高等代数 数学分析 ...
:	:	:

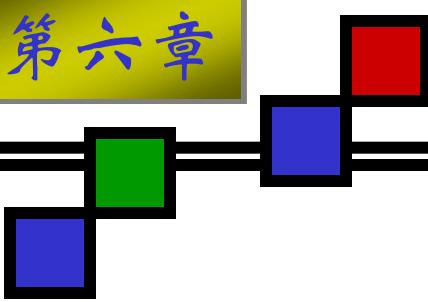


## 用二维表表示

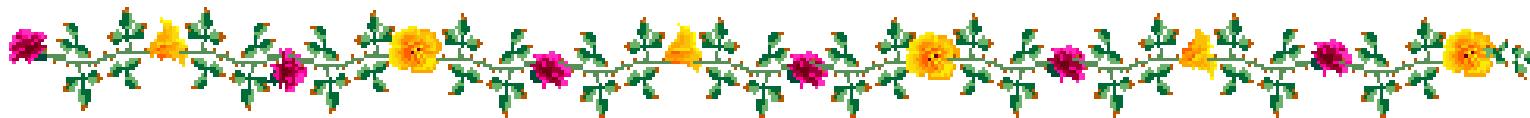
*Teaching*

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	数学分析
数学	李 勇	微分方程
数学	李 勇	高等代数
数学	张 平	数学分析
数学	张 平	微分方程
数学	张 平	高等代数
...	...	...





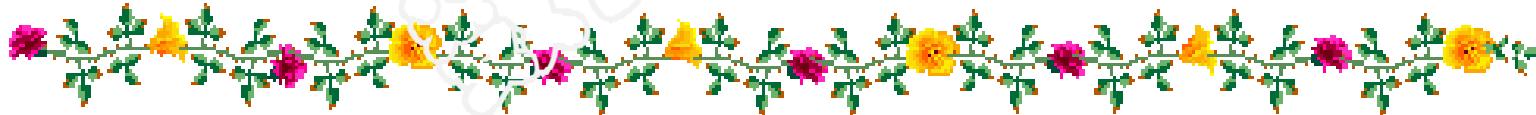
- ❖ **Teaching** ∈ BCNF
- ❖ **Teaching** 具有唯一候选码(C, T, B), 即全码



## Teaching模式中存在的问题

- (1) 数据冗余度大
- (2) 插入操作复杂
- (3) 删 除操作复杂
- (4) 修改操作复杂

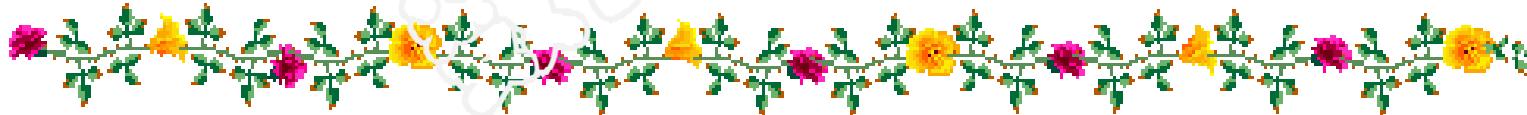
存在  
多值依赖



### ◆ 定义6.9

设  $R(U)$  是一个属性集  $U$  上的一个关系模式， $X$ 、 $Y$  和  $Z$  是  $U$  的子集，并且  $Z = U - X - Y$ 。关系模式  $R(U)$  中 多值依赖  $X \rightarrow\!\!\! \rightarrow Y$  成立，当且仅当对  $R(U)$  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。

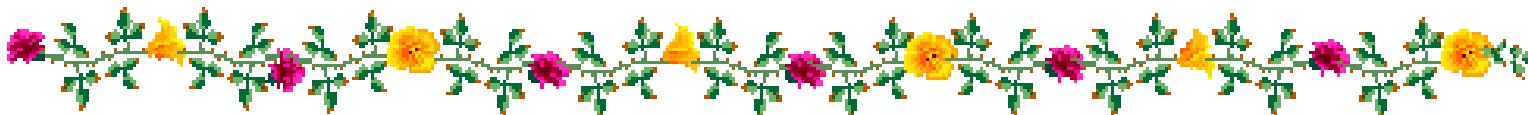
### ◆ 例 Teaching (C, T, B)

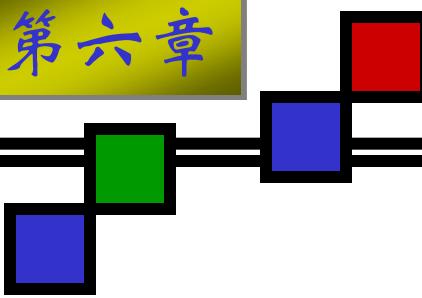


- ❖ 例 **Teaching (C, T, B)** , 对于一个 (物理, 光学原理) 有一组 T值 {李勇, 王军} , 这组值仅仅决定于课程C上的值 (物理) 。也就是说对于另一个 (物理, 普通物理学) , 它对应的一组T值仍是 {李勇, 王军} , 尽管这时参考书B的值已经变了
- ❖ 因此T多值依赖于C, 即  $C \rightarrow\!\!\!-\rightarrow T$

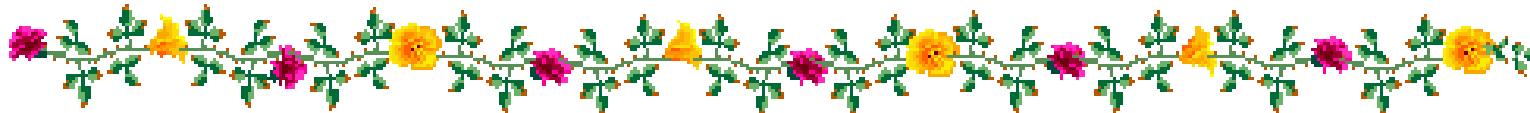
### 多值依赖的特点:

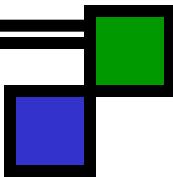
- (1) 允许X的一个值决定Y的一组值, 这种决定关系与Z取值无关。
- (2) 多值依赖是全模式的依赖关系





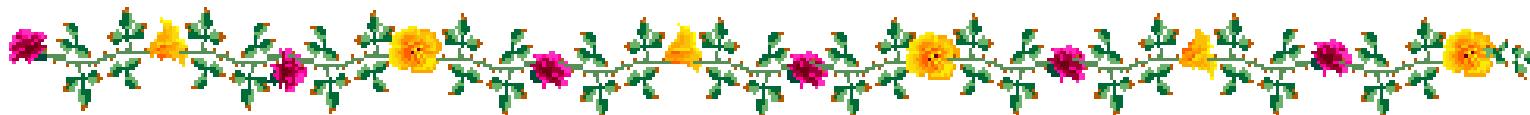
➤ 多值依赖的另一个等价的形式化的定义：在  $R$  ( $U$ ) 的任一关系  $r$  中，如果存在元组  $t, s$  使得  $t[X]=s[X]$ ，那么就必然存在元组  $w, v \in r$ ，( $w, v$  可以与  $s, t$  相同)，使得  $w[X]=v[X]=t[X]$ ，而  $w[Y]=t[Y]$ ， $w[Z]=s[Z]$ ， $v[Y]=s[Y]$ ， $v[Z]=t[Z]$  (即交换  $s, t$  元组的  $Y$  值所得的两个新元组必在  $r$  中)，则  **$Y$  多值依赖于  $X$** ，记为  **$X \rightarrow\!\! \rightarrow Y$** 。这里， $X, Y$  是  $U$  的子集， $Z=U-X-Y$ 。

