

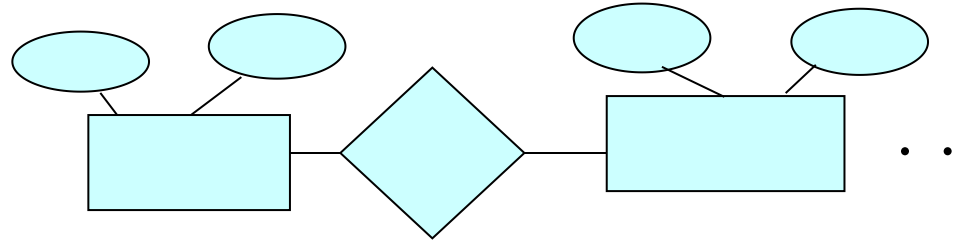
## 8. Normalization

# Main Issues:

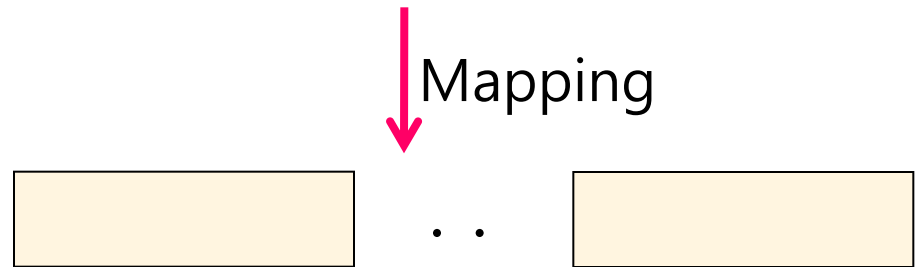
- What problems are caused by **redundancy**?
- What is criteria for **good** relation?
- What is **functional dependency**?
  - Concepts of **FDs**
  - Inference Rules for FDs
  - Closure Property
  - Relationship : FD and Key
- What is **normalization**?
  - First Normal Form (1**NF**)
  - Second Normal Form (2**NF**)
  - Third Normal Form (3**NF**)
  - Boyce Codd Normal Form (BC**NF**)

# Relational Design : Normalization

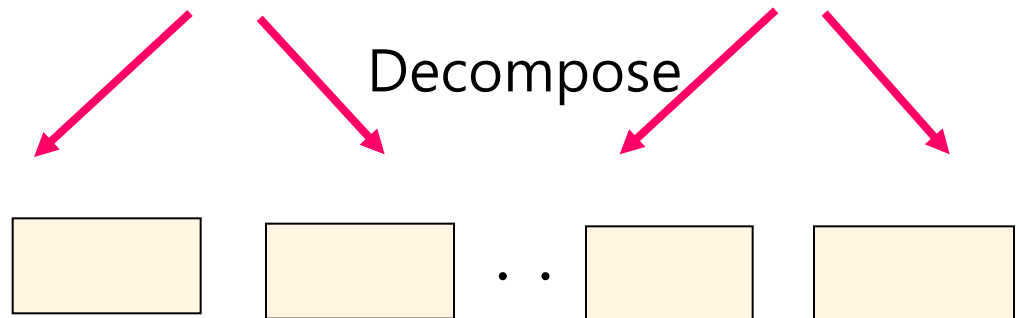
ER Schema:



Relational Schema:



Normalized relations:  
Eliminates redundancy  
and anomalies



# Problems : Bad Relational Schema





- For a “badly” designed relation, the following problems occur:
  - **Redundancy**: The same information is repeated.
  - **Insert Anomaly**: We can not insert new information.
  - **Delete Anomaly**: We may lose unwanted information.
  - **Update Anomaly**: We need to change it in many places.

# "Bad" Relation: Example

- 다음 relation의 문제점은?

Bad      단, PK = {SSN}

<u>SSN</u>	name	age	DNO	dname	budget
11111	abe	45	d1	network	500K
22222	eve	28	d3	big data	900K
33333	jim	37	d2	security	700K
44444	bob	35	d2	security	700K
55555	paul	25	d3	big data	900K
66666	dick	35	d2	security	700K
77777	eva	60	d3	big data	900K
88888	jane	40	d2	security	700K

- Redundancy? 
- Insert Anomaly? 
- Delete Anomaly? 
- Update Anomaly? 

