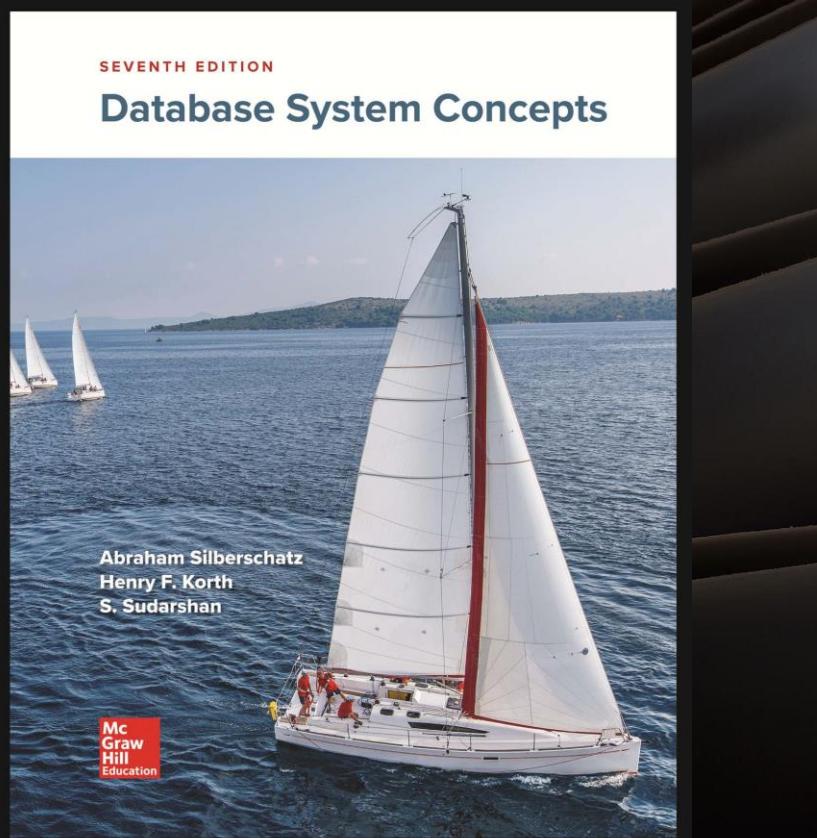


# IF2240 – Basis Data Relational Database Design (part 2)



# Sumber

---

- Silberschatz, Korth, Sudarshan: "Database System Concepts", 7<sup>th</sup> Edition
  - Chapter 7: Relational Database Design

# Capaian

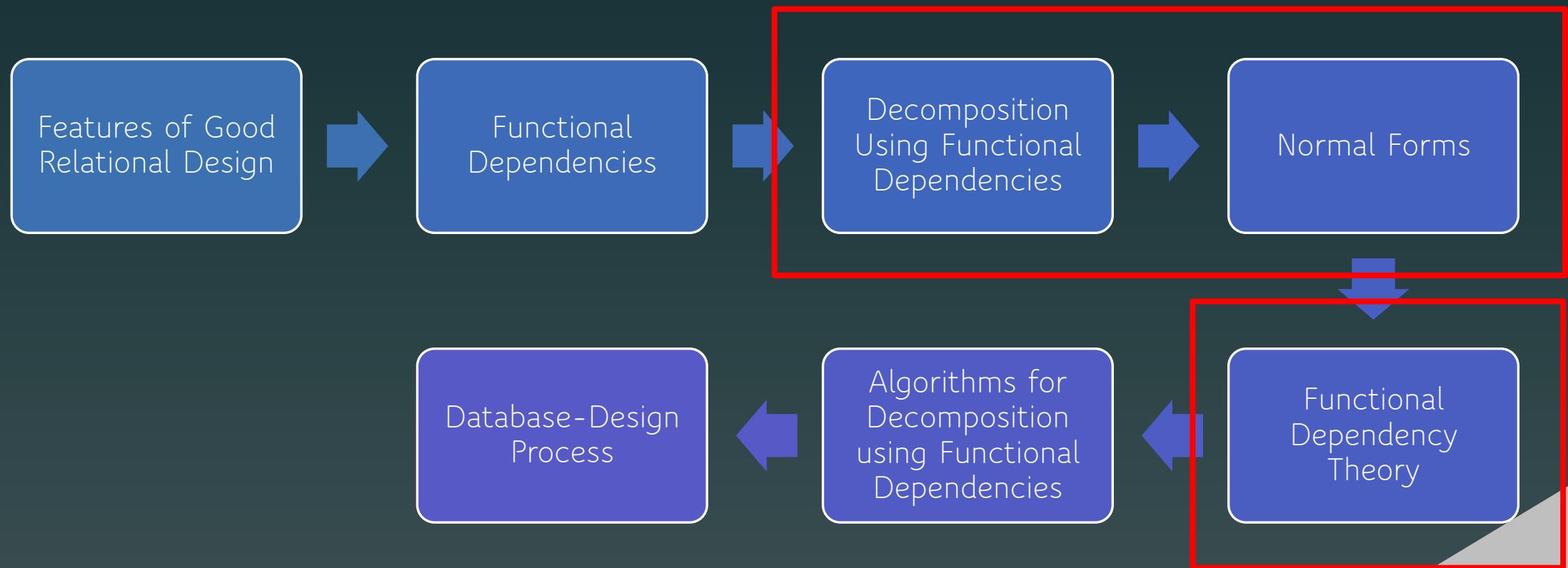
---

- Mahasiswa dapat menghasilkan desain basis data relasional yang baik berdasarkan prinsip-prinsip yang diberikan



