



# 5 范式理论

刘跃文 博士，副教授

西安交通大学管理学院

信息管理与电子商务系

[liuyuewen@mail.xjtu.edu.cn](mailto:liuyuewen@mail.xjtu.edu.cn)

V1, 2017-10-24

# 8.1 Combine Schemas? 合并模式 [理解]

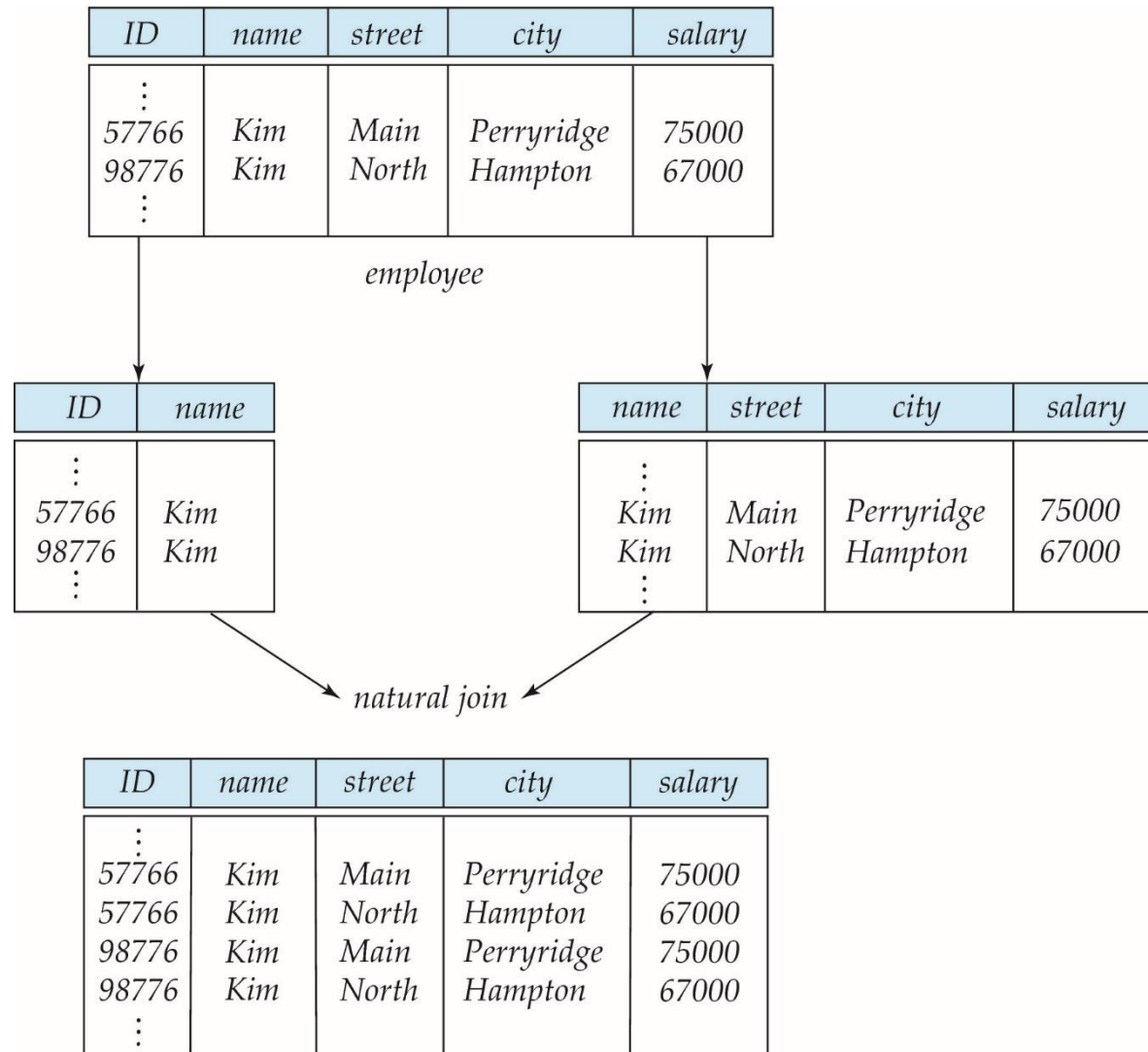
- Suppose we combine *instructor* and *department* into *inst\_dept*
  - (No connection to relationship set *inst\_dept*)
- Result is possible repetition of information