## "Bad" Relation : Problems

- **Redundancy:** Each department information is repeated as the number of employees work for it.
- **Insert Anomaly:** Suppose that we insert new department; Then, we can not insert that department if some employee is not assigned to it. Why?
- **Delete Anomaly:** Suppose that we delete some department; Then, we may lose all employee information working for it. What if we delete last employee working for department?
- **Update Anomaly:** Suppose that we change the name of some department; Then, it may cause this update to be made for every employee working for it.

# Solution : Decomposition

- We decompose Bad relation as follows:

Good1    PK = {SSN}

<u>SSN</u>	name	age	DNO
11111	abe	45	d1
22222	eve	28	d3
33333	jim	37	d2
44444	bob	35	d2
55555	paul	25	d3
66666	dick	35	d2
77777	eva	60	d3
88888	jane	40	d2

Good2    PK = {DNO}

<u>DNO</u>	dname	budget
d1	network	500K
d2	security	700K
d3	big data	900K

- Are these two relations good?
- Can we recover original information exactly?

# Solution : Decomposition

- Those two relations are good because:
  - **No Redundancy** : Each department information occurs only once.
  - **No Insert Anomaly** : We can insert new department even though no employee is assigned to it.
  - **No Delete Anomaly** : Even if we delete some department, we still can keep all the employees working for it.
  - **No Update Anomaly** : We can change the name of department only once.



# Bad Relations : Example

- 다음 relation의 문제점은? 해결책은?

POOR			PK = {SSN, PNO}		
SSN	PNO	hours	ename	pname	plocation
11111	p1	45	abe	notebook	LA
11111	p2	28	abe	printer	NY
44444	p1	37	bob	notebook	LA
44444	p2	35	bob	printer	NY
44444	p3	25	bob	MP3	SF
55555	p1	35	ann	notebook	LA
77777	p1	23	eva	notebook	LA
77777	p2	40	eva	printer	NY

- Redundancy?
- Insert Anomaly?
- Delete Anomaly?
- Update Anomaly?

# Guidelines for Good Relations

- GUIDELINE 1:
  - Design a relational schema with **no redundancy**.
  - Design a relational schema with **no insertion, deletion, update anomalies**.
- GUIDELINE 2:
  - Design a relational schema to satisfy the **lossless join condition**.
  - **No spurious** tuples must be generated by doing a natural join of any relations.
- GUIDELINE 3:
  - Design a relational schema such that their tuples will have as **few NULL** values as possible.
  - Attributes that are NULL frequently could be placed in separate relations (with the primary key).

# Functional Dependency (FD)

- A Functional dependency (FD) is used to specify *formal measures* of the **"goodness"** of relational designs.
- FDs are **constraints** that are derived from the **meaning** and *inter-relationships* of the attributes.
- FDs are derived from the **real-world constraints** on the attributes.
- FDs and keys are used to define **normal forms** for relations.

# Functional Dependency (FD)

- A FD  $X \rightarrow Y$  in a relation R is defined as: For any two tuples **t1** and **t2** in R, if **t1[X] = t2[X]** holds, then **t1[Y] = t2[Y]** also holds where **X, Y** : a set of attributes in R.
- Whenever two tuples have the same value for **X**, they also must have the same value for **Y**.
- Value(s) of **X** uniquely (functionally) decides the value(s) of **Y**.
- If  $X \rightarrow Y$  holds in R, this does not say that  $Y \rightarrow X$  holds.
- If every tuple for X has distinct value(s), then X can decide any attribute(s) in R; Why?

# FD : Example

- Given a tuple and the value(s) of X, we can decide the corresponding value of Y .
- $R = (A, B, C, D, E)$   
FD : (1)  $A \rightarrow B$   
(2)  $C \rightarrow D$

R

A	B	C	D	E
a1	b1	c3	d2	e1
a1	?	c2	d1	e2
a2	b2	c4	d4	e3
a1	?	c1	d4	e4
a2	?	c4	?	e1

# FD : Example

●  $R = (A, B, C, D, E)$

FD : (1)  $A \rightarrow \{B, D\}$

(2)  $\{B, C\} \rightarrow E$

R

A	B	C	D	E
a1	b1	c1	d2	e1
a1	?	c2	?	e2
a2	b2	c4	d4	e3
a1	?	c3	?	e4
a3	b2	c4	d2	?