# Outline

---



# Lossless Decomposition

---

For the case of  $R = (R_1, R_2)$

- $r = \prod_{R1} (r) \bowtie \prod_{R2} (r)$

A decomposition of  $R$  into  $R_1$  and  $R_2$  is lossless decomposition if at least one of the following dependencies is in  $F^+$ :

- $R_1 \cap R_2 \rightarrow R_1$
- $R_1 \cap R_2 \rightarrow R_2$

# Lossless Decomposition

For the case of  $R = (R_1, R_2)$

- $r = \prod_{R1}(r) \bowtie \prod_{R2}(r)$

A decomposition of  $R$  into  $R_1$  and  $R_2$  is lossless decomposition if at least one of the following dependencies is in  $F^+$ :

- $R_1 \cap R_2 \rightarrow R_1$
- $R_1 \cap R_2 \rightarrow R_2$

- $R = (A, B, C)$   
 $F = \{A \rightarrow B, B \rightarrow C\}$
- $R_1 = (A, B), R_2 = (B, C)$ 
  - Lossless decomposition:

$$R_1 \cap R_2 = \{B\} \text{ and } B \rightarrow BC$$

- $R_1 = (A, B), R_2 = (A, C)$ 
  - Lossless decomposition:

$$R_1 \cap R_2 = \{A\} \text{ and } A \rightarrow AB$$



# Dependency Preservation

---

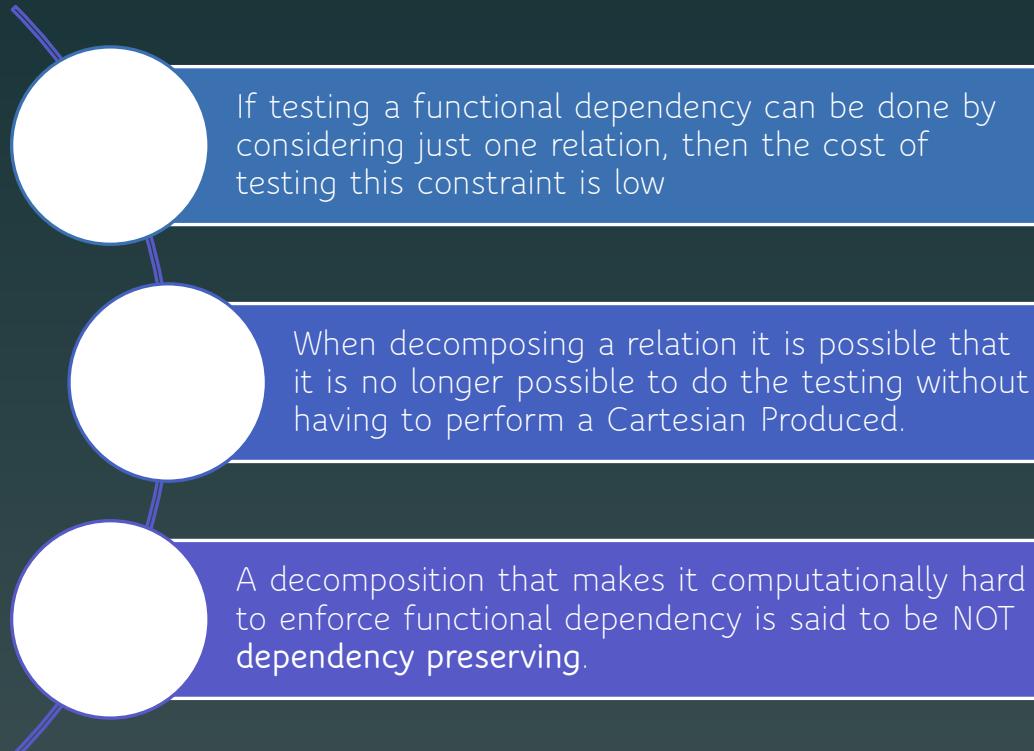
If testing a functional dependency can be done by considering just one relation, then the cost of testing this constraint is low

When decomposing a relation it is possible that it is no longer possible to do the testing without having to perform a Cartesian Product.

A decomposition that makes it computationally hard to enforce functional dependency is said to be **NOT dependency preserving**.

# Dependency Preservation

---



- Consider a schema:

$$\text{dept\_advisor}(s\_ID, i\_ID, \text{dept\_name})$$

- With functional dependencies:

$$i\_ID \rightarrow \text{dept\_name}$$
$$s\_ID, \text{dept\_name} \rightarrow i\_ID$$

- In the above design we are forced to repeat the department name once for each time an instructor participates in a *dept\_advisor* relationship.