ID	name	salary	dept_name	building	budget
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

# What About Smaller Schemas? [理解]

- Suppose we had started with *inst\_dept*. How would we know to split up (**decompose**) 拆分 it into *instructor* and *department*?
- Write a rule “if there were a schema (*dept\_name*, *building*, *budget*), then *dept\_name* would be a candidate key”
- Denote as a **functional dependency** 函数依赖:  
$$dept\_name \rightarrow building, budget$$
- In *inst\_dept*, because *dept\_name* is not a candidate key, the building and budget of a department may have to be repeated.
  - This indicates the need to decompose *inst\_dept*
- Not all decompositions are good. Suppose we decompose *employee*(*ID*, *name*, *street*, *city*, *salary*) into  
*employee1* (*ID*, *name*)  
*employee2* (*name*, *street*, *city*, *salary*)
- The next slide shows how we lose information -- we cannot reconstruct the original *employee* relation -- and so, this is a **lossy decomposition** 有损分解.

# A Lossy Decomposition 有损分解 [理解]



# Example of Lossless-Join Decomposition 无损分解 [理解]

- **Lossless join decomposition**

- Decomposition of  $R = (A, B, C)$

$$R_1 = (A, B) \quad R_2 = (B, C)$$

A	B	C
$\alpha$	1	A
$\beta$	2	B

$r$

A	B
$\alpha$	1
$\beta$	2

$\Pi_{A,B}(r)$

B	C
1	A
2	B

$\Pi_{B,C}(r)$

$\Pi_A(r) \bowtie \Pi_B(r)$

A	B	C
$\alpha$	1	A
$\beta$	2	B

## 8.2 First Normal Form 第一范式 1NF [掌握]

- Domain is **atomic 原子域** if its elements are considered to be indivisible units
  - Examples of non-atomic domains:
    - Set of names, composite attributes
    - Identification numbers like CS101 that can be broken up into parts
- A relational schema R is in **first normal form** if the domains of all attributes of R are atomic [第一范式的定义]
- Non-atomic values complicate storage and encourage redundant (repeated) storage of data
  - Example: Set of accounts stored with each customer, and set of owners stored with each account
  - We assume all relations are in first normal form (and revisit this in Chapter 22: Object Based Databases)



编号	品名	进货		销售		备注
		数量	单价	数量	单价	

编号	品名	进货数量	进货单价	销售数量	销售单价	备注

# First Normal Form (Cont'd) [理解]

- Atomicity 原子性 is actually a property of how the elements of the domain are used. [重要的是 怎么用，而不是 是什么].
  - Example: Strings would normally be considered indivisible
  - Suppose that students are given roll numbers which are strings of the form *CS0012* or *EE1127*
  - If the first two characters are extracted to find the department, the domain of roll numbers is not atomic.
  - Doing so is a bad idea: leads to encoding of information in application program rather than in the database.
- 身份证号是原子域吗？



## 8.3 Goal — Devise a Theory for the Following

- Decide whether a particular relation  $R$  is in “good” form.
- In the case that a relation  $R$  is not in “good” form, decompose it into a set of relations  $\{R_1, R_2, \dots, R_n\}$  such that
  - each relation is in good form
  - the decomposition is a lossless-join decomposition
- Our theory is based on:
  - functional dependencies 函数依赖

# Functional Dependencies 函数依赖 [掌握]

- Constraints on the set of legal relations.
- Require that the value for a certain set of attributes determines uniquely the value for another set of attributes.
- A functional dependency is a generalization of the notion of a *key*.  
函数依赖是 键 的一般化概念。

# Functional Dependencies (Cont.)

- Let  $R$  be a relation schema

$$\alpha \subseteq R \text{ and } \beta \subseteq R$$

- The **functional dependency**

$$\alpha \rightarrow \beta$$

**holds on**  $R$  if and only if for any legal relations  $r(R)$ , whenever any two tuples  $t_1$  and  $t_2$  of  $r$  agree on the attributes  $\alpha$ , they also agree on the attributes  $\beta$ . That is,

$$t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$$

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

1	4
1	5
3	7

- On this instance,  $A \rightarrow B$  does **NOT** hold, but  $B \rightarrow A$  does hold.