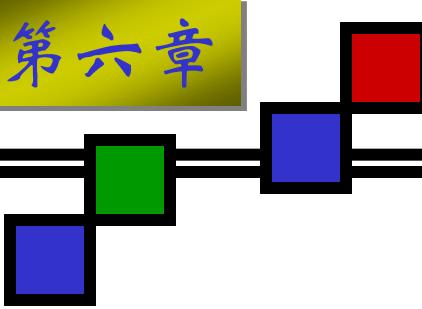




❖ 平凡多值依赖和非平凡的多值依赖

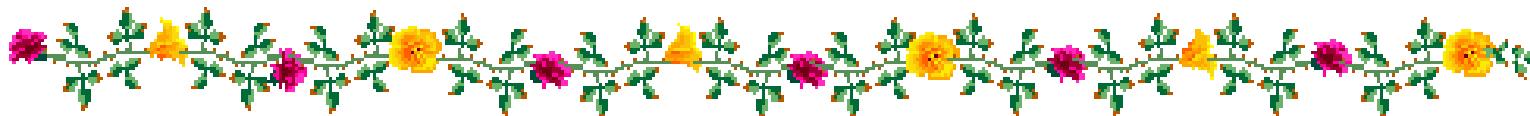
- 若  $X \rightarrow\rightarrow Y$ , 而  $Z = \varphi$ , 即  $Z$  为空, 则称  $X \rightarrow\rightarrow Y$  为 **平凡的多值依赖**
- 否则称  $X \rightarrow\rightarrow Y$  为 **非平凡的多值依赖**



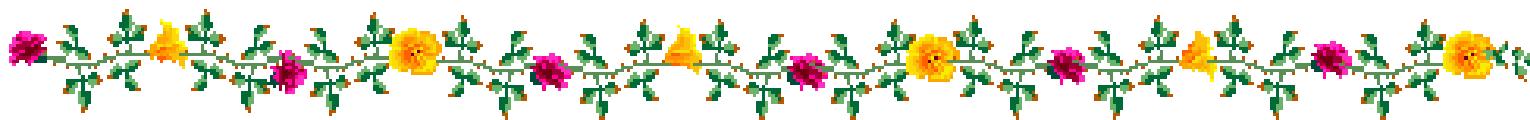


## [例10] 关系模式WSC (W, S, C)

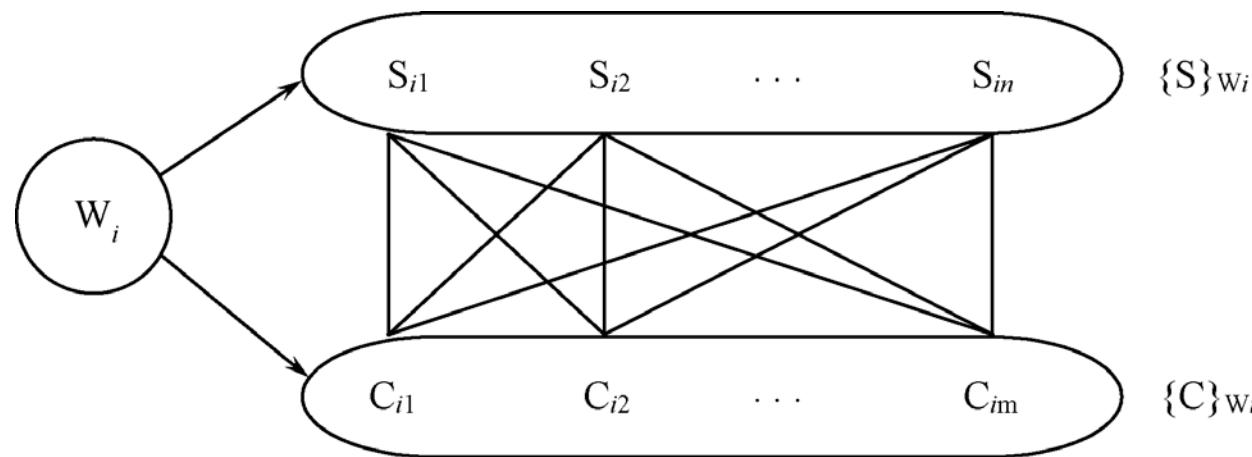
- W表示仓库, S表示保管员, C表示商品
- 假设每个仓库有若干个保管员, 有若干种商品
- 每个保管员保管所在的仓库的所有商品
- 每种商品被所有保管员保管



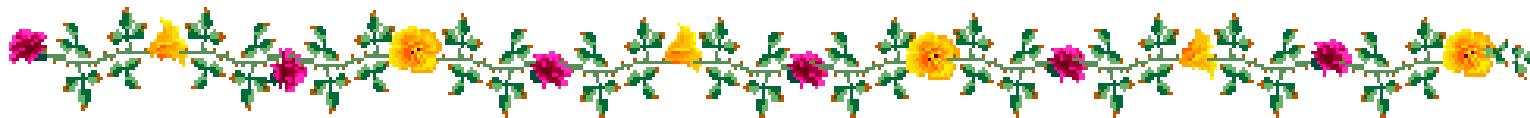
W	S	C
W1	S1	C1
W1	S1	C2
W1	S1	C3
W1	S2	C1
W1	S2	C2
W1	S2	C3
W2	S3	C4
W2	S3	C5
W2	S4	C4
W2	S4	C5

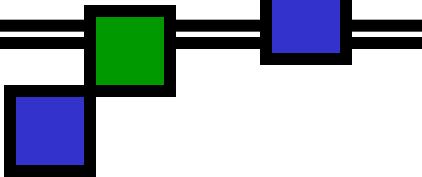


用下图表示这种对应

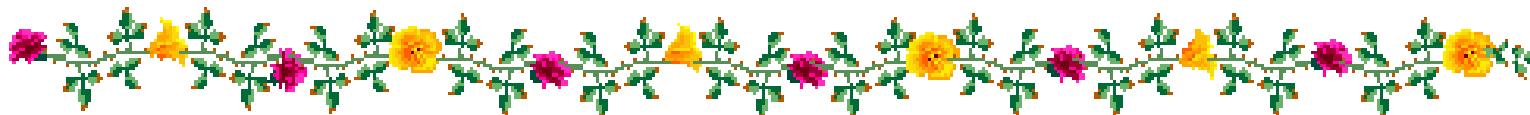


$W \rightarrow S$  且  $W \rightarrow C$





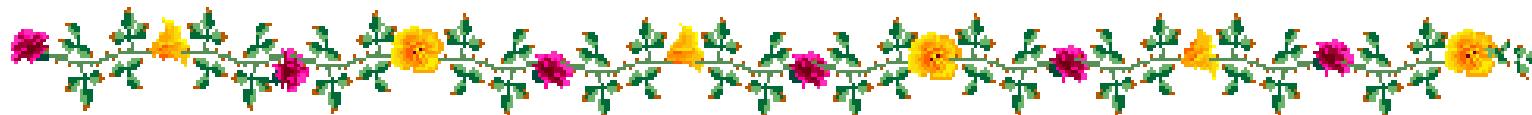
- (1) 多值依赖具有对称性 若 $X \rightarrow\rightarrow Y$ , 则 $X \rightarrow\rightarrow Z$ , 其中 $Z = U - X - Y$
- (2) 多值依赖具有传递性 若 $X \rightarrow\rightarrow Y$ ,  $Y \rightarrow\rightarrow Z$ , 则 $X \rightarrow\rightarrow Z - Y$
- (3) 函数依赖是多值依赖的特殊情况。若 $X \rightarrow Y$ , 则 $X \rightarrow\rightarrow Y$ 。
- (4) 若 $X \rightarrow\rightarrow Y$ ,  $X \rightarrow\rightarrow Z$ , 则 $X \rightarrow\rightarrow Y \cup Z$ 。
- (5) 若 $X \rightarrow\rightarrow Y$ ,  $X \rightarrow\rightarrow Z$ , 则 $X \rightarrow\rightarrow Y \cap Z$ 。
- (6) 若 $X \rightarrow\rightarrow Y$ ,  $X \rightarrow\rightarrow Z$ , 则 $X \rightarrow\rightarrow Y - Z$ ,  $X \rightarrow\rightarrow Z - Y$ 。