# Dependency Preservation

---

- Consider a schema:

$dept\_advisor(s\_ID, i\_ID, dept\_name)$

- With functional dependencies:

$i\_ID \rightarrow dept\_name$

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

- In the above design we are forced to repeat the department name once for each time an instructor participates in a  $dept\_advisor$  relationship.

s_ID	i_ID	dept_name
Clark	Srinivasan	Comp. Sci.
Clark	Wu	Info. Syst.
Bruce	Wu	Info. Syst
Diana	Katz	Comp. Sci
Peter	Srinivasan	Comp. Sci
Peter	Kim	Elec. Eng..

To fix this, we need to decompose  $dept\_advisor$ . Any decomposition will not include all the attributes in  $s\_ID \ dept\_name \rightarrow i\_ID$ . Thus, the composition NOT be dependency preserving.

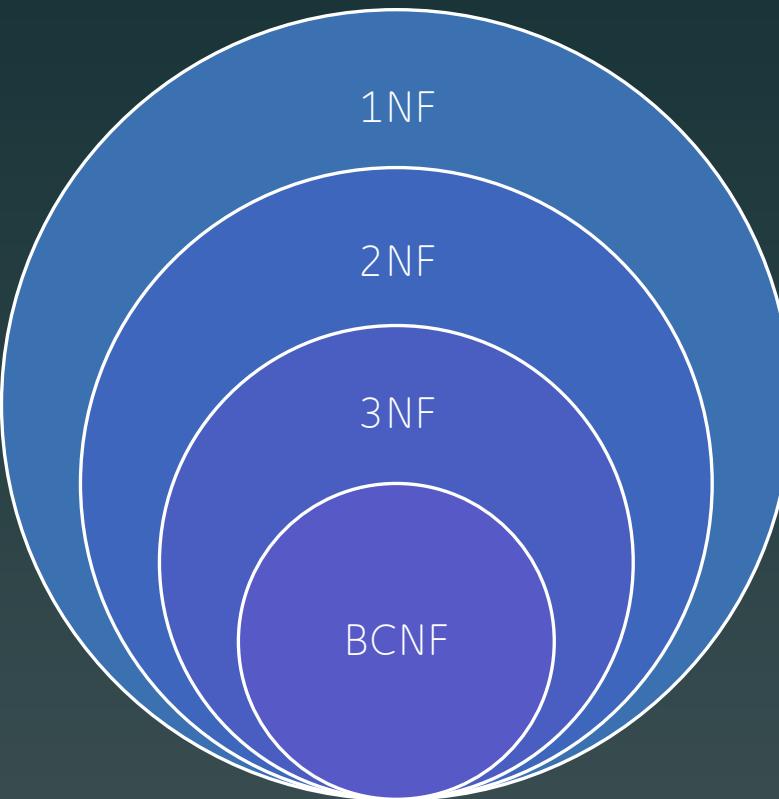


# Normal Forms

---

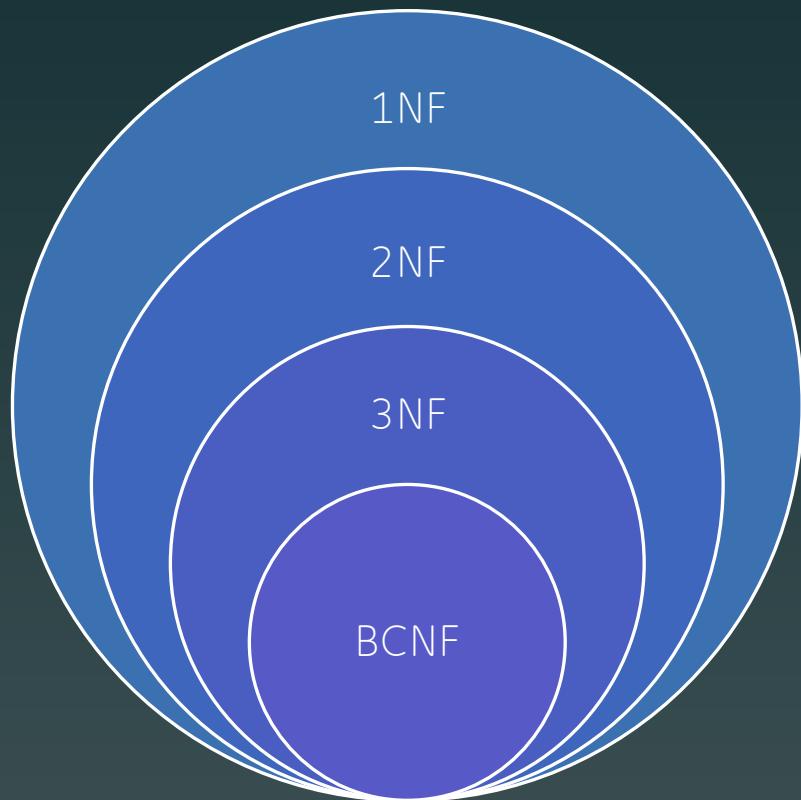
# Normal Forms

---



# First Normal Forms

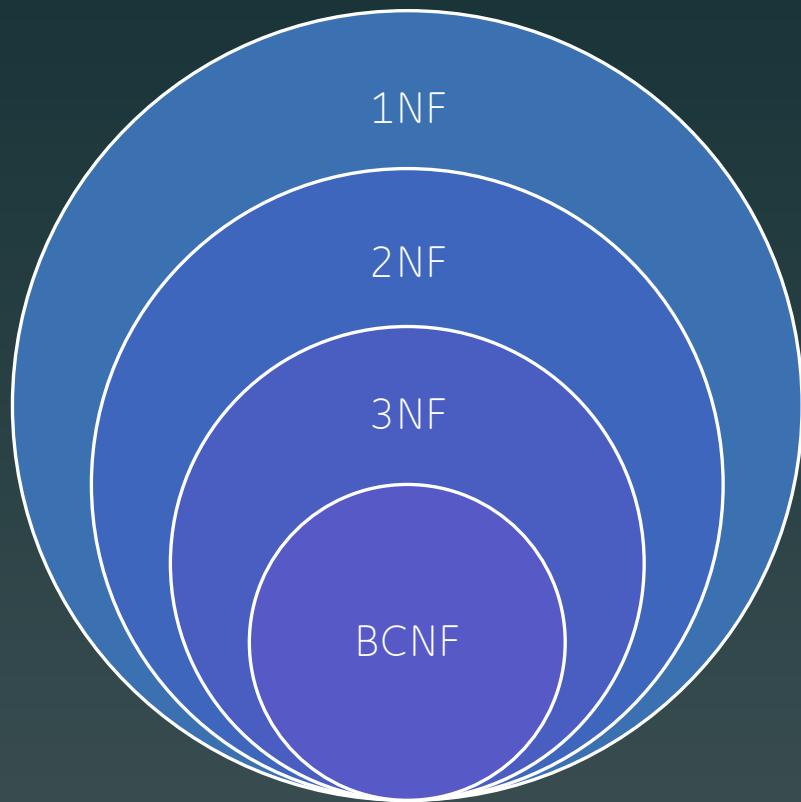
---



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

# Second Normal Forms

---



A relation schema  $R$  is in **second normal form (2NF)** if each attribute  $A$  in  $R$  meets one of the following criteria:

- It appears in a candidate key.