# Functional Dependencies (Cont.)

- $K$  is a superkey for relation schema  $R$  if and only if  $K \rightarrow R$
- $K$  is a candidate key for  $R$  if and only if
  - $K \rightarrow R$ , and
  - for no  $\alpha \subset K, \alpha \rightarrow R$  例如：学号，姓名，成绩
- Functional dependencies allow us to express constraints that cannot be expressed using superkeys. Consider the schema:

*inst\_dept* (*ID*, *name*, *salary*, *dept\_name*, *building*, *budget*).

We expect these functional dependencies to hold:

*dept\_name*  $\rightarrow$  *building*

and *ID*  $\rightarrow$  *building*

but would not expect the following to hold:

*dept\_name*  $\rightarrow$  *salary*

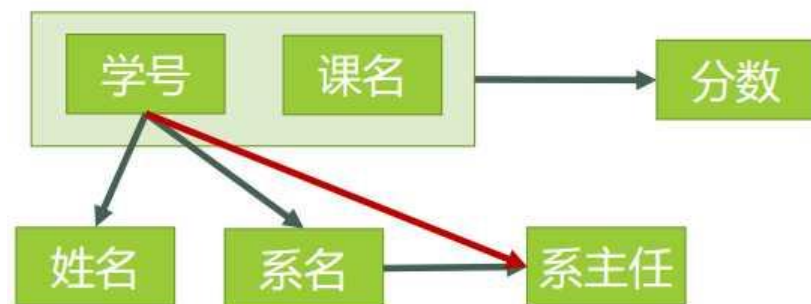
# Functional Dependencies (Cont.) [掌握]

- A functional dependency is **trivial 平凡的** if it is satisfied by all instances of a relation
  - Example:
    - $ID, name \rightarrow ID$
    - $name \rightarrow name$
  - In general,  $\alpha \rightarrow \beta$  is trivial if  $\beta \subseteq \alpha$

# 例：找出下表中的函数依赖 [掌握]

学号	姓名	系名	系主任	课名	分数
1022211101	李小明	经济系	王强	高等数学	95
1022211101	李小明	经济系	王强	大学英语	87
1022211101	李小明	经济系	王强	普通化学	76
1022211102	张莉莉	经济系	王强	高等数学	72
1022211102	张莉莉	经济系	王强	大学英语	98
1022211102	张莉莉	经济系	王强	计算机基础	88
1022511101	高芳芳	法律系	刘玲	高等数学	82
1022511101	高芳芳	法律系	刘玲	法学基础	82

- 以下函数依赖关系成立：
  - 学号  $\rightarrow$  姓名
  - 系名  $\rightarrow$  系主任
  - 学号  $\rightarrow$  系主任
  - (学号, 课名)  $\rightarrow$  分数
- 但以下函数依赖关系则不成立：
  - 学号  $\rightarrow$  课名
  - 学号  $\rightarrow$  分数
  - 课名  $\rightarrow$  系主任
  - (学号, 课名)  $\rightarrow$  姓名



- 完全函数依赖

- 在一张表中，若  $X \rightarrow Y$ ，且对于  $X$  的任何一个真子集（假如属性组  $X$  包含超过一个属性的话）， $X' \rightarrow Y$  不成立，那么称  $Y$  对于  $X$  **完全函数依赖**。

$$X \xrightarrow{F} Y$$

- 部分函数依赖

- 假如  $Y$  函数依赖于  $X$ ，但同时  $Y$  并不完全函数依赖于  $X$ ，那么就称  $Y$  **部分函数依赖于**  $X$ 。

$$X \xrightarrow{P} Y$$

- 传递函数依赖

- 假如  $Z$  函数依赖于  $Y$ ，且  $Y$  函数依赖于  $X$ （且  $Y$  不包含于  $X$ ， $X$  不函数依赖于  $Y$ ），那么就称  $Z$  **传递函数依赖于**  $X$ 。