(1) 多值依赖的有效性与属性集的范围有关

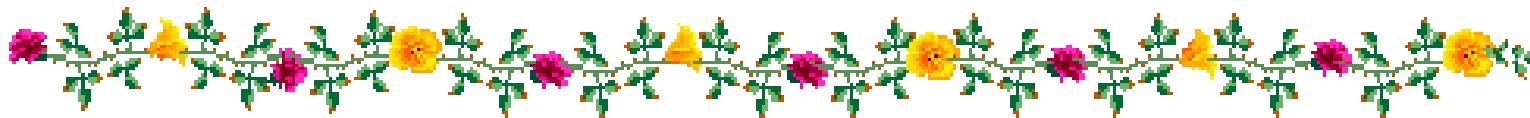
(2)

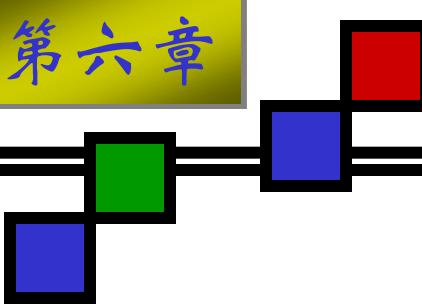
- 若函数依赖 $X \rightarrow Y$ 在 $R(U)$ 上成立，则对于任何 $Y' \subset Y$ 均有 $X \rightarrow Y'$ 成立
- 多值依赖 $X \rightarrow\rightarrow Y$ 若在 $R(U)$ 上成立，不能断言对于任何 $Y' \subset Y$ 有 $X \rightarrow\rightarrow Y'$ 成立



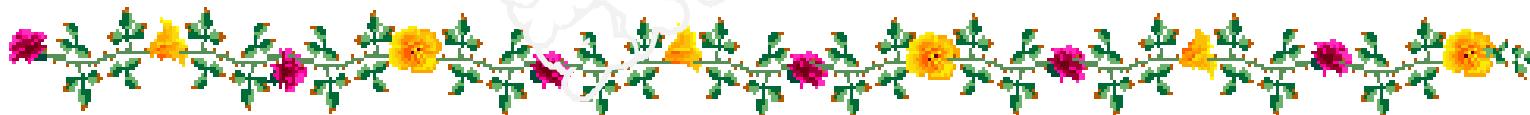
## 6.2 规范化

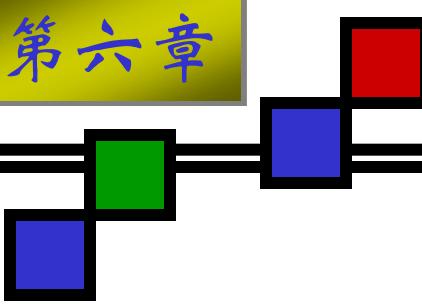
1. 函数依赖
2. 码
3. 范式
4. **2NF**
5. **3NF**
6. **BCNF**
7. 多值依赖
8. **4NF**
9. 规范化小结





- ❖ 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  $X \rightarrow\!\!\rightarrow Y$  ( $Y \not\subseteq X$ ) ,  $X$ 都含有码, 则  $R \in 4NF$ 。
- ❖ 如果  $R \in 4NF$ , 则  $R \in BCNF$ 
  - 不允许有非平凡且非函数依赖的多值依赖
  - 允许的非平凡多值依赖是函数依赖





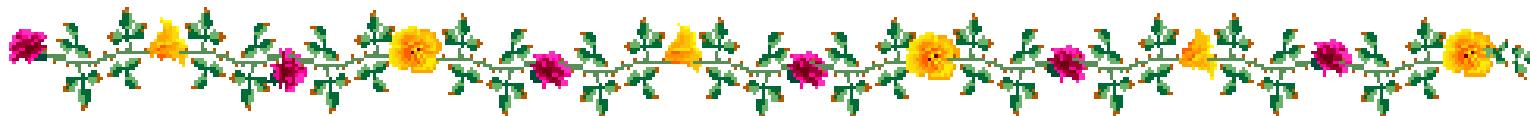
例: **Teaching(C,T,B)**  $\not\in$  4NF

存在非平凡的多值依赖  $C \rightarrow\!\!\rightarrow T$ , 且 C 不是码

■ 用投影分解法把**Teaching**分解为如下两个关系模式:

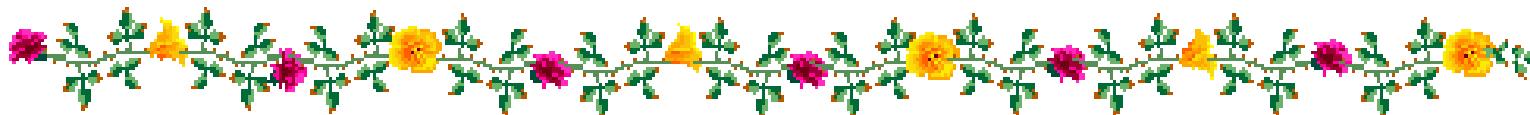
**CT(C, T) ∈ 4NF CB(C, B) ∈ 4NF**

$C \rightarrow\!\!\rightarrow T$ ,  $C \rightarrow\!\!\rightarrow B$  是平凡多值依赖

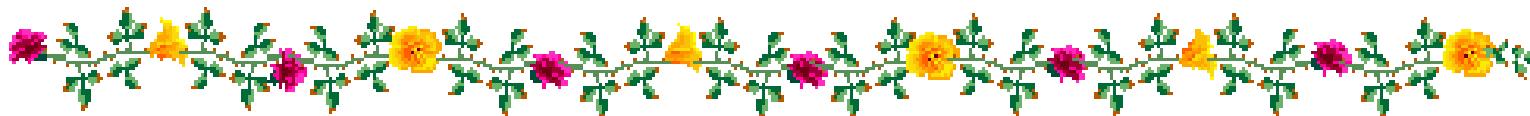


## 6.2 规范化

1. 函数依赖
2. 码
3. 范式
4. **2NF**
5. **3NF**
6. **BCNF**
7. 多值依赖
8. **4NF**
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，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