- It is not partially dependent on a candidate key.

# 2NF Example

---

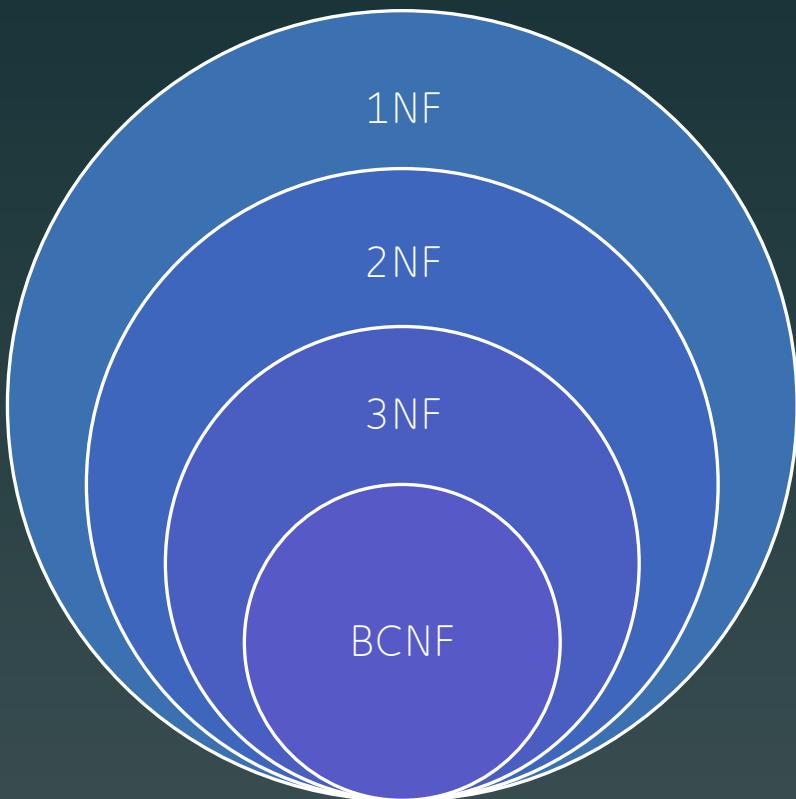
- StudentCourse(sID, sName, cID, cName, Grade)
- $F = \{sID \rightarrow sName,$   
 $cID \rightarrow cName,$   
 $sID, cID \rightarrow Grade\}$

sID	sName	cID	cName	Grade
18219500	Clark	IF2140	Database Modeling	A
18219500	Clark	IF2240	Database	B
13519300	Bruce	IF3140	Database Technologies	B
13519300	Bruce	II2250	Database Systems	A
13518550	Diana	IF2240	Database	B

StudentCourse is not 2NF

# Boyce-Codd Normal Forms

---



- 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  $R$

# Boyce-Codd Normal Forms

---

- 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 :

*in\_dep (ID, name, salary, dept\_name, building, budget )*

is *not* in BCNF because :

- $dept\_name \rightarrow building\ budget$  holds on *in\_dep*

- Decompose *in\_dep* into *instructor* and *department*

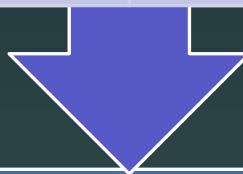
- *instructor (ID, name, salary, dept\_name)* is in BCNF
- *department (dept\_name, building, budget)* is in BCNF

# Decomposing a Schema into BCNF

Let:

$R$  be a schema  $R$  that is not in BCNF.

