

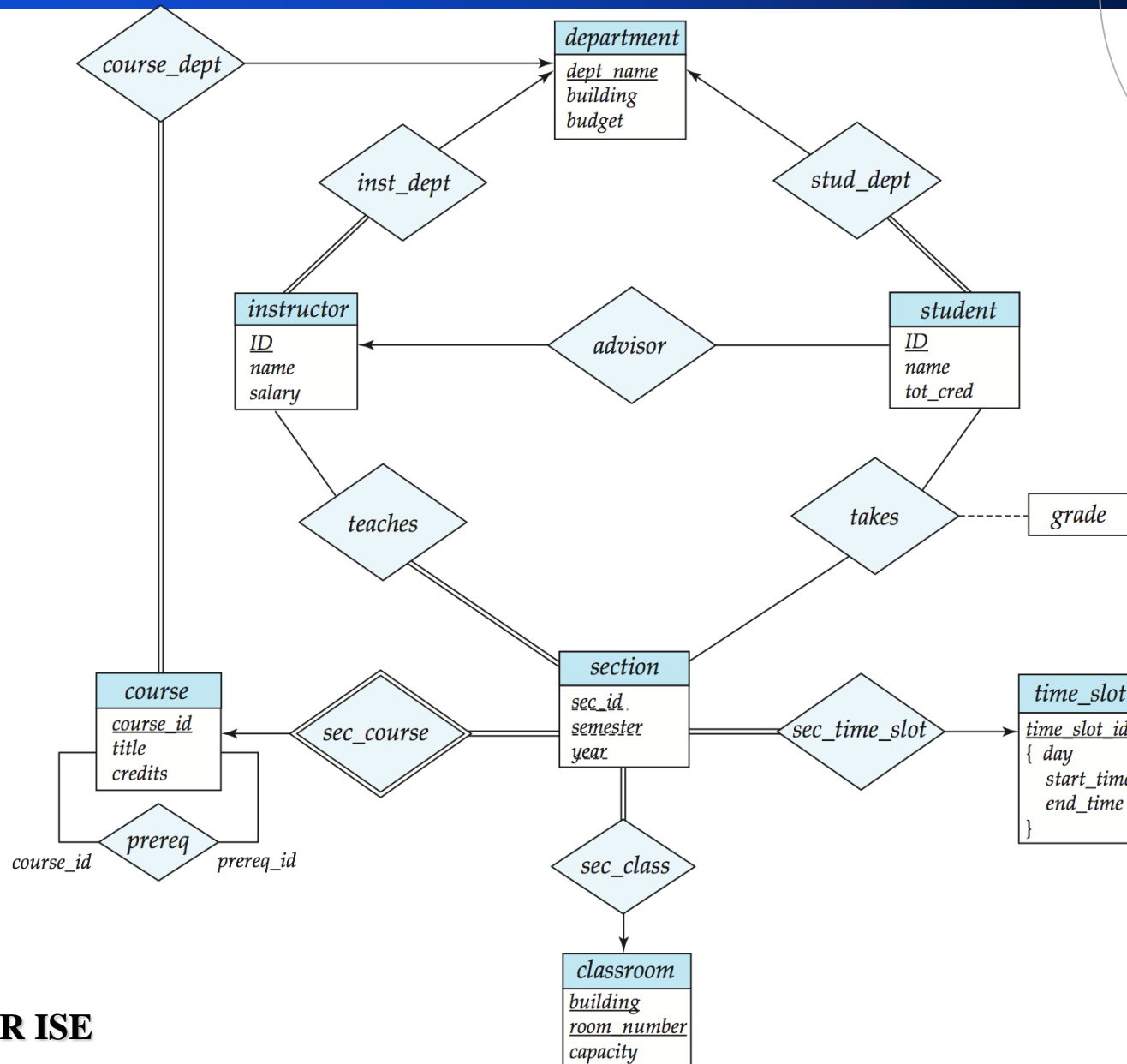
# Principles of Database Systems



## Relational Database Design



# E-R diagram for the University Enterprise





# Features of Good Relational Designs

Which is better?

A

*instructor(ID, name, dept name, salary)*  
*department(dept name, building, budget)*

VS

B

*inst dept (ID, name, salary, dept name, building, budget)*

提交

# Features of Good Relational Designs



<i>ID</i>	<i>name</i>	<i>salary</i>	<i>dept_name</i>	<i>building</i>	<i>budget</i>
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

# Features of Good Relational Designs



- Are all decompositions(分解) of schemas helpful?