❖ 关系模式规范化的基本步骤

消除决定属性  
集非码的非平  
凡函数依赖

**1NF**

↓ 消除非主属性对码的部分函数依赖

**2NF**

↓ 消除非主属性对码的传递函数依赖

**3NF**

↓ 消除主属性对码的部分和传递函数依赖

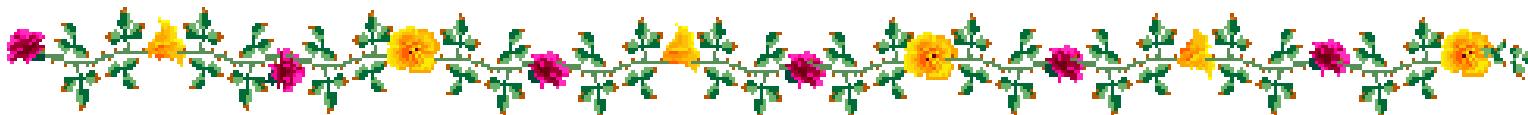
**BCNF**

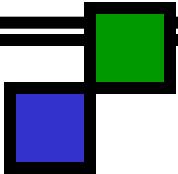
↓ 消除非平凡且非函数依赖的多值依赖

**4NF**



- ❖ 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





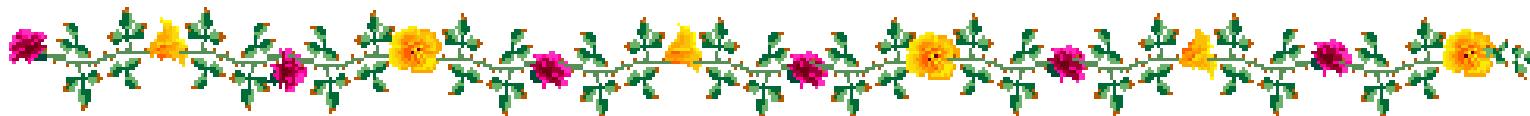
## 6.1 问题的提出

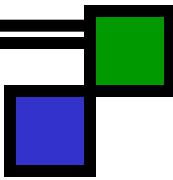
## 6.2 规范化

## 6.3 数据依赖的公理系统

## \*6.4 模式的分解

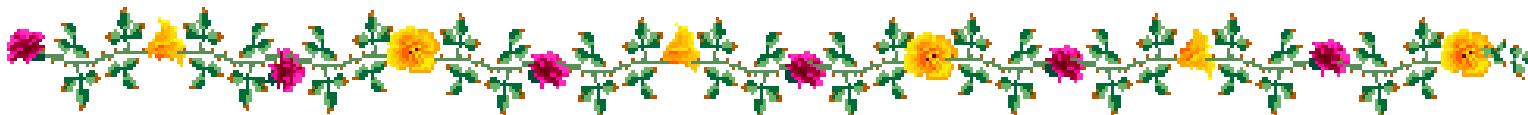
## 6.5 小结

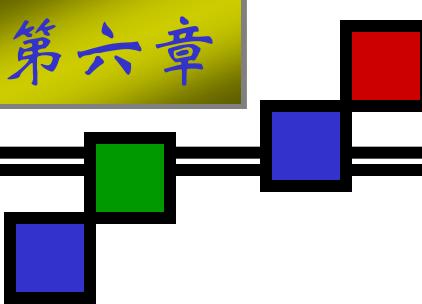




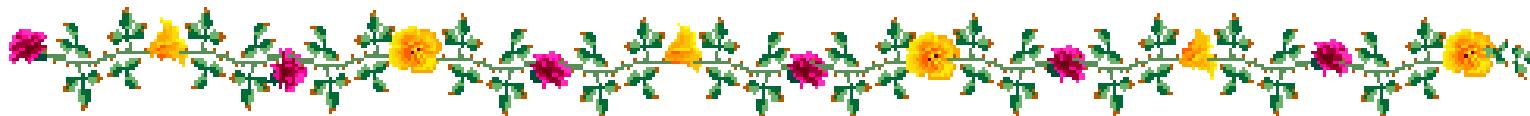
### ◆逻辑蕴含

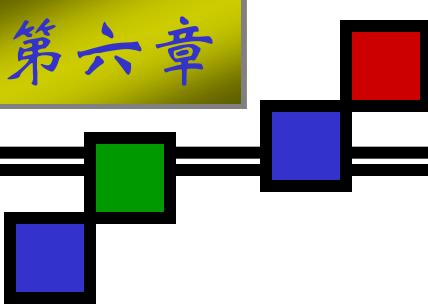
定义6.11 对于满足一组函数依赖 $F$ 的关系模式 $R <U, F>$ , 其任何一个关系 $r$ , 若函数依赖 $X \rightarrow Y$ 都成立, (即 $r$ 中任意两元组 $t, s$ , 若 $t[X] = s[X]$  , 则 $t[Y] = s[Y]$  ) , 则称 $F$ 逻辑蕴含 $X \rightarrow Y$



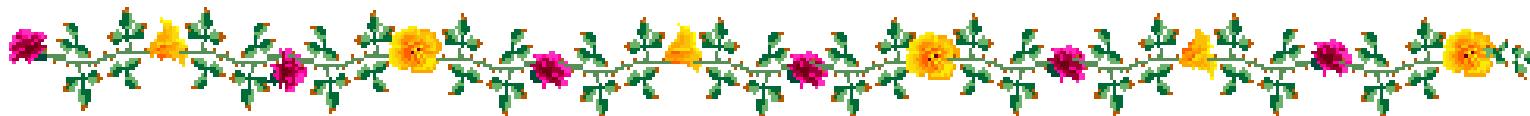


- ◆ Armstrong公理系统—函数依赖的一个有效而完备的公理系统
- ◆ Armstrong公理系统设U为属性集总体， F是U上的一组函数依赖， 于是有关系模式R<U, F>, 对于R<U, F>来说有下列推理规则





- A1. **自反律 (Reflexivity)** : 若  $Y \subseteq X \subseteq U$ , 则  $X \rightarrow Y$  为 F 所蕴含。
- A2. **增广律 (Augmentation)** : 若  $X \rightarrow Y$  为 F 所蕴含, 且  $Z \subseteq U$ , 则  $XZ \rightarrow YZ$  为 F 所蕴含。
- 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。





## 定理 6.1 Armstrong推理规则是正确的

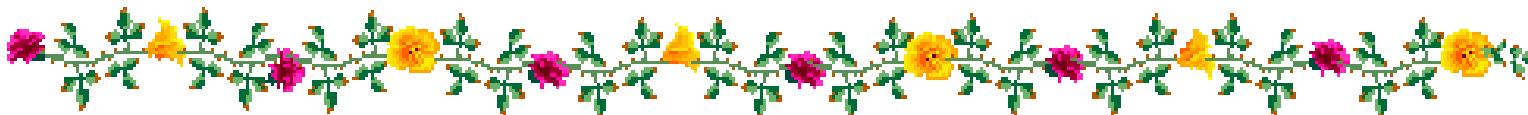
(I) 自反律: 若  $Y \subseteq X \subseteq U$ , 则  $X \rightarrow Y$  为  $F$  所蕴含

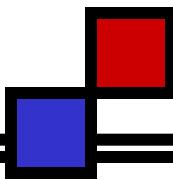
证: 设  $Y \subseteq X \subseteq U$

对  $R <U, F>$  的任一关系  $r$  中的任意两个元组  $t, s$ :

若  $t[X] = s[X]$ , 由于  $Y \subseteq X$ , 有  $t[y] = s[y]$ , 所以  $X \rightarrow Y$  成立,