$$X \xrightarrow{T} Z$$



- 存在的问题：
  - 数据冗余过大
  - 插入异常
  - 删除异常
  - 修改异常

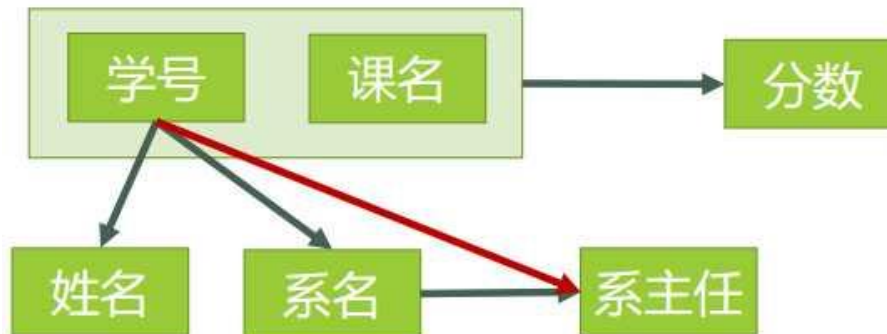
学号	姓名	系名	系主任	课名	分数
1022211101	李小明	经济系	王强	高等数学	95
1022211101	李小明	经济系	王强	大学英语	87
1022211101	李小明	经济系	王强	普通化学	76
1022211102	张莉莉	经济系	王强	高等数学	72
1022211102	张莉莉	经济系	王强	大学英语	98
1022211102	张莉莉	经济系	王强	计算机基础	88
1022511101	高芳芳	法律系	刘玲	高等数学	82
1022511101	高芳芳	法律系	刘玲	法学基础	82

## 第二范式 2NF [掌握]

- 2NF在1NF的基础上，消除了非主属性对于候选码的部分函数依赖。
  - **主属性** 包含在任何一个候选码中的属性为主属性。
  - **非主属性** 不包含在任何一个候选码中的属性为非主属性。

# 不满足2NF的原因 [掌握]

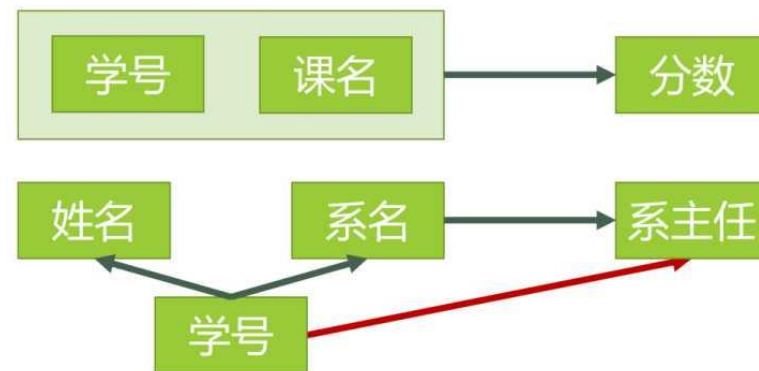
- 对于  $(\text{学号}, \text{课名}) \rightarrow \text{姓名}$ ，有  $\text{学号} \rightarrow \text{姓名}$ ，存在非主属性 **姓名** 对码  $(\text{学号}, \text{课名})$  的部分函数依赖。
- 对于  $(\text{学号}, \text{课名}) \rightarrow \text{系名}$ ，有  $\text{学号} \rightarrow \text{系名}$ ，存在非主属性 **系名** 对码  $(\text{学号}, \text{课名})$  的部分函数依赖。
- 对于  $(\text{学号}, \text{课名}) \rightarrow \text{系主任}$ ，有  $\text{学号} \rightarrow \text{系主任}$ ，存在非主属性 **系主任** 对码  $(\text{学号}, \text{课名})$  的部分函数依赖。





学号	课名	分数
1022211101	高等数学	95
1022211101	大学英语	87
1022211101	普通化学	76
1022211102	高等数学	72
1022211102	大学英语	98
1022211102	计算机基础	88
1022511101	高等数学	82
1022511101	法学基础	82

学号	姓名	系名	系主任
1022211101	李小明	经济系	王强
1022211102	张莉莉	经济系	王强
1022511101	高芳芳	法律系	刘玲



# 第三范式 3NF [掌握]

- **第三范式 (3NF)** 3NF在2NF的基础之上，消除了非主属性对于码的传递函数依赖。
  - 如果存在非主属性对于码的传递函数依赖，则不符合3NF的要求。
  - 学号  $\rightarrow$  系名，同时 系名  $\rightarrow$  系主任，所以存在非主属性系主任对于码学号的传递函数依赖，所以学生表的设计，不符合3NF的要求。