*employee (ID, name, street, city, salary)*

VS

*employee1 (ID, name)*

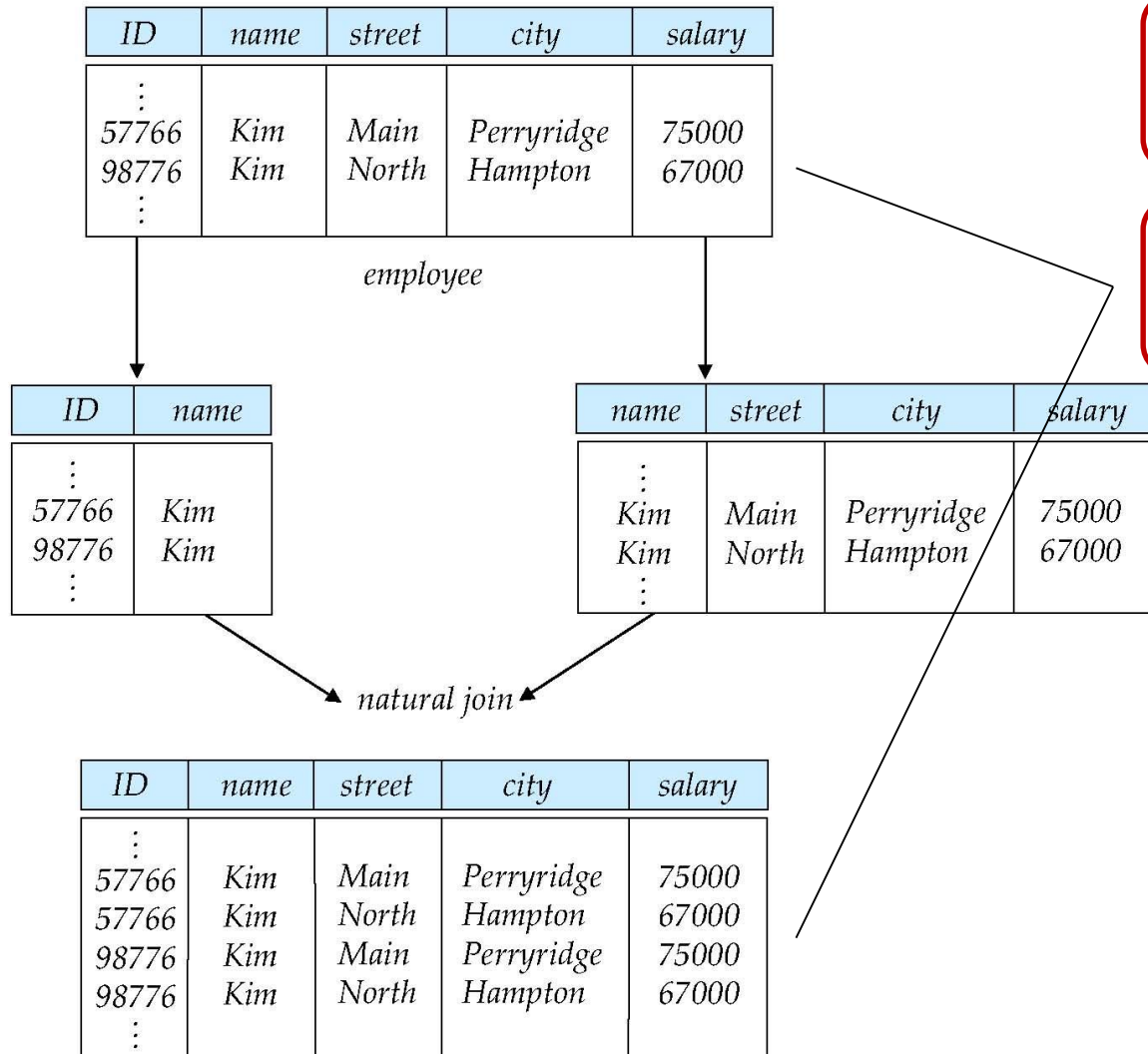
*employee2 (name, street, city, salary)*

# Features of Good Relational Designs



*lossless decompositions*  
(无损分解)

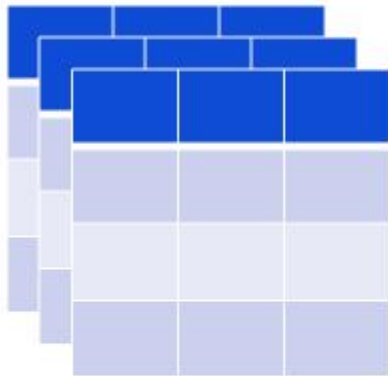
*lossy decompositions*  
(有损分解)