自反律得证



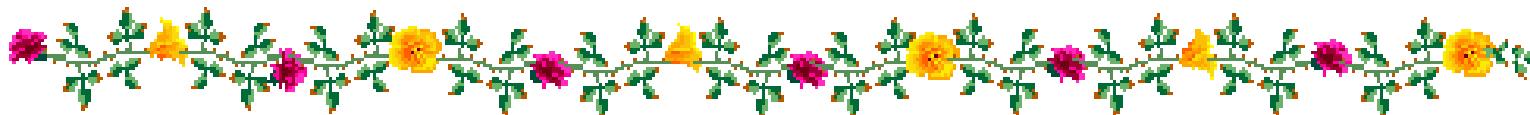


## 定理 6.1 Armstrong推理规则是正确的（续）

(2) 增广律: 若  $X \rightarrow Y$  为  $F$  所蕴含, 且  $Z \subseteq U$ , 则  $XZ \rightarrow YZ$  为  $F$  所蕴含。

证: 设  $X \rightarrow Y$  为  $F$  所蕴含, 且  $Z \subseteq U$ 。

设  $R < U, F >$  的任一关系  $r$  中任意的两个元组  $t, s$ : 若  $t[XZ] = s[XZ]$ , 则有  $t[X] = s[X]$  和  $t[Z] = s[Z]$ ; 由  $X \rightarrow Y$ , 于是有  $t[Y] = s[Y]$ , 所以  $t[YZ] = s[YZ]$ , 所以  $XZ \rightarrow YZ$  为  $F$  所蕴含, 增广律得证。





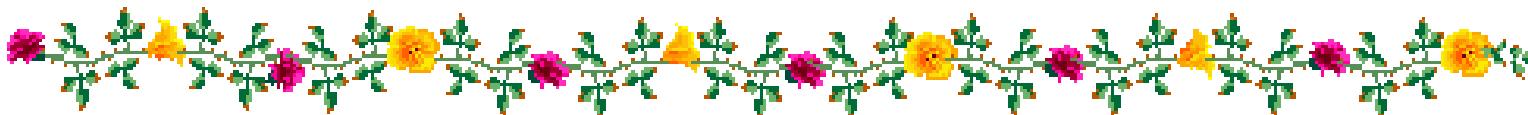
## 定理 6.1 Armstrong推理规则是正确的（续）

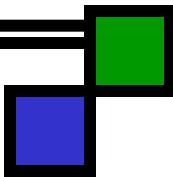
(3) 传递律：若  $X \rightarrow Y$  及  $Y \rightarrow Z$  为  $F$  所蕴含，则  $X \rightarrow Z$  为  $F$  所蕴含。

证：设  $X \rightarrow Y$  及  $Y \rightarrow Z$  为  $F$  所蕴含。

对  $R < U, F >$  的任一关系  $r$  中的任意两个元组  $t, s$ ：

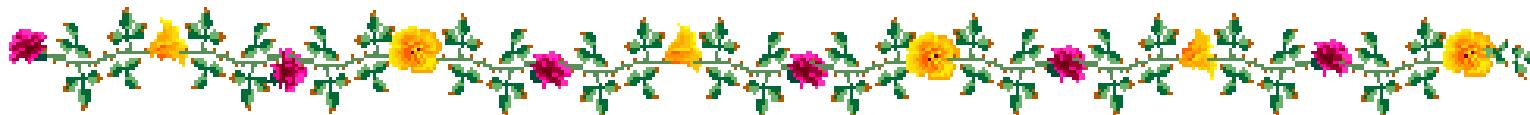
若  $t[X] = s[X]$ ，由于  $X \rightarrow Y$ ，有  $t[Y] = s[Y]$ ；再由  $Y \rightarrow Z$ ，  
有  $t[Z] = s[Z]$ ，所以  $X \rightarrow Z$  为  $F$  所蕴含，传递律得证。

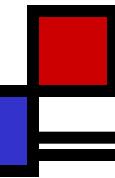
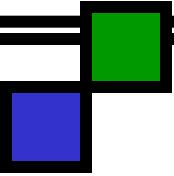




1. 根据A1, A2, A3这三条推理规则可以得到下面三条推理规则:

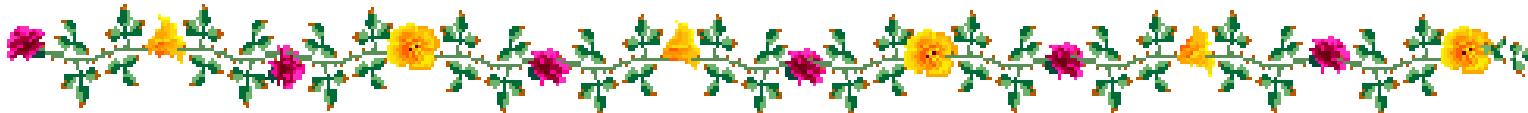
- 合并规则: 由 $X \rightarrow Y$ ,  $X \rightarrow Z$ , 有 $X \rightarrow YZ$ 。
- 伪传递规则: 由 $X \rightarrow Y$ ,  $WY \rightarrow Z$ , 有 $XW \rightarrow Z$ 。
- 分解规则: 由 $X \rightarrow Y$ 及  $Z \subseteq Y$ , 有 $X \rightarrow Z$ 。

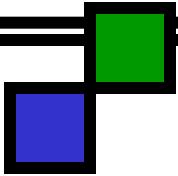




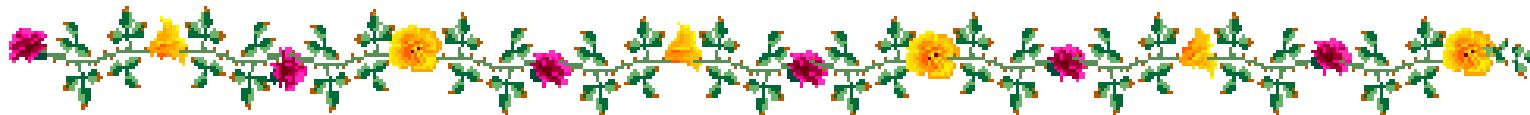
2.根据合并规则和分解规则，可得引理6.1

**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 ( $i=1, 2, \dots, k$ )



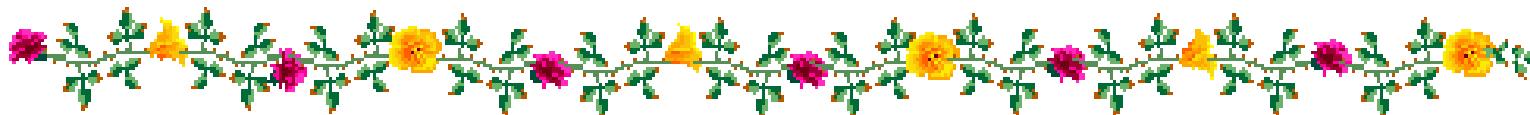


- ❖ Armstrong公理系统是有效的、完备的
  - **有效性:** 由 $F$ 出发根据Armstrong公理推导出来的每一个函数依赖一定在 $F^+$ 中；
  - **完备性:**  $F^+$ 中的每一个函数依赖，必定可以由 $F$ 出发根据Armstrong公理推导出来



**定义6.12** 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  $F^+$ 。

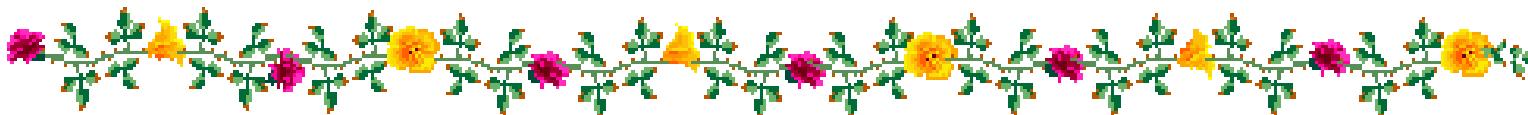
**定义6.13** 设  $F$  为属性集  $U$  上的一组函数依赖， $X \subseteq U$ ， $X_F^+ = \{ A | X \rightarrow A \text{ 能由 } F \text{ 根据 Armstrong 公理导出} \}$ ， $X_F^+$  称为属性集  $X$  关于函数依赖集  $F$  的闭包