## FD : 실제 예 (1)

- FD는 한 relation에서 실세계의 제약조건을 반영. DB Designer는 이러한 제약조건을 FD로 변환하여 정의;

STUDENT(ID, club, prof#, course#, major)

- Each student joins only one club.  
ID  $\rightarrow$  club
- Each student has only one major.  
ID  $\rightarrow$  major
- Each professor teaches only one course.  
prof#  $\rightarrow$  course#

## FD : 실제 예 (1)

- FD는 relation 안의 모든 tuple들이 항상 만족해야 하는 조건.

$ID \rightarrow club$

$prof\# \rightarrow course\#$

$ID \rightarrow major$

ID	club	STUDENT prof#	course#	major
Bob	Ski	Kim	DB	Physics
Bob	Ski	Ted	C	Physics
Eve	Music	Kim	DB	
Business				
Bob	??	Cal	Java	??
Abe	Tennis	Ted	??	Math
Eve	??	Cal	??	??
Abe	??	Kim	??	??

- 만약 새로운 tuple <Eve, Dance, Kim, ...>을 insert 한다면?
- 만약 tuple <Bob, ..., DB, ...>에서 OS로 update 한다면?



## FD : 실제 예 (2)

Student(S#, C#, sname, age, cname, credit, grade, D#, dname, office)

- Each student has only one his/her name and age.

$$S\# \rightarrow \{sname, age\}$$

- Each course has only one its name and credit.

$$C\# \rightarrow \{cname, credit\}$$

- Each student belongs to only one department.

$$S\# \rightarrow D\#$$

- Each department has only one its name and office.

$$D\# \rightarrow \{dname, office\}$$

- For each student and his/her taken course, only one grade exists.

$$\{S\#, C\#\} \rightarrow grade$$

## FD : 실제 예 (2)

### Student

(S#, C#, sname, age, cname, credit, grade, D#, dname, office)									
S1	C1	bob	27	DB	3	A	D1	CS	O1
S1	C2	bob	27	OS	3	B	D1	CS	O1
S2	C1	Joe	25	DB	3	B	D1	CS	O1
S2	C3	Joe	25	Algebra	3	B	D2	Math	O2
S3	C1	tom	30	DB	3	A	D1	CS	O1
S3	C2	tom	30	OS	3	C	D1	CS	O1
S3	C3	tom	30	Algebra	3	B	D2	Math	O2
S4	C1	abe	24	DB	3	C	D1	CS	O1
S4	C2	abe	24	OS	3	A	D1	CS	O1
S4	C3	abe	24	Algebra	3	B	D2	Math	O2

- 1) 앞의 FD들을 모두 만족하는 tuple들의 예를 10 개 보여라.
- 2) 어떤 정보들이 여러 번 반복되는지 보여라.
- 3) Super Key를 찾아 보여라. Key는 무엇인가?

## FD : 실제 예 (3)

- 다음 EMPLOYEE relation에서 각 FD가 만족되는지 검증하라.

예 3 :

<u>SSN</u>	<u>name</u>	<u>phone</u>	<u>position</u>
11111	bob	1234	engineer
22222	abe	2345	sales
33333	bob	2345	sales
55555	eve	1234	lawyer

1) SSN  $\rightarrow$  name ?    answer :

2) position  $\rightarrow$  phone ?    answer :

3) phone  $\rightarrow$  position ?    answer :

4) {position, phone}  $\rightarrow$  phone ?    answer :

## FD : 실제 예 (4)

- 다음 PURCHASE relation에서 성립하는 FD들을 모두 나열하라.  
예 4 :

<u>CID</u>	<u>PID</u>	<u>Cname</u>	<u>Pname</u>	<u>Age</u>	<u>Price</u>
C4	P5	Bob	PC	35	\$500
C4	P7	Bob	Phone	35	\$300
C8	P5	Joe	PC	40	\$550
C7	P4	Tom	Printer	45	\$400
C8	P7	Joe	Phone	40	\$320

- $CID \rightarrow Cname, CID \rightarrow Age$
- $Cname \rightarrow Age$
- $PID \rightarrow Pname$
- $\{CID, PID\} \rightarrow Cname, \{CID, PID\} \rightarrow Pname, \{CID, PID\} \rightarrow Age$
- $\{CID, PID\} \rightarrow Price, \{CID, PID\} \rightarrow CID, \{CID, PID\} \rightarrow PID$

# Inference Rules (1)

- Armstrong's Inference Rules

(단,  $X, Y, Z$  : a set of attributes in a relation R)

예: Student(ID, name, age, course, grade, major)

- IR1 : (Reflexive) If  $Y \subseteq X$ , then  $X \rightarrow Y$ .