# Outline



## Relation schemes

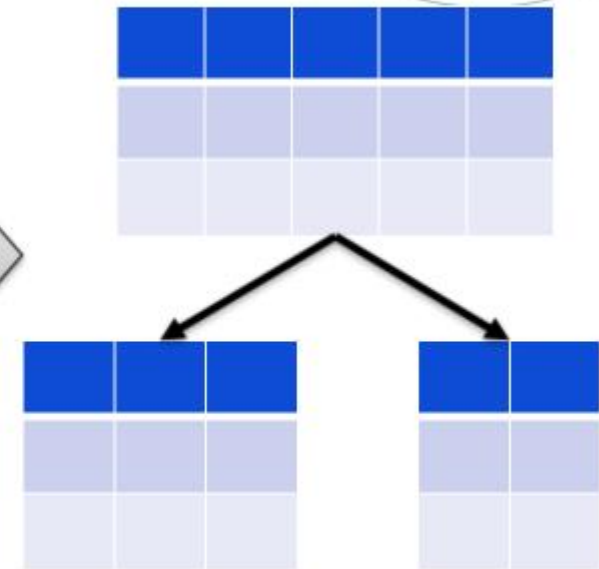


## Analysis



1. Functional Dependencies
2. Normal Forms

## Decomposition



3. Functional-Dependency Theory
4. BCNF Decomposition Algorithm





# Functional Dependencies(函数依赖)

# Functional Dependencies



- Definition

- Consider a relation schema  $r(R)$ , and let  $\alpha \subseteq R$  and  $\beta \subseteq R$ .
- Given an instance of  $r(R)$ , we say that the instance satisfies the **functional dependency**  $\alpha \rightarrow \beta$  if for all pairs of tuples  $t_i$  and  $t_j$  in the instance

$$t_i[\alpha] = t_j[\alpha] \rightarrow t_i[\beta] = t_j[\beta]$$

- We say that  **$\alpha \rightarrow \beta$  holds on schema  $r(R)$**  when every legal instance of schema  $r(R)$  satisfies the functional dependency  $\alpha \rightarrow \beta$ .

# Functional Dependencies



学号	姓名	院系	系主任	课程号	得分
sno	sname	dept	dleader	cno	score
<u>S1</u>	张翠山	武当派	张三丰	<u>C1</u>	90
<u>S1</u>	张翠山	武当派	张三丰	<u>C2</u>	80
<u>S2</u>	谢逊	明教	张无忌	<u>C3</u>	100
<u>S2</u>	谢逊	明教	张无忌	<u>C3</u>	90

✓  $sno \rightarrow sname$

✓  $cno \nrightarrow score$

✓  $sname \nrightarrow dept$

✓  $sname \nrightarrow score$

# Keys and Functional Dependencies



- Using the functional-dependency notation we can we define **super key**.
  - ***K is a superkey*** of  $r(R)$  if the functional dependency  $K \rightarrow R$  holds on  $r(R)$ .
  - ***K is a superkey*** if, for every legal instance of  $r(R)$ , for every pair of tuples  $t_i$  and  $t_j$  from the instance,
$$t_i[K] = t_j[K] \rightarrow t_i = t_j$$
- ***K is a candidate key*** for  $R$  if and only if
  - $K \rightarrow R$ , and
  - for no  $\alpha \subset K$ ,  $\alpha \rightarrow R$

# Functional Dependencies



- Some functional dependencies are said to be **trivial**(平凡的) because they are satisfied by all relations.
  - $A \rightarrow A$
  - $AB \rightarrow A$
- A functional dependency of the form  $\alpha \rightarrow \beta$  is **trivial** if  $\beta \subseteq \alpha$ .
- We will use the notation  $F^+$  to denote the **closure**(闭包) of the set  $F$ , that is, the set of all functional dependencies that can be inferred given the set  $F$ .