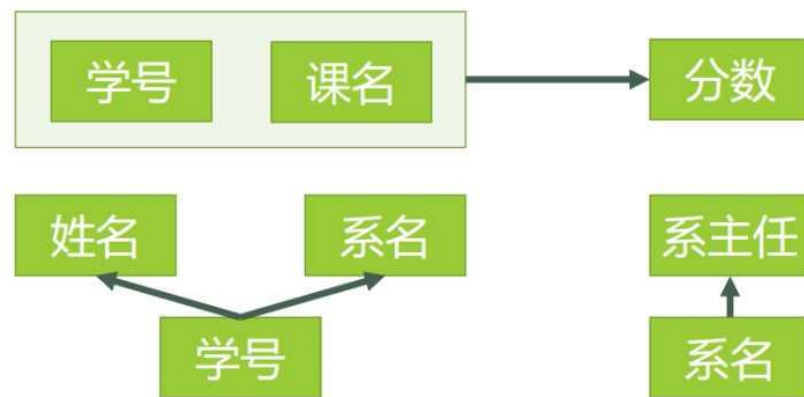




学号	课名	分数
1022211101	高等数学	95
1022211101	大学英语	87
1022211101	普通化学	76
1022211102	高等数学	72
1022211102	大学英语	98
1022211102	计算机基础	88
1022511101	高等数学	82
1022511101	法学基础	82

学号	姓名	系名
1022211101	李小明	经济系
1022211102	张莉莉	经济系
1022511101	高芳芳	法律系

系名	系主任
经济系	王强
经济系	王强
法律系	刘玲



# Boyce-Codd Normal Form BCNF

## BC范式 [掌握]

A relation schema  $R$  is in BCNF with respect to a set  $F$  of functional dependencies if for all functional dependencies in  $F^+$  of the form

$$\alpha \rightarrow \beta$$

where  $\alpha \subseteq R$  and  $\beta \subseteq R$ , at least one of the following holds:

- $\alpha \rightarrow \beta$  is trivial (i.e.,  $\beta \subseteq \alpha$ )
- $\alpha$  is a superkey for  $R$

Example schema *not* in BCNF:

*instr\_dept* (*ID\_name*, *salary*, *dept\_name*, *building*, *budget*)

because *dept\_name*  $\rightarrow$  *building*, *budget*

holds on *instr\_dept*, but *dept\_name* is not a superkey

# 不满足BC范式时的分解方法[理解]

- Suppose we have a schema  $R$  and a non-trivial dependency  $\alpha \rightarrow \beta$  causes a violation of BCNF.

We decompose  $R$  into:

$(\alpha \cup \beta)$

$(R - (\beta - \alpha))$  注：没有就不减。

- In our example,
  - $\alpha = dept\_name$
  - $\beta = building, budget$and  $inst\_dept$  is replaced by
  - $(\alpha \cup \beta) = (dept\_name, building, budget)$
  - $(R - (\beta - \alpha)) = (ID, name, salary, dept\_name)$



## 另一个例子：学生与导师 [理解]

- 每个老师只能在一个系工作
- 每个学生可以修不同系的双学位，每个学位可以有一名导师
- 表设计：
  - 老师ID，系名
  - 学生ID，系名，老师ID
- 满足第三范式，不满足BC范式（老师ID→系名）