예:  $\{name, age\} \rightarrow name$ ,  $\{name, age\} \rightarrow age$

- IR2 : (Augment) If  $X \rightarrow Y$ , then  $\{X, Z\} \rightarrow \{Y, Z\}$ .

예: If  $ID \rightarrow age$ , then  $\{ID, name\} \rightarrow \{age, name\}$ .

- IR3 : (Transitive) If  $X \rightarrow Y$  and  $Y \rightarrow Z$ , then  $X \rightarrow Z$ .

예: If  $ID \rightarrow name$  and  $name \rightarrow age$ , then  $ID \rightarrow age$ .

# Inference Rules (2)

- Armstrong's Inference Rules

- IR4 : (Decompose) If  $X \rightarrow \{Y, Z\}$ , then  $X \rightarrow Y$  and  $X \rightarrow Z$ .

예: If  $ID \rightarrow \{\text{name}, \text{age}\}$ , then  $ID \rightarrow \text{name}$  and  $ID \rightarrow \text{age}$ .

- IR5 : (Union) If  $X \rightarrow Y$  and  $X \rightarrow Z$ , then  $X \rightarrow \{Y, Z\}$ .

예: If  $ID \rightarrow \text{name}$  and  $ID \rightarrow \text{age}$ , then  $ID \rightarrow \{\text{name}, \text{age}\}$ .

➤ 참고 : If  $\{X, Y\} \rightarrow Z$ , then  $X \rightarrow Z$  and  $Y \rightarrow Z$  는 성립하지 않음.

예: 만약  $\{ID, \text{course}\} \rightarrow \text{grade}$  일 때,

$ID \rightarrow \text{grade}$ ,  $\text{course} \rightarrow \text{grade}$  는 성립 안 함.

# Trivial FDs are Mostly Ignored

- A FD:  $X \rightarrow Y$  where  $Y \subseteq X$  (by IR1) is called **trivial**. Otherwise, it is non-trivial.
- Every trivial FD holds in every relation.
- Example: Let  $R = \{A, B, C\}$ ; Then,  
 $\{A, B, C\} \rightarrow \{A, B\}$ ,  $\{A, B, C\} \rightarrow \{B, C\}$ ,  
 $\{A, B\} \rightarrow \{A\}$ ,  $\{A\} \rightarrow \{A\}$ , . . . are all trivial FDs.
- Too many trivial FDs in a relation exist; Thus, they are trivial!  
For this reason, we mostly consider only non-trivial FDs.

# Three Basic Rules

- The rules IR4, IR5 can be derived from IR1, IR2, IR3.
- Given a set of FDs  $F$ , we can infer “additional new” FDs that hold whenever the FDs in  $F$  hold.
- IR1, IR2, and IR3 are called “sound” and “complete”.
  - **Sound** : Derive only correct FDs that truly are held;  
(= Does not derive any incorrect FDs that are not held.)
  - **Complete** : Derive all possible correct FDs that are held.



## Three Basic Rules

- IR4 : (Decompose) If  $X \rightarrow \{Y, Z\}$ , then  $X \rightarrow Y$  and  $X \rightarrow Z$ .

Proof: Using 3 Basic Rules

(Given) 1)  $X \rightarrow \{Y, Z\}$

(By Reflexive IR1) 2)  $\{Y, Z\} \rightarrow Y$ ,

(By Transitive IR3) From 1), 2), we infer  $X \rightarrow Y$

Similar proof for  $X \rightarrow Z$ .

- IR5 : (Union) If  $X \rightarrow Y$  and  $X \rightarrow Z$ , then  $X \rightarrow \{Y, Z\}$ .