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

$F^+ = \{ X \rightarrow \varphi, Y \rightarrow \varphi, Z \rightarrow \varphi, XY \rightarrow \varphi, XZ \rightarrow \varphi, YZ \rightarrow \varphi, XYZ \rightarrow \varphi,$   
 $X \rightarrow X, Y \rightarrow Y, Z \rightarrow Z, XY \rightarrow X, XZ \rightarrow X, YZ \rightarrow Y, XYZ \rightarrow X,$   
 $X \rightarrow Y, Y \rightarrow Z, XY \rightarrow Y, XZ \rightarrow Y, YZ \rightarrow Z, XYZ \rightarrow Y,$   
 $X \rightarrow Z, Y \rightarrow YZ, XY \rightarrow Z, XZ \rightarrow Z, YZ \rightarrow YZ, XYZ \rightarrow Z,$   
 $X \rightarrow XY, XY \rightarrow XY, XZ \rightarrow XY, XYZ \rightarrow XY,$   
 $X \rightarrow XZ, XY \rightarrow YZ, XZ \rightarrow XZ, XYZ \rightarrow YZ,$   
 $X \rightarrow YZ, XY \rightarrow XZ, XZ \rightarrow XY, XYZ \rightarrow XZ,$   
 $X \rightarrow ZYZ, XY \rightarrow XYZ, XZ \rightarrow XYZ, XYZ \rightarrow XYZ \}$

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

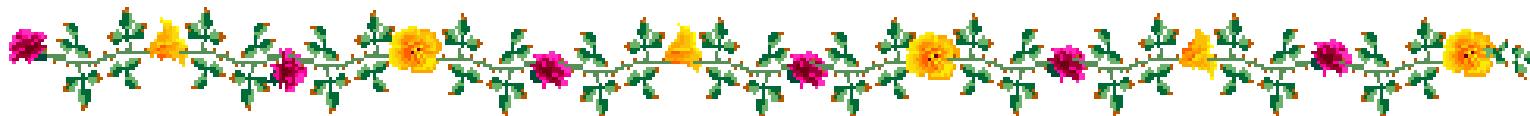


◆ 引理6.2

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 $Y \subseteq X_F^+$

◆ 用途

将判定 $X \rightarrow Y$ 是否能由 $F$ 根据**Armstrong**公理导出的问题，转化为求出 $X_F^+$ 、判定 $Y$ 是否为 $X_F^+$ 的子集的问题



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:  $X, F$  步骤:      输出:  $X_F^+$

(1) 令 $X^{(0)} = X, i=0$

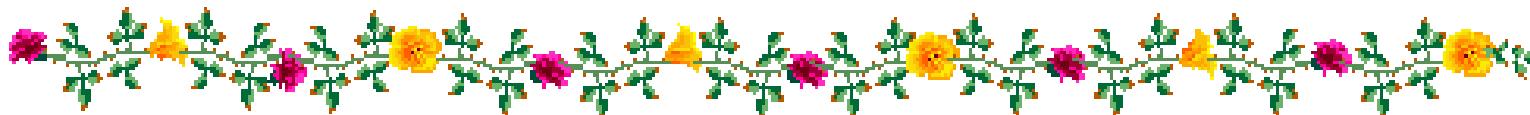
(2) 求 $B$ , 这里 $B = \{ A | (\exists V)(\exists W)(V \rightarrow W \in F \wedge V \subseteq X^{(i)} \wedge A \in W) \}$ ;

(3)  $X^{(i+1)} = B \cup X^{(i)}$

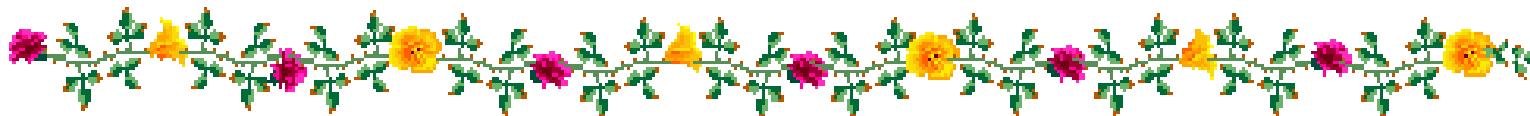
(4) 判断 $X^{(i+1)} = X^{(i)}$  吗?

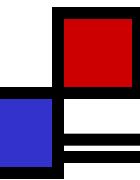
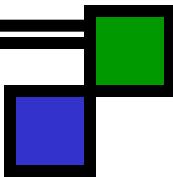
(5) 若相等或 $X^{(i)} = U$ , 则 $X^{(i)}$  就是 $X_F^+$ , 算法终止。

(6) 若否, 则  $i=i+1$ , 返回第 (2) 步。



对于算法6.1，令 $a_i = |X^{(i)}|$ ， $\{a_i\}$ 形成一个步长大于1的严格递增的序列，序列的上界是 $|U|$ ，因此该算法最多 $|U| - |X|$ 次循环就会终止。





[例1] 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 设  $X^{(0)} = AB$ ;

(1)  $X^{(1)} = AB \cup CD = ABCD$ 。

(2)  $X^{(0)} \neq X^{(1)}$   $X^{(2)} = X^{(1)}$   
 $\cup BE = ABCDE$ 。

(3)  $X^{(2)} = U$ , 算法终止  $\rightarrow (AB)_F = ABCDE$

◦

+



❖ 定理6.2 Armstrong公理系统是有效的、完备的

❖ 证明：

1. 有效性

可由定理6.1得证

2. 完备性

只需证明逆否命题：若函数依赖 $X \rightarrow Y$ 不能由 $F$ 从 Armstrong公理导出，那么它必然不为 $F$ 所蕴含



(1) 引理: 若  $V \rightarrow W$  成立, 且  $V \subseteq X_F^+$ , 则  $W \subseteq X_F^+$

(2) 构造一张二维表  $r$ , 它由下列两个元组构成, 可以证明  $r$  必是  $R(U, F)$  的一个关系, 即  $F^+$  中的全部函数依赖在  $r$  上成立。

$X_F^+$	$U-X_F^+$
11.....1	00.....0
11.....1	11.....1

(3) 若  $X \rightarrow Y$  不能由  $F$  从 Armstrong 公理导出, 则  $Y$  不是  $X_F^+$  的子集。



**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。

**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  $F^+ = G^+$ 。



**定义6.15** 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。亦称为**最小依赖集或最小覆盖**。

- (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。

[例2] 关系模式  $S < U, F >$ , 其中:

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

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

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

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

$F$  是最小覆盖, 而  $F'$  不是。

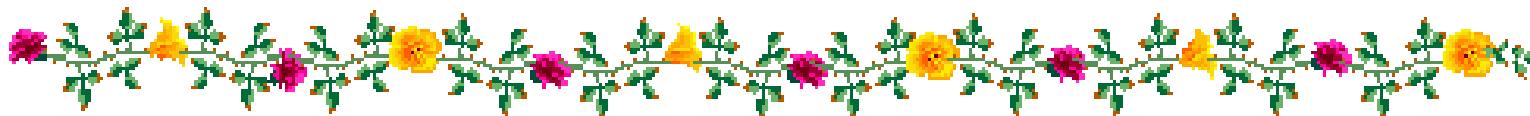
因为:  $F' - \{ Sno \rightarrow Mname \}$  与  $F'$  等价