# Comparison of BCNF and 3NF [理解]

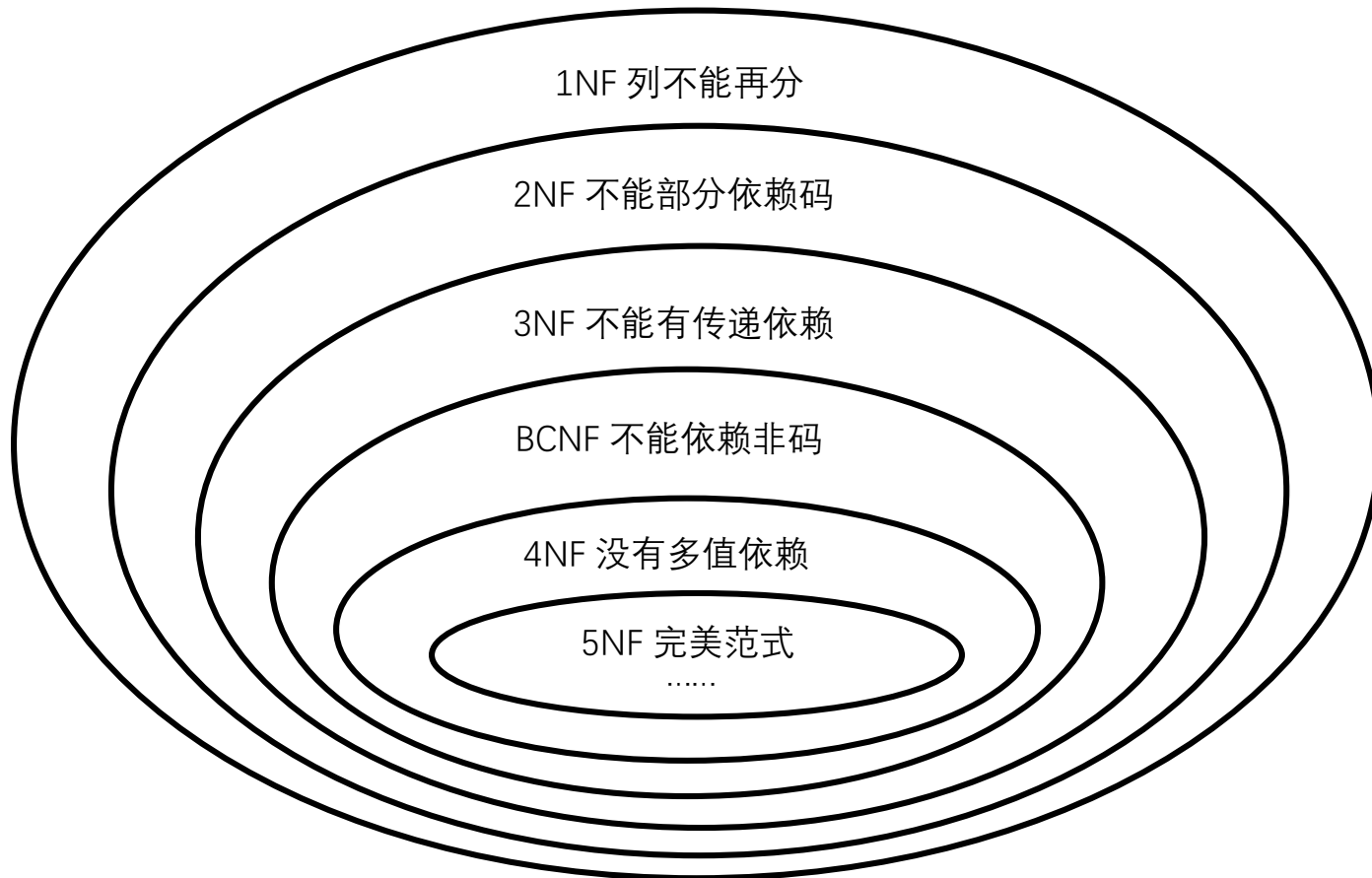
- It is always possible to decompose a relation into a set of relations that are in 3NF such that:
  - the decomposition is lossless
  - the dependencies are preserved 保持依赖
- It is always possible to decompose a relation into a set of relations that are in BCNF such that:
  - the decomposition is lossless
  - it may not be possible to preserve dependencies.

## 第四范式 [理解]

- 另一个例子：
  - 假设每个学生可以修两个专业的学位
  - 同时，每个学生可以有多个电话号码
- 表设计：
  - 学生ID，系名，电话号码
- 没有违反3NF，BCNF
- 违反第四范式：存在多值依赖
- 拆分：学生ID，系名；学生ID，电话号码


学生ID	系名	电话号码
1001	管理	号码1
1001	管理	号码2
1001	管理	号码3
1001	计算机	号码1
1001	计算机	号码3
1001	计算机	号码3

- 多值函数依赖  $\alpha \twoheadrightarrow \beta$
- A relation schema  $R$  is in **4NF** with respect to a set  $D$  of functional and multivalued dependencies if for all multivalued dependencies in  $D^+$  of the form  $\alpha \twoheadrightarrow \beta$ , where  $\alpha \subseteq R$  and  $\beta \subseteq R$ , at least one of the following hold:
  - $\alpha \twoheadrightarrow \beta$  is trivial (i.e.,  $\beta \subseteq \alpha$  or  $\alpha \cup \beta = R$ )
  - $\alpha$  is a superkey for schema  $R$
- If a relation is in 4NF it is in BCNF



# 小结

- 函数依赖：平凡/非平凡；完全/部分；传递
- 分解：有损分解/无损分解
- 范式：1NF, 2NF, 3NF, BCNF, 4NF



谢谢！

liuyuewen@xjtu.edu.cn