Proof: Using 3 Basic Rules

(Given) 1)  $X \rightarrow Y$  and 2)  $X \rightarrow Z$ ,

(By Augment IR2) 3)  $X \rightarrow \{X, Y\}$ , 4)  $\{X, Y\} \rightarrow \{Y, Z\}$

(By Transitive IR3), we infer  $X \rightarrow \{Y, Z\}$

# Why Inference Rules Needed?

- When designing relation  $R$ , we specify a set of FDs for  $R$ .
- Database designer usually explicitly states only FD's which are obvious.
- Without knowing exactly what all tuples are, we must be able to derive other new FD's that hold on  $R$ .
- By using inference rules, we can infer new FDs from  $R$ .
- This is essential when we discuss design of good relations.

# Use of Inference Rules : Example

- $R = (A, B, C, G, H, I)$   
F :
  - 1)  $A \rightarrow B$
  - 2)  $A \rightarrow C$
  - 3)  $\{C, G\} \rightarrow H$
  - 4)  $\{C, G\} \rightarrow I$
  - 5)  $B \rightarrow H$
- Some rules derived from F
  - $A \rightarrow H$  (by transitivity from 1)  $A \rightarrow B$  and 5)  $B \rightarrow H$ )
  - $\{A, G\} \rightarrow I$   
(by augmenting 2)  $A \rightarrow C$  with  $G$ , to get  $\{A, G\} \rightarrow \{C, G\}$   
and then, transitivity with 4)  $\{C, G\} \rightarrow I$ )
  - $\{C, G\} \rightarrow \{H, I\}$   
(by augmenting 4)  $\{C, G\} \rightarrow I$  to infer  $\{C, G\} \rightarrow \{C, G, I\}$ ,  
and augmenting 3)  $\{C, G\} \rightarrow H$  with  $I$  to infer  
 $\{C, G, I\} \rightarrow H, I$ , and then, transitivity)

# Use of Inference Rules : Example

- Product(name, category, color, department, price)  
F: 1) name  $\rightarrow$  color  
2) category  $\rightarrow$  department  
3) {color, category}  $\rightarrow$  price
- Is {name, category}  $\rightarrow$  price derived from F? Yes  
1) We apply augment rule IR2 to 1) name  $\rightarrow$  color to obtain;  
4) {name, category}  $\rightarrow$  {color, category}  
2) Then, apply transitive rule IR3 to 4), 3) to obtain;  
{name, category}  $\rightarrow$  price
- Is {name, category}  $\rightarrow$  color derived from F? Yes  
1) We apply reflexive rule IR1 to to obtain;  
5) {name, category}  $\rightarrow$  name  
2) Then, apply transitive rule IR3 to 5), 1) to obtain;  
{name, category}  $\rightarrow$  color

# Closure of FDs

- **Closure** of a set of FDs **F** (denoted by **F<sup>+</sup>**) is the set of all FDs that can be inferred from **F**
- Let **X** be a set of attribute(s) that appear on LHS (Left Hand Side) of each FD in **F**. Then, **closure** of **X** with respect to **F** (denoted by **X<sup>+</sup>**) is the set of **all attributes that are functionally decided by X**.
- **X<sup>+</sup>** can be calculated by repeatedly applying IR1, IR2, IR3 using the FDs in **F**

# Closure Test Algorithm

- Compute the **closure** of **X** (denoted by **X<sup>+</sup>**) under F.

(1) Basis: Let **X<sup>+</sup>** = **X**.

(2) Induction:

- Step 1: Look for a FD with left hand side **Y** that is a subset of the current **X<sup>+</sup>**. (that means, **Y**  $\subseteq$  **X<sup>+</sup>**)
- Step 2: If the FD is **Y**  $\rightarrow$  **Z**, add **Z** to **X<sup>+</sup>**.
- Repeat step 1 and 2 until no more FDs are derived.

# Closure of $X$ : 예 1

- $R = \{A, B, C, D, E, F\}$

$F : 1) A \rightarrow B$

2)  $C \rightarrow D, E$

3)  $\{A, C\} \rightarrow F$

- Given  $F$ , compute the following closure sets  $\mathbf{X}^+$ .