# Trivial(平凡的) Functional Dependencies



- $\alpha \rightarrow \beta$  is trivial if  $\beta \subseteq \alpha$

学号	姓名	院系	系主任	课程号	得分
sno	sname	dept	dleader	cno	score
<u>S1</u>	张翠山	武当派	张三丰	<u>C1</u>	90
<u>S1</u>	张翠山	武当派	张三丰	<u>C2</u>	80
<u>S2</u>	谢逊	明教	张无忌	<u>C3</u>	100
<u>S2</u>	谢逊	明教	张无忌	<u>C3</u>	90

(姓名, 院系)  $\rightarrow$  院系

(姓名, 院系)  $\rightarrow$  姓名

(张翠山, 武当山)  $\rightarrow$  武当山; (明教, 张无忌)  $\rightarrow$  张无忌

# Functional Dependencies



- Some functional dependencies are said to be **trivial**(平凡的) because they are satisfied by all relations.
  - $A \rightarrow A$
  - $AB \rightarrow A$
- A functional dependency of the form  $\alpha \rightarrow \beta$  is **trivial** if  $\beta \subseteq \alpha$ .
- We will use the notation  $F^+$  to denote the **closure**(闭包) of the set  $F$ , that is, the set of all functional dependencies that can be inferred given the set  $F$ .



# Functional-Dependency Theory



# Closure of a Set of Functional Dependencies



- **Armstrong's axioms**(公理)

- **Reflexivity rule**(自反律). If  $\alpha$  is a set of attributes and  $\beta \subseteq \alpha$ , then  $\alpha \rightarrow \beta$  holds.
- **Augmentation rule**(增补律). If  $\alpha \rightarrow \beta$  holds and  $\gamma$  is a set of attributes, then  $\gamma\alpha \rightarrow \gamma\beta$  holds.
- **Transitivity rule**(传递律). If  $\alpha \rightarrow \beta$  holds and  $\beta \rightarrow \gamma$  holds, then  $\alpha \rightarrow \gamma$  holds.

# Closure of a Set of Functional Dependencies



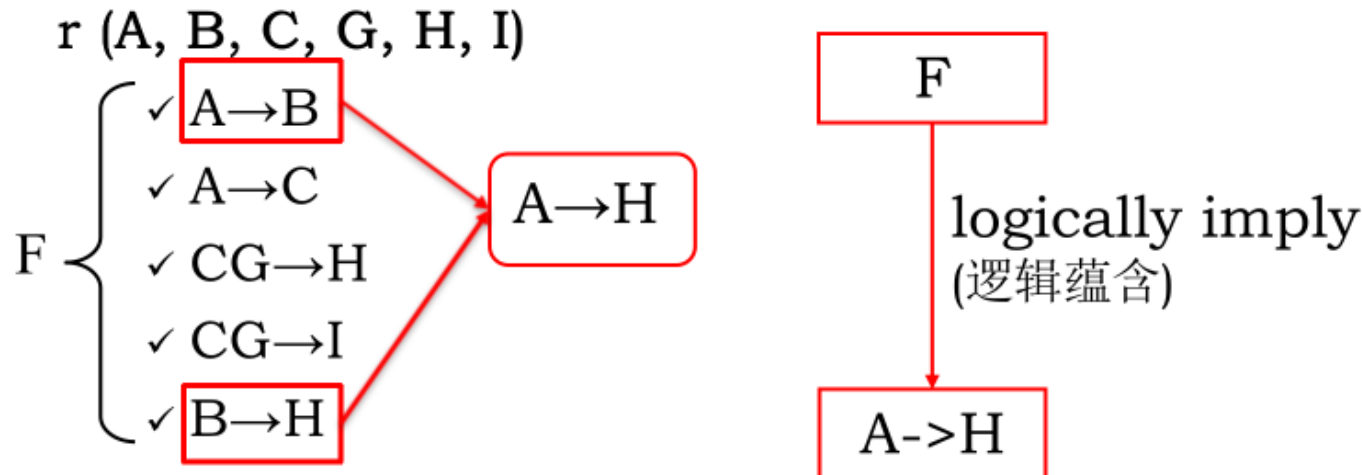
- additional rules
  - **Union rule**(合并律). If  $\alpha \rightarrow \beta$  holds and  $\alpha \rightarrow \gamma$  holds, then  $\alpha \rightarrow \beta\gamma$  holds.
  - **Decomposition rule**(分解律). If  $\alpha \rightarrow \beta\gamma$  holds, then  $\alpha \rightarrow \beta$  holds and  $\alpha \rightarrow \gamma$  holds.
  - **Pseudotransitivity rule**(伪传递律). If  $\alpha \rightarrow \beta$  holds and  $\gamma\beta \rightarrow \delta$  holds, then  $\alpha\gamma \rightarrow \delta$  holds.