$\alpha \rightarrow \beta$  be the FD that causes a violation.



We decompose  $R$  into:

$$(\alpha \cup \beta)$$

$$(R - (\beta - \alpha))$$

# Decomposing a Schema into BCNF

Let:

$R$  be a schema  $R$  that is not in BCNF.

$\alpha \rightarrow \beta$  be the FD that causes a violation.

We decompose  $R$  into:

$(\alpha \cup \beta)$

$(R - (\beta - \alpha))$

- Example schema :

$in\_dep (ID, name, salary, dept\_name, building, budget)$

is *not* in BCNF because :

- $dept\_name \rightarrow building$  holds on  $in\_dep$

$\alpha$

$\beta$

$(R - (\beta - \alpha))$

- Decompose  $in\_dep$  into  $instructor$  and  $department$

- $instructor (ID, name, salary, dept\_name)$  is in BCNF
- $department (dept\_name, building, budget)$  is in BCNF

$\alpha \cup \beta$

# Decomposing a Schema into BCNF

$$R = (A, B, C)$$

$$F = \{A \rightarrow B, B \rightarrow C\}$$

$$\begin{aligned} R_1 &= (A, B) \\ R_2 &= (B, C) \end{aligned}$$

Lossless-join  
decomposition

Dependency  
preserving

$$\begin{aligned} R_1 \cap R_2 &= \{B\} \\ \text{and } B &\rightarrow R_2 \end{aligned}$$

# Decomposing a Schema into BCNF

$$R = (A, B, C)$$

$$F = \{A \rightarrow B, B \rightarrow C\}$$

$$R_1 = (A, B)$$

$$R_2 = (B, C)$$

Lossless-join decomposition

Dependency preserving

$$R_1 = (A, B)$$

$$R_2 = (A, C)$$

Lossless-join  
decomposition:

Not  
dependency  
preserving

$$R_1 \cap R_2 = \{A\}$$

and  $A \rightarrow AB$

checking  $B \rightarrow C$   
needs  $R_1 \bowtie R_2$

# BCNF and Dependency Preservation

It is not always possible to achieve both BCNF and dependency preservation

$dept\_advisor(s\_ID, i\_ID, department\_name)$ ,  
 $F = \{ i\_ID \rightarrow dept\_name, s\_ID \text{ dept\_name} \rightarrow i\_ID \}$



$dept\_advisor$  is not in BCNF

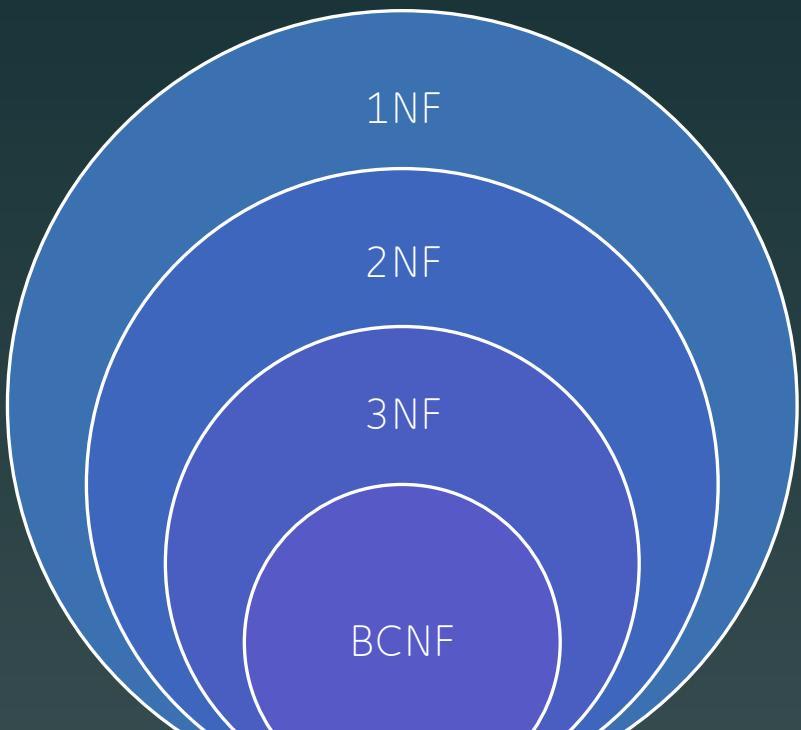
$i\_ID$  is not a superkey.