-  $\{A\}^+ = \{A, B\}$  ( $\because \{A\}^+ = \{A\}$ ; Then, by rule 1),  $\{A\}^+ = \{A, B\}$  )

-  $\{C\}^+ = \{C, D, E\}$  ( $\because \{C\}^+ = \{C\}$ ; Then, by rule 2),  $\{C\}^+ = \{C, D, E\}$  )

-  $\{A, C\}^+ = \{A, C, B, D, E, F\}$

( $\because \{A, C\}^+ = \{A, C\}$ ;

Then, by rule 1),  $\{A, C\}^+ = \{A, C, B\}$  ) ( $A \subseteq \{A, C\}$ )

Then, by rule 2),  $\{A, C\}^+ = \{A, C, B, D, E\}$  ( $C \subseteq \{A, C, B\}$ )

Then, by rule 3),  $\{A, C\}^+ = \{A, C, B, D, E, F\}$  ( $\{A, C\} \subseteq \{A, C, B, D, E\}$ )

## Closure of X : 예 2

- $R = \{A, B, C, D\}$

F : 1)  $\{A, B\} \rightarrow C$

2)  $\{A, D\} \rightarrow B$

3)  $B \rightarrow D$

- Given F, we compute the following closure sets.

$\{A, B\}^+ = \{A, B, C, D\}$ ; New FD  $\{A, B\} \rightarrow D$  를 유도함

$\{A, D\}^+ = \{A, B, C, D\}$ ; New FD  $\{A, D\} \rightarrow C$  를 유도함

$\{B\}^+ = \{B, D\}$ ;



## Closure of X : 예 3

- Class(ID, course#, prof#, credit, text, publisher, room, capacity)  
F : 1)  $ID \rightarrow \{course\#, prof\#, credit, text, publisher, room, capacity\}$   
2)  $course\# \rightarrow credit$   
3)  $\{course\#, prof\#\} \rightarrow \{text, room\}$   
4)  $text \rightarrow publisher$   
5)  $room \rightarrow capacity$

- Given F, we compute the following closure sets.

$$\{ID\}^+ = \text{[icon]}$$

$$\{course\# \}^+ = \text{[icon]}$$

$$\{course, prof\#\}^+ = \{course\#, prof\#, credit, text, publisher, room, \text{[icon]} capacity\}$$

$$\{text\}^+ =$$

$$\{room\}^+ =$$

- Question: What is key?    Key  $\equiv \{ID\}$

# Key와 FD의 관계

- Let  $R = \{ A_1, A_2, \dots, A_n \}$  and  $X \subseteq R$  ;  
**X** is a **key** if the following two conditions are held;

## (1) Uniqueness

$X \rightarrow \{A_1, A_2, \dots, A_n\} = R \in F^+ \quad (\text{즉, } \{X\}^+ = R)$   
(= X decides **all** attributes in R)

## (2) Minimality

For no proper subset  $Y \subseteq X$ ,  
 $Y \rightarrow \{A_1, A_2, \dots, A_n\} = R \in F^+ \quad (\text{즉, } \{Y\}^+ \neq R \text{ for any } Y)$   
(= Any proper subset of X is not a super key)

- If condition (1) only is held, X is a **Super Key**.

# Key와 FD의 관계

- MOVIE (title, year, length, color, director, producer)
- Let's consider the following FDs;  
F : 1) {title, year}  $\rightarrow$  length  
2) {title, year}  $\rightarrow$  color  
3) {title, year}  $\rightarrow$  director  
4) {title, year}  $\rightarrow$  producer
- Given a title and a year, there is a unique length, a unique color, a unique director, and a unique producer.
- Thus, {title, year} is a super key;
- Question : Is {title, year} also a key?
- Question : Is {title, year, color} also a key?

# FD로부터 Key 구하는 법

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

F :    1)  $A \rightarrow B$   
         2)  $A \rightarrow C$   
         3)  $\{C, G\} \rightarrow H$   
         4)  $\{C, G\} \rightarrow I$   
         5)  $B \rightarrow H$

- Is  $\{A, G\}$  a key?    "Yes"

1) Is  $\{A, G\}$  a super key? ( $= \{A, G\}^+ = R$ ?)

- $\{A, G\}^+ = \{A, G\}$
- $\{A, G\}^+ = \{A, B, C, G\}$  (because  $A \rightarrow C$  and  $A \rightarrow B$ )
- $\{A, G\}^+ = \{A, B, C, G, H\}$  (because  $C, G \rightarrow H$ )
- $\{A, G\}^+ = \{A, B, C, G, H, I\}$  (because  $C, G \rightarrow I$ )

2) Is any proper subset of  $\{A, G\}$  a superkey?