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

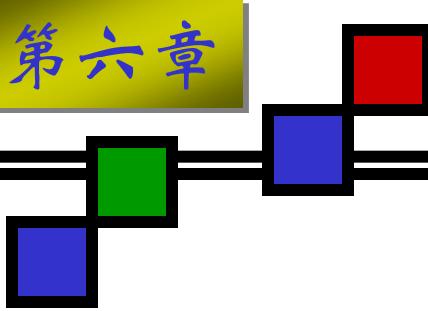


**定理6.3** 每一个函数依赖集 $F$ 均等价于一个极小函数依赖集 $F_m$ 。此 $F_m$ 称为 $F$ 的最小依赖集。

证明：构造性证明，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 $Y = A_1A_2 \dots A_k$ ,  $k > 2$ , 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。



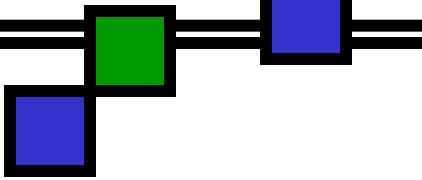
[例3]  $F = \{A \rightarrow B, B \rightarrow A, B \rightarrow C, A \rightarrow C, C \rightarrow A\}$

$F_{m1}$ 、 $F_{m2}$ 都是 $F$ 的最小依赖集：  $F_{m1} = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$

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

- ❖  $F$ 的最小依赖集 $F_m$ 不唯一
- ❖ 极小化过程( 定理6.3的证明 )也是检验 $F$ 是否为极小依赖集 的一个算法





## 6.1 问题的提出

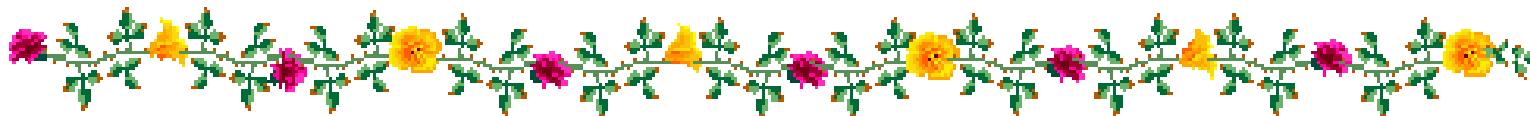
## 6.2 规范化

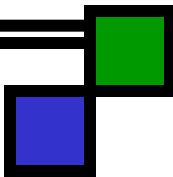
## 6.3 数据依赖的公理系统

## \*6.4 模式的分解

## 6.5 小结

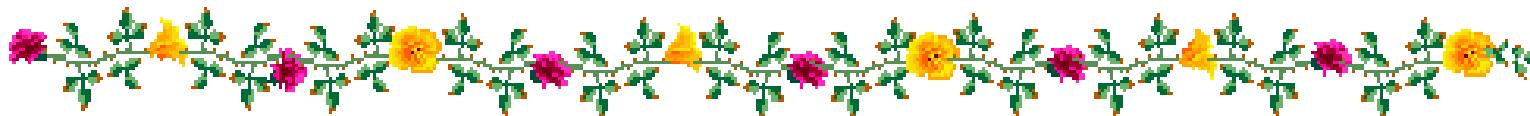
- ❖ 把低一级的关系模式分解为若干个高一级的关系模式的方法不是唯一的
- ❖ 只有能够保证分解后的关系模式与原关系模式等价，分解方法才有意义





三种模式分解等价的定义：

1. 分解具有无损连接性
2. 分解要保持函数依赖
3. 分解既要保持函数依赖，又要具有无损连接性



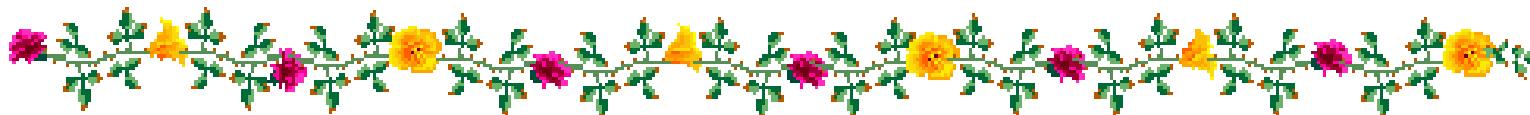
定义6.16 关系模式  $R<U, F>$  的一个分解：

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

n

$U = \bigcup_{i=1}^n U_i$ , 且不存在  $U_i \subseteq U_j$ ,  $F_i$  为  $F$  在  $U_i$  上的投影

定义6.17 函数依赖集合  $\{X \rightarrow Y \mid X \rightarrow Y \in F^+ \wedge XY \subseteq U_i\}$  的一个 覆盖  $F_i$  叫作  $F$  在属性  $U_i$  上的投影



例：**S-L** (**Sno, Sdept, Sloc**)

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

**S-L** ∈ 2NF

分解方法可以有多种：

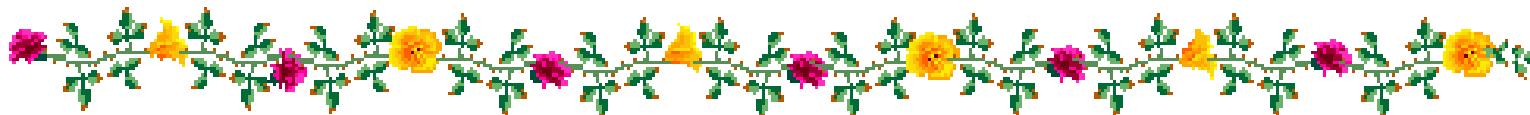
1. **S-L** 分解为三个关系模式： **SN(Sno)**  
**SD(Sdept)** **SO(Sloc)**

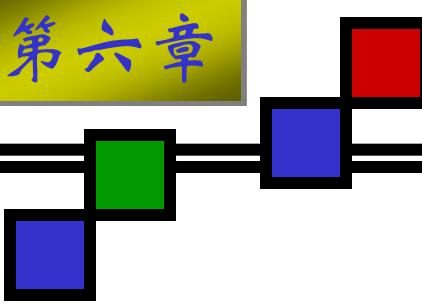
2. **SL** 分解为下面二个关系模式：

**NL(Sno, Sloc)** **DL(Sdept, Sloc)**

3. 将**SL** 分解为下面二个关系模式：

**ND(Sno, Sdept)** **NL(Sno, Sloc)**

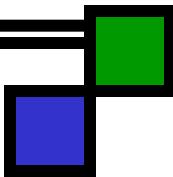




- ❖ 关系模式  $R<U, F>$  的一个分解  $\rho = \{ R_1 < U_1, F_1 >, R_2 < U_2, F_2 >, \dots, R_n < U_n, F_n > \}$

若  $R$  与  $R1$ 、 $R2$ 、 $\dots$ 、 $Rn$  自然连接的结果相等，则称关系模式  $R$  的这个分解  $\rho$  具有无损连接性 (*Lossless join*)