Any decomposition will not include all the attributes in  $s\_ID \text{ dept\_name} \rightarrow i\_ID$

Thus, the decomposition is NOT dependency preserving

# Third Normal Forms



Third condition is a minimal relaxation of BCNF to ensure dependency preservation

A relation schema  $R$  is in **third normal form** (**3NF**) if for all:

$$\alpha \rightarrow \beta \text{ in } F^+$$

at least one of the following holds:

- $\alpha \rightarrow \beta$  is trivial (i.e.,  $\beta \in \alpha$ )
- $\alpha$  is a superkey for  $R$
- Each attribute  $A$  in  $\beta - \alpha$  is contained in a candidate key for  $R$ .

# Third Normal Forms

---

A relation schema  $R$  is in **third normal form (3NF)** if for all:

$$\alpha \rightarrow \beta \text{ in } F^+$$

at least one of the following holds:

- $\alpha \rightarrow \beta$  is trivial (i.e.,  $\beta \in \alpha$ )
- $\alpha$  is a superkey for  $R$
- Each attribute  $A$  in  $\beta - \alpha$  is contained in a candidate key for  $R$ .

*dept\_advisor(s\_ID, i\_ID, dept\_name)*  
 $F = \{ i\_ID \rightarrow dept\_name,$   
 $s\_ID\ dept\_name \rightarrow i\_ID \}$

- Two candidate keys =  $\{s\_ID, dept\_name\}, \{s\_ID, i\_ID\}$
- *dept\_advisor* is not in BCNF
- *dept\_advisor*, however, is in 3NF
  - $s\_ID, dept\_name$  is a superkey
  - $i\_ID$  is NOT a superkey, but *dept\_name* is contained in a candidate key

# Redundancy in 3NF

---

Consider schema  $\text{dept\_advisor}(s\_ID, i\_ID, \text{dept\_name})$  with functional dependencies  $F = \{ i\_ID \rightarrow \text{dept\_name}, s\_ID \text{ dept\_name} \rightarrow i\_ID \}$  which is in 3NF.

s_ID	i_ID	dept_name
Clark	Srinivasan	Comp. Sci.
Clark	Wu	Info. Syst.
Bruce	Wu	Info. Syst
Diana	Katz	Comp. Sci
Peter	Srinivasan	Comp. Sci
Peter	Kim	Elec. Eng..

What is wrong with the table?

- Repetition of information
- Need to use null values (e.g., to represent the relationship *Einstein, Physics* where there is no corresponding value for  $s\_ID$ )

# Comparison of BCNF and 3NF

## Advantages to 3NF over BCNF.

- It is always possible to obtain a 3NF design without sacrificing losslessness or dependency preservation.

## Disadvantages to 3NF.

- We may have to use null values to represent some of the possible meaningful relationships among data items.
- There is the problem of repetition of information.

# Goals of Normalization

---

Let  $R$  be a relation scheme with a set  $F$  of functional dependencies.

Decide whether a relation scheme  $R$  is in “good” form.

In the case that  $R$  is not in “good” form, need to decompose it into  $\{R_1, R_2, \dots, R_n\}$  such that:

Each relation scheme is in good form

The decomposition is a lossless decomposition

Preferably, the decomposition should be dependency preserving.

# How good is BCNF?

---

- There are database schemas in BCNF that do not seem to be sufficiently normalized
- Consider a relation

*inst\_info (ID, child\_name, phone)*

- where an instructor may have more than one phone and can have multiple children
- Instance of *inst\_info*

ID	child_name	phone
99999	David	512-555-1234
99999	David	512-555-4321
99999	William	512-555-1234
99999	William	512-555-4321

There are no non-trivial functional dependencies and therefore the relation is in BCNF

# How good is BCNF?

---

- There are database schemas in BCNF that do not seem to be sufficiently normalized
- Consider a relation

*inst\_info (ID, child\_name, phone)*

- where an instructor may have more than one phone and can have multiple children

ID	child_name	phone
99999	David	512-555-1234
99999	David	512-555-4321
99999	William	512-555-1234
99999	William	512-555-4321

- Insertion anomalies –  
i.e., if we add a phone 981-992-3443 to 99999,  
we need to add two tuples
  - (99999, David, 981-992-3443)
  - (99999, William, 981-992-3443)

# How good is BCNF?

---

- There are database schemas in BCNF that do not seem to be sufficiently normalized
- Consider a relation

*inst\_info (ID, child\_name, phone)*

- where an instructor may have more than one phone and can have multiple children

ID	child_name	phone
99999	David	512-555-1234
99999	David	512-555-4321
99999	William	512-555-1234
99999	William	512-555-4321

- It is better to decompose *inst\_info* into:

- *inst\_child*:

ID	child_name
99999	David
99999	William

- *inst\_phone*:

ID	phone
99999	512-555-1234
99999	512-555-4321

This suggests the need for higher normal forms, such as Fourth Normal Form (4NF)

# Functional-Dependency Theory

---

# Functional-Dependency Theory

---

## Roadmap

The formal theory that tells us which functional dependencies are implied logically by a given set of functional dependencies.