# Closure of a Set of Functional Dependencies



- Given a relational schema  $r(R)$ , a functional dependency  $f$  on  $R$  is **logically implied** (逻辑蕴含) by a set of functional dependencies  $F$  on  $r$  if every instance of  $r(R)$  that satisfies  $F$  also satisfies  $f$ .

- Example

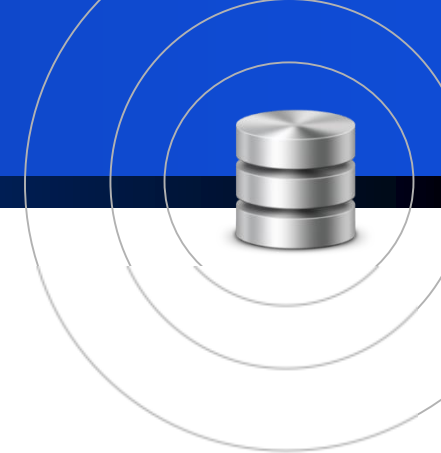


# Closure of a Set of Functional Dependencies



- Let  $F$  be a set of functional dependencies. The **closure** (闭包) of  $F$ , denoted by  $F^+$ , is the set of all functional dependencies logically implied by  $F$ .

# Closure (闭包) of F



- $F^+ = F$
- Repeat
  1. For each  $\alpha \rightarrow \beta$  in  $F^+$   
Using **reflexivity** (自反律) and **augmentation** (增补律) to generate new functional dependency.
  2. For each  $\alpha \rightarrow \beta, \gamma \rightarrow \sigma$  pair in  $F^+$   
Using **transitivity** (传递律) to generate new functional dependency.
  3. Renew  $F^+$
- Until  $F^+$  unchanged



# Normal Forms

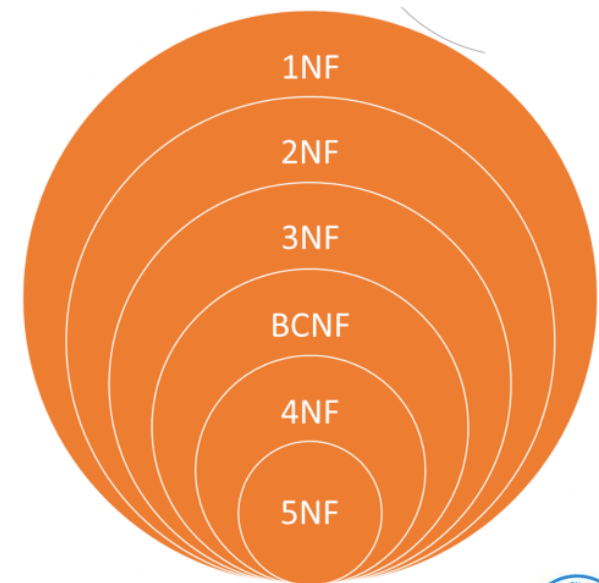
## -范式

# Atomic Domains and First Normal Form



- A domain is **atomic**(原子的) if elements of the domain are considered to be indivisible(不可分的) units.
- We say that a relation schema  $R$  is **in first normal form** (1NF) if the domains of all attributes of  $R$  are atomic.

$R \in 1NF$



# Atomic Domains



- The **E-R model** allows entity sets and relationship sets to have attributes that have some degree of substructure(子结构).
- However, when we **create tables** from E-R designs that contain these types of attributes, we eliminate(消除) this substructure.
- A domain is **atomic**(原子的) if elements of the domain are considered to be indivisible(不可分的) units.

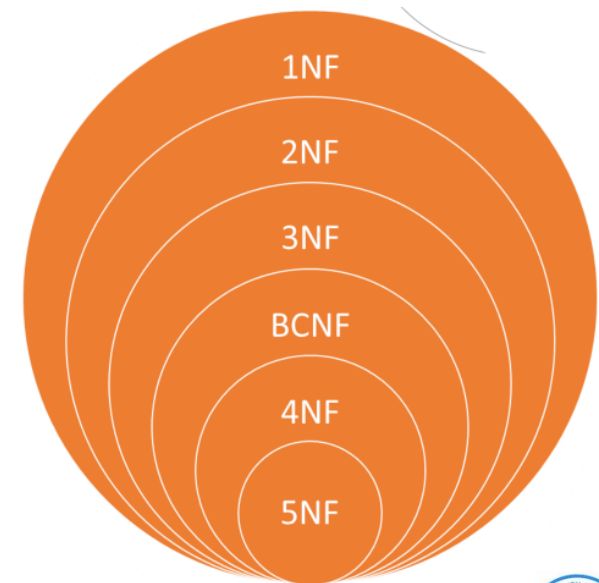


# Atomic Domains and First Normal Form



- A domain is **atomic**(原子的) if elements of the domain are considered to be indivisible(不可分的) units.
- We say that a relation schema  $R$  is **in first normal form** (1NF) if the domains of all attributes of  $R$  are atomic.