- ❖ 具有无损连接性的分解保证不丢失信息
- ❖ 无损连接性不一定能解决插入异常、删除异常、修改复杂、数据冗余等问题



### 第3种分解方法具有无损连接性

问题:这种分解方法没有保持原关系中的函数依赖

- $SL$  中的函数依赖  $Sdept \rightarrow Sloc$  没有投影到关系模式  $ND$ 、 $NL$  上

设关系模式  $R<U, F>$  被分解为若干个关系模式

$$R_1<U_1, F_1>, R_2<U_2, F_2>, \dots, R_n<U_n, F_n>$$

(其中  $U = U_1 \cup U_2 \cup \dots \cup U_n$ , 且不存在  $U_i \subseteq U_j$ ,  $F_i$  为  $F$  在  $U_i$  上的 投影), 若  $F$  所逻辑蕴含的函数依赖一定也由分解得到的某 个关系模式中的函数依赖  $F_i$  所逻辑蕴含, 则称关系模式  $R$  的 这个分解是保持函数依赖的 (**Preserve dependency**)

4. 将 $SL$ 分解为下面二个关系模式：

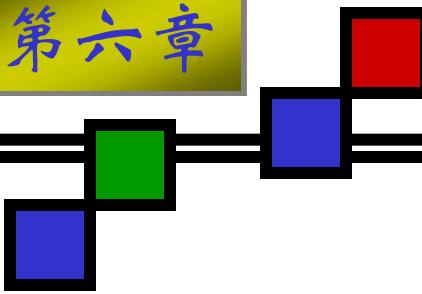
$ND(Sno, Sdept)$

$DL(Sdept, Sloc)$

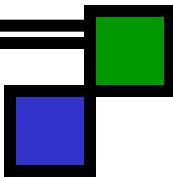
这种分解方法就保持了函数依赖

- ❖ 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。

- 第1种分解方法既不具有无损连接性，也未保持函数依赖，它不是原关系模式的一个等价分解
- 第2种分解方法保持了函数依赖，但不具有无损连接性 第3种分解方法具有无损连接性，但未持函数依赖 第4种分解方法既具有无损连接性，又保持了函数依赖



- ❖ 算法6.2 判别一个分解的无损连接性
- ❖ 算法6.3 (合成法) 转换为3NF的保持函数依赖的分解。
- ❖ 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
- ❖ 算法6.5 (分解法) 转换为BCNF的无损连接分解
- ❖ 算法6.6 达到4NF的具有无损连接性的分解



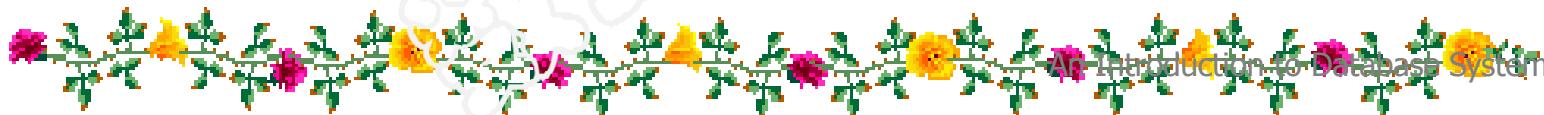
## 6.1 问题的提出

## 6.2 规范化

## 6.3 数据依赖的公理系统

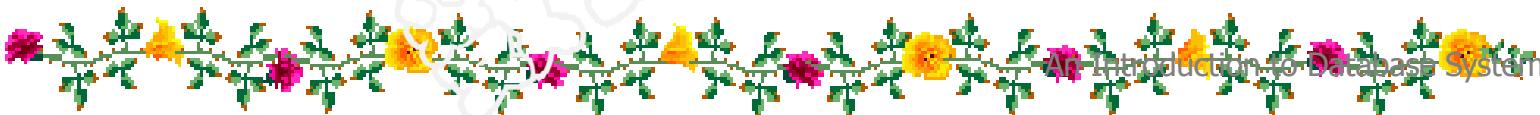
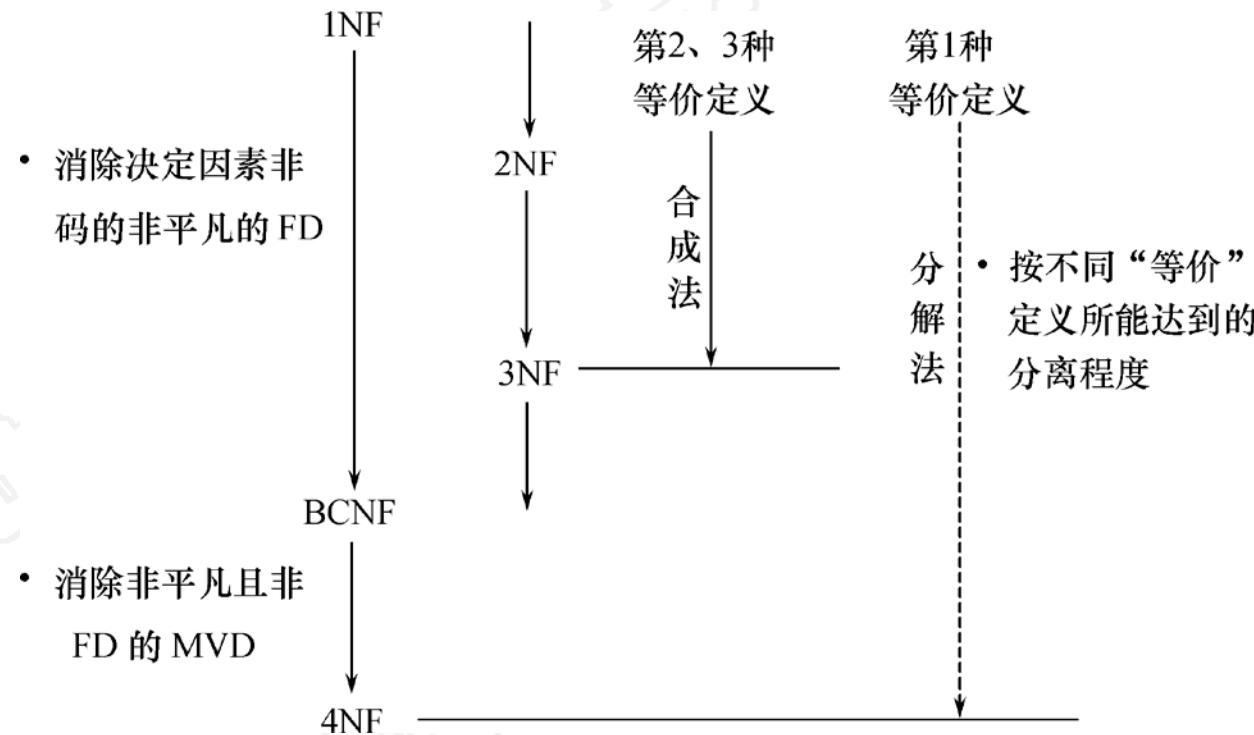
## \*6.4 模式的分解

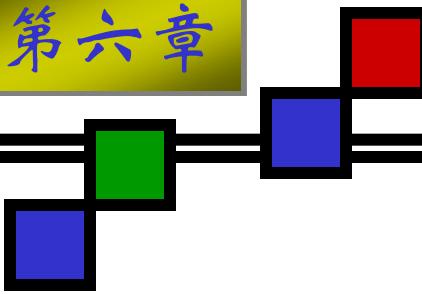
## 6.5 小结



## 关系模式的规范化，其基本思想

:





- ❖ 若要求分解具有无损连接性，那么模式分解一定能够达到**4NF**
- ❖ 若要求分解保持函数依赖，那么模式分解一定能够达到**3NF**，但不一定能够达到**BCNF**
- ❖ 若要求分解既具有无损连接性，又保持函数依赖，则模式分解一定能够达到**3NF**，但不一定能够达到**BCNF**

- ❖ 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