Algorithms to generate lossless decompositions into BCNF and 3NF

Algorithms to test if a decomposition is dependency-preserving

# Closure of a Set of Functional Dependencies

We can compute  $F^+$ , the closure of  $F$ , by repeatedly applying Armstrong's Axioms:

Reflexive rule:

- if  $\beta \subseteq \alpha$ , then  $\alpha \rightarrow \beta$

Augmentation rule:

- if  $\alpha \rightarrow \beta$ , then  $\gamma \alpha \rightarrow \gamma \beta$

Transitivity rule:

- if  $\alpha \rightarrow \beta$ , and  $\beta \rightarrow \gamma$ , then  $\alpha \rightarrow \gamma$

# Closure of a Set of Functional Dependencies

We can compute  $F^+$ , the closure of  $F$ , by repeatedly applying Armstrong's Axioms:

Reflexive rule:

- if  $\beta \subseteq \alpha$ , then  $\alpha \rightarrow \beta$

Augmentation rule:

- if  $\alpha \rightarrow \beta$ , then  $\gamma \alpha \rightarrow \gamma \beta$

Transitivity rule:

- if  $\alpha \rightarrow \beta$ , and  $\beta \rightarrow \gamma$ ,  
then  $\alpha \rightarrow \gamma$

Sound

Complete

# Closure of a Set of Functional Dependencies

We can compute  $F^+$ , the closure of  $F$ , by repeatedly applying **Armstrong's Axioms**:

Reflexive rule:

- if  $\beta \subseteq \alpha$ , then  $\alpha \rightarrow \beta$

Augmentation rule:

- if  $\alpha \rightarrow \beta$ ,  
then  $\gamma \alpha \rightarrow \gamma \beta$

Transitivity rule:

- if  $\alpha \rightarrow \beta$ , and  $\beta \rightarrow \gamma$ ,  
then  $\alpha \rightarrow \gamma$

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

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

Some members of  $F^+$

- $A \rightarrow H$ 
  - by transitivity from  $A \rightarrow B$  and  $B \rightarrow H$
- $AG \rightarrow I$ 
  - by augmenting  $A \rightarrow C$  with  $G$ , to get  $AG \rightarrow CG$
  - then transitivity from  $AG \rightarrow CG$  and  $CG \rightarrow I$

# Closure of a Set of Functional Dependencies

We can compute  $F^+$ , the closure of  $F$ , by repeatedly applying **Armstrong's Axioms**:

Reflexive rule:

- if  $\beta \subseteq \alpha$ , then  $\alpha \rightarrow \beta$

Augmentation rule:

- if  $\alpha \rightarrow \beta$ ,  
then  $\gamma \alpha \rightarrow \gamma \beta$

Transitivity rule:

- if  $\alpha \rightarrow \beta$ , and  $\beta \rightarrow \gamma$ ,  
then  $\alpha \rightarrow \gamma$

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

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

Some members of  $F^+$

- $CG \rightarrow HI$ 
  - by augmenting  $CG \rightarrow I$  with  $CG$  to infer  $CG \rightarrow CGI$ ,
  - then augmenting of  $CG \rightarrow H$  with  $I$  to infer  $CGI \rightarrow HI$ ,
  - then transitivity  $CG \rightarrow CGI$  and  $CGI \rightarrow HI$

# Closure of Functional Dependencies

Additional rules:

Union rule:

- If  $\alpha \rightarrow \beta$  holds and  $\alpha \rightarrow \gamma$  holds,  
then  $\alpha \rightarrow \beta\gamma$  holds.

Decomposition rule:

- If  $\alpha \rightarrow \beta\gamma$  holds,  
then  $\alpha \rightarrow \beta$  holds and  $\alpha \rightarrow \gamma$  holds.

Pseudotransitivity rule:

- If  $\alpha \rightarrow \beta$  holds and  $\beta \rightarrow \delta$  holds,  
then  $\alpha\beta \rightarrow \delta$  holds.

# Procedure for Computing $F^+$

---

- To compute the closure of a set of functional dependencies  $F$ :

$F^+ = F$

repeat

    for each functional dependency  $f$  in  $F^+$

        apply reflexivity and augmentation rules on  $f$

        add the resulting functional dependencies to  $F^+$

    for each pair of functional dependencies  $f_1$  and  $f_2$  in  $F^+$

        if  $f_1$  and  $f_2$  can be combined using transitivity

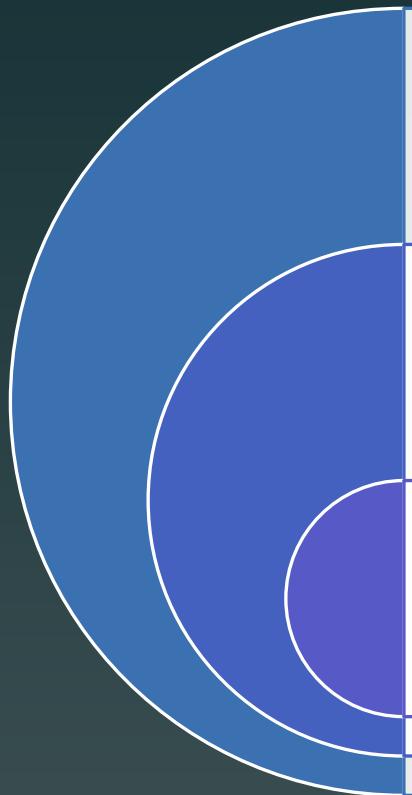
            then add the resulting functional dependency to  $F^+$