$R \in 1NF$



# Normal Forms



- A relation schema  $R$  is **in first normal form** (1NF) if the domains of all attributes of  $R$  are **atomic**.

学号	姓名	院系	系主任	课程号	得分
sno	sname	dept	dleader	cno	score
<u>S1</u>	张翠山	武当派	张三丰	<u>C1</u>	90
<u>S1</u>	张翠山	武当派	张三丰	<u>C2</u>	80
<u>S2</u>	谢逊	明教	张无忌	<u>C3</u>	100
<u>S2</u>	谢逊	明教	张无忌	<u>C3</u>	90

# Normal Forms



学号	姓名	院系	系主任	课程号	得分
sno	sname	dept	dleader	cno	score
<u>S1</u>	张翠山	武当派	张三丰		
<u>S2</u>	张无忌	武当派	张三丰	<u>C2</u>	

- 张翠山刚刚加入武当派，还没来得及选一个武功秘籍去修炼，就无法插入数据。
- 张无忌选了C2，但是不想学了，打算把这本秘籍还回去，本人信息也得删掉。



# Boyce–Codd Normal Form



## ボイスコード正規形

- **Boyce–Codd normal form (BCNF)**, eliminates all redundancy that can be discovered based on functional dependencies
  - BCNF消除所有基于函数依赖能够发现的冗余
- 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 a trivial functional dependency (that is,  $\beta \subseteq \alpha$ ).
  - $\alpha$  is a superkey for schema  $R$ .

# Boyce–Codd Normal Form



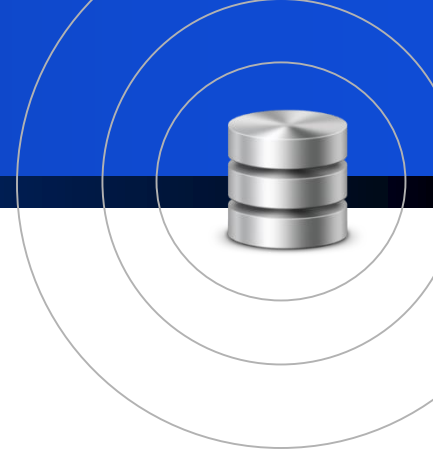
- A database design is in BCNF if each member of the set of relation schemas that constitutes the design is in BCNF.
- *EXAMPLE*
  - *inst\_dept* (*ID*, *name*, *salary*, *dept\_name*, *building*, *budget*)
  - Is it in BCNF?
    - —NO
    - $dept\_name \rightarrow budget$
  - *instructor* (*ID*, *name*, *dept\_name*, *salary*)
  - *department* (*dept\_name*, *building*, *budget*)

# Boyce–Codd Normal Form



- We now state a general rule for decomposing(分解) that are not in BCNF.
  - Let  $R$  be a schema that is not in BCNF.
  - Then there is at least one nontrivial functional dependency  $\alpha \rightarrow \beta$  such that  $\alpha$  is not a superkey for  $R$ .
  - We replace  $R$  in our design with two schemas:
    - $(\alpha \cup \beta)$
    - $(R - (\beta - \alpha))$

# Boyce–Codd Normal Form

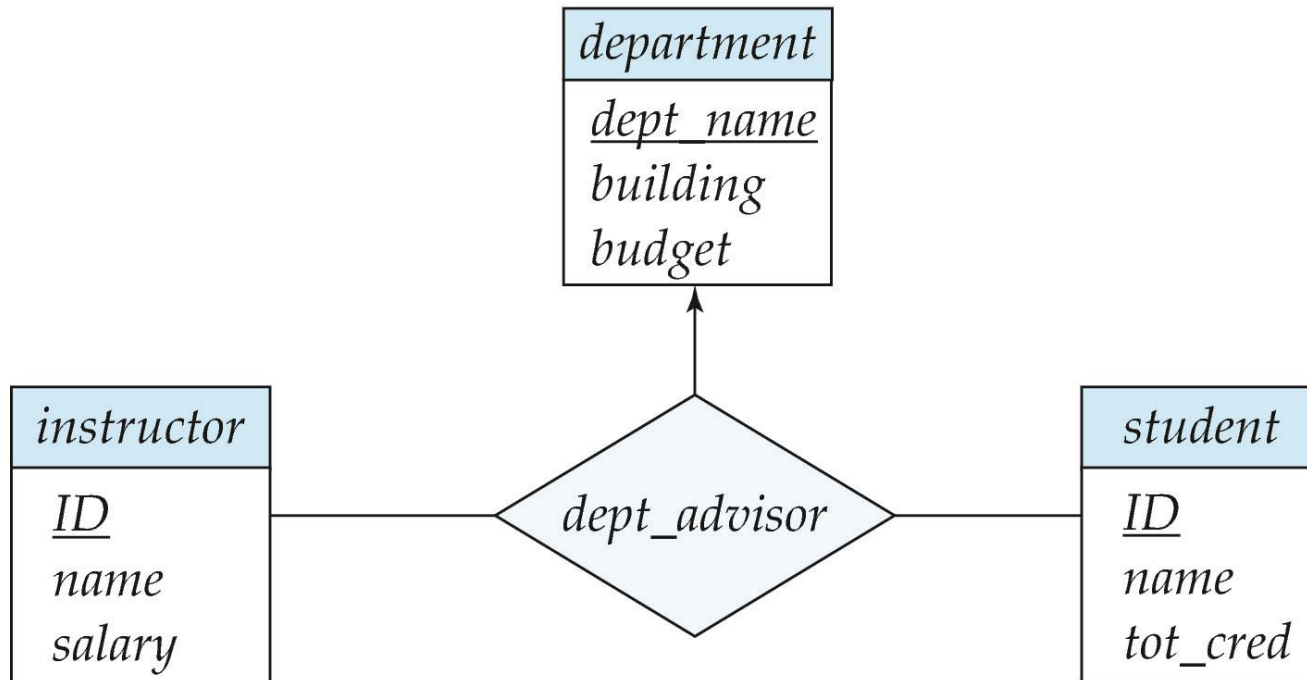


- In the case of *inst\_dept* above
  - $\alpha = dept\_name$
  - $\beta = \{building, budget\}$
- then *inst\_dept* is replaced by
  - $(\alpha \cup \beta) = (dept\_name, building, budget)$
  - $(R - (\beta - \alpha)) = (ID, name, dept\_name, salary)$

# BCNF and Dependency Preservation



- In some cases, decomposition(分解) into BCNF can prevent efficient testing of certain functional dependencies.





# BCNF and Dependency Preservation



- *dept\_advisor* (*s\_ID*, *i\_ID*, *dept\_name*)
  - “an instructor can act as advisor for only a single department.”
  - “a student may have more than one advisor, but at most one corresponding to a given department”.

$i\_ID \rightarrow dept\_name$

$s\_ID, dept\_name \rightarrow i\_ID$

- We see that *dept\_advisor* is not in BCNF because *i\_ID* is not a superkey.

# BCNF and Dependency Preservation



- We see that *dept\_advisor* is not in BCNF because *i\_ID* is not a superkey. Following our rule for BCNF decomposition, we get:
  - (*s\_ID*, *i\_ID*)
  - (*i\_ID*, *dept\_name*)

# BCNF and Dependency Preservation



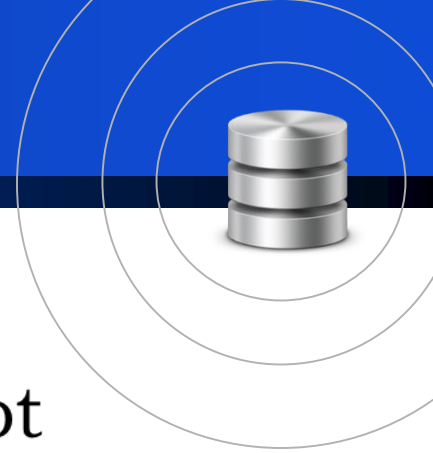
- However, in our BCNF design, there is no schema that includes all the attributes appearing in the functional dependency ***s\_ID, dept\_name → i\_ID***.
- We say our design is **not dependency preserving**(不是保持依赖的).