until  $F^+$  does not change any further

- NOTE: We shall see an alternative procedure for this task later

# Closure of Attribute Sets

---

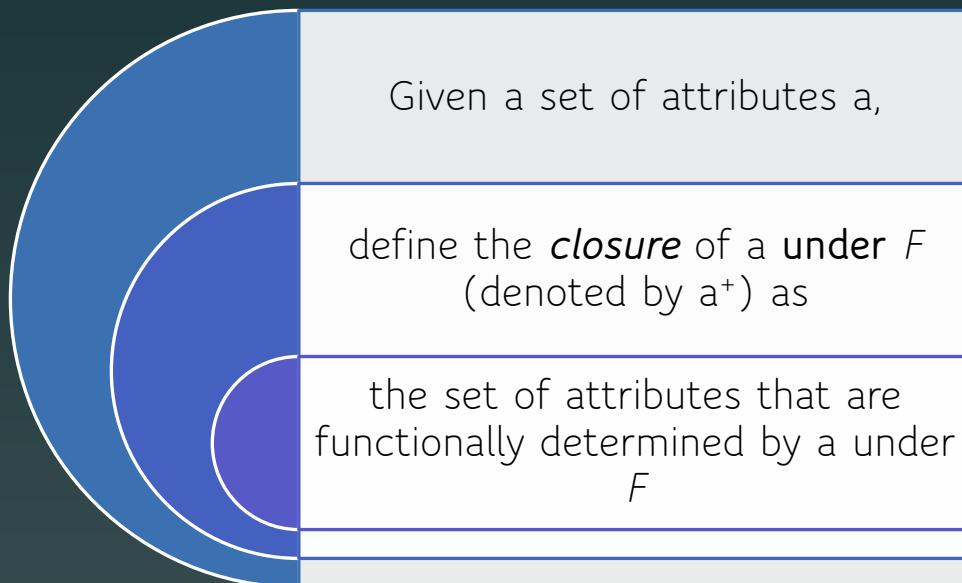


Given a set of attributes  $a$ ,

define the *closure* of  $a$  under  $F$   
(denoted by  $a^+$ ) as

the set of attributes that are  
functionally determined by  $a$  under  $F$

# Closure of Attribute Sets



Algorithm to compute  $\alpha^+$ , the closure of  $\alpha$  under  $F$

```
result :=  $\alpha$ ;
while (changes to result) do
    for each  $\beta \rightarrow \gamma$  in  $F$  do
        begin
            if  $\beta \subseteq result$  then
                result := result  $\cup$   $\gamma$ 
        end
```

# Closure of Attribute Sets

---

Algorithm to compute  $\alpha^+$ , the closure of  $\alpha$  under  $F$

```
result :=  $\alpha$ ;
while (changes to  $result$ ) do
    for each  $\beta \rightarrow \gamma$  in  $F$  do
        begin
            if  $\beta \subseteq result$  then
                 $result := result \cup \gamma$ 
        end
```

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

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

$$(AG)^+?$$

$$result = AG$$

1.  $result = ABCG$       ( $A \rightarrow C, A \rightarrow B$  and  $A \subseteq AG$ )
  2.  $result = ABCGH$       ( $CG \rightarrow H$  and  $CG \subseteq AGBC$ )
- $$result = ABCGHI \quad (CG \rightarrow I \text{ and } CG \subseteq AGBCH)$$

# Closure of Attribute Sets

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

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

$(AG)^+?$

$result = AG$

1.  $result = ABCG$        $(A \rightarrow C, A \rightarrow B \text{ and } A \subseteq AB)$

2.  $result = ABCGH$        $(CG \rightarrow H \text{ and } CG \subseteq AGBC)$

$result = ABCGHI$        $(CG \rightarrow I \text{ and } CG \subseteq AGBCH)$

Is AG a candidate key?

1. Is AG a super key?

1. Does  $AG \rightarrow R? == \text{Is } R \subseteq (AG)^+$

2. Is any subset of AG a superkey?

1. Does  $A \rightarrow R? == \text{Is } R \subseteq (A)^+$

2. Does  $G \rightarrow R? == \text{Is } R \subseteq (G)^+$

3. In general: check for each subset  
of size  $n-1$

∴ AG is a candidate key of R

# Uses of Attribute Closure

Testing for superkey:

compute  $\alpha^+$ ,

check if  $\alpha^+$  contains all attributes of  $R$

Testing functional dependencies

To check if  $\alpha \rightarrow \beta$  holds,  
just check if  $\beta \subseteq \alpha^+$ .

Computing closure of  $F$

for each  $\gamma \subseteq R$ ,  
find the closure  $\gamma^+$ ,

for each  $S \subseteq \gamma^+$ ,  
output a functional dependency  $\gamma \rightarrow S$ .

# Canonical Cover

NEXT MEETING...