# Third Normal Form



- A relation schema  $R$  is in **third normal form** 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 a trivial functional dependency.
  - $\alpha$  is a superkey for  $R$ .
  - **Each attribute  $A$  in  $\beta - \alpha$  is contained in a candidate key for  $R$ .**

# Third Normal Form



- The definition of 3NF allows certain functional dependencies that are not allowed in BCNF.
- A dependency  $\alpha \rightarrow \beta$  that satisfies only the third alternative of the 3NF definition is not allowed in BCNF, but is allowed in 3NF.

# Third Normal Form

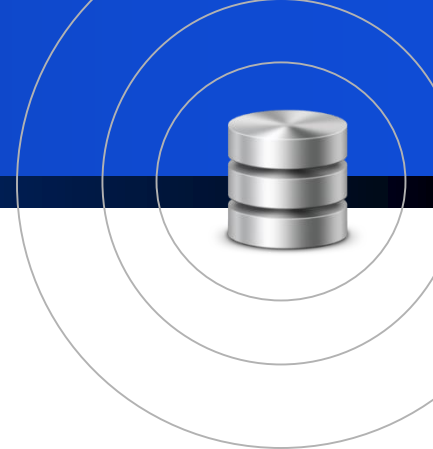


- Now, let us again consider the *dept\_advisor* relationship set, which has the following functional dependencies:
  - $i\_ID \rightarrow dept\ name$
  - $s\_ID, dept\_name \rightarrow i\ ID$
- $\alpha = i\ ID$ ,  $\beta = dept\_name$ , and  $\beta - \alpha = dept\_name$ .
  - ***dept\_advisor* is in 3NF.**

# 例题

- R (科室, 医生, 患者)

患者, 科室  $\rightarrow$  医生    医生  $\rightarrow$  科室  
是否满足 3NF?



# Review



- 无损分解/有损分解-不損失分解/損失分解
- 函数依赖理论- 関数従属性
- 原子域和第一范式-正規化
- BCNF&3NF-ボイスーコード正規形&第3正規化



# Closure of Attribute Sets



- We say that an attribute  $B$  is **functionally determined**(函数确定) by  $\alpha$  if  $\alpha \rightarrow B$ .
- To test whether a set  $\alpha$  is a superkey, we must devise an algorithm for computing the set of attributes functionally determined by  $\alpha$ .
- One way of doing this is to compute  $F^+$ , take all functional dependencies with  $\alpha$  as the left-hand side, and take the union of the right-hand sides of all such dependencies.

# Closure of Attribute Sets



- Let  $\alpha$  be a set of attributes.
- We call the set of all attributes functionally determined by  $\alpha$  under a set  $F$  of functional dependencies the **closure**(闭包) of  $\alpha$  under  $F$ ;
- We denote it by  $\alpha^+$ .

$r(A, B, C, G, H, I)$

$A \rightarrow B$

$A \rightarrow C$

$CG \rightarrow H$

$CG \rightarrow I$

$B \rightarrow H$

# BCNF Decomposition



- Testing for BCNF
  - To check if a nontrivial(非平凡) dependency  $\alpha \rightarrow \beta$  causes a violation of BCNF, compute  $\alpha^+$  (the attribute closure of  $\alpha$ ), and verify that it includes all attributes of  $R$ ; that is, it is a superkey of  $R$ .
  - To check if a relation schema  $R$  is in BCNF, it suffices to check only the dependencies in the given set  $F$  for violation of BCNF, rather than check all dependencies in  $F^+$ .

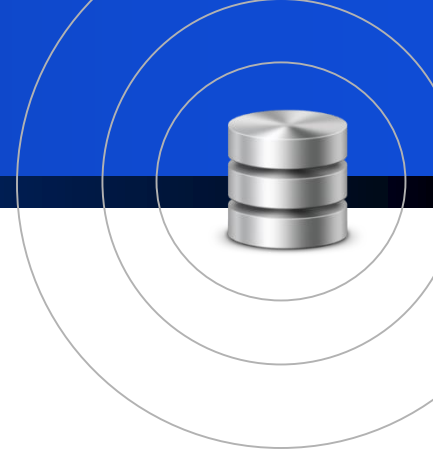
# Lossless Decomposition



- We say that the decomposition is a **lossless decomposition**(无损分解) if there is no loss of information by replacing  $r(R)$  with two relation schemas  $r_1(R_1)$  and  $r_2(R_2)$ .

select \*  
from (select  $R_1$  from  $r$ )  
natural join  
(select  $R_2$  from  $r$ )

$$\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) = r$$



- $R(A, B, C, D)$

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

候选码是? 是否满足BCNF?



# Thanks