- Does  $A \rightarrow R$ ? No! (because  $\{A\}^+ = \{A, B, C, H\}$ )
- Does  $G \rightarrow R$ ? No! (because  $\{G\}^+ = \{G\}$ )

# FD로부터 Key 구하는 법

## ADDRESS

city	street	zip
NY	16th	23508
NY	17th	23508
NY	30th	23509
NY	34th	23509
LA	16th	90600

F : 1) {city, street}  $\rightarrow$  zip  
2) zip  $\rightarrow$  city

- There are two keys; Why?

Key 1 = {city, street}

Key 2 = {street, zip}

- {city, street, zip} is a super key, but not a key.
- {city, zip} is neither a super key nor a key.

# FD로부터 Key 구하는 법 : Exercise

- Find key(s) in each relation

예 1 : PRODUCT (name, price, category, color)

F : 1) {name, category}  $\rightarrow$  price  
2) category  $\rightarrow$  color

예 2 : ENROLL (student, address, course, room, time)

F : 1) student  $\rightarrow$  address  
2) {room, time}  $\rightarrow$  course  
3) {student, course}  $\rightarrow$  {room, time}

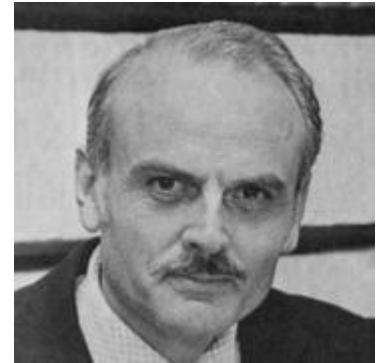
# Finding a Key Algorithm

- Given a set **F** of functional dependencies, Find a key **K** for relation **R**.
  - (1) Set **K = R**.
  - (2) For each attribute **A** in **K**
    - a) Compute  $(\mathbf{K} - \mathbf{A})^+$  with respect to **F**
    - b) If  $(\mathbf{K} - \mathbf{A})^+$  contains **all** the attributes in **R**, then  
then set  $\mathbf{K} = \mathbf{K} - \{\mathbf{A}\};$
- We start by setting K all the attributes of R.
- We then remove one attribute at a time and check the remaining attributes still form a super key.

# Normalization : History

- **E. F. Codd** : English Computer Scientist;

While working for IBM, he first proposed the process of normalization and established normal forms: 1NF, 2NF, 3NF, BCNF;



*"A Relational Model of Data for Large Shared Data Banks"*

- "There is, in fact, a very simple elimination procedure which we shall call **normalization**. Through **decomposition**, non-simple domains are replaced by "domains whose elements are atomic (= non-decomposable) values."



# Normalization

- Normalization:
  - Process of **decomposing "bad"** relations into "smaller", but **"good"** (no redundancy, no anomalies) relations
- Normal forms:
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - Boyce Codd Normal Form (BCNF)
  - Fourth Normal Form (4NF)
- 2NF, 3NF, BCNF: based on **keys** and **FDs**.
- 4NF: based on **keys** and **MVDs**.

# First Normal Form (1NF)

- A relation **R** is **1NF** if
  - (1) each tuple must be identified by primary key;
  - (2) each attribute must have **atomic** (= **only one**) value;
- In other words, **R** does not allow
  - a) multi-valued attribute
  - b) composite attribute
  - c) Combination of a) and b): "Nested relation"

# First Normal Form (1NF)

- Consider a PERSON relation; This is not in 1NF because {phones} is multi-valued attribute.

PERSON(SSN, name, age, {phones}), PK = {SSN}

F : SSN  $\rightarrow$  {name, age}

PERSON

<u>SSN</u>	name	age	phones
100	abe	57	{1234, 2345, 3456}
200	bob	27	4567
300	eve	45	{5678, 6789}

- There are 3 methods to normalize into 1NF.

## Convert into 1NF : Method A

- Method A: Expand the key so that there is a separate tuple for each phone number;

PERSON			PK = {SSN, phones}
<u>SSN</u>	name	age	<u>phones</u>
100	abe	57	1234
100	abe	57	2345
100	abe	57	3456
200	bob	27	4567
300	eve	45	5678
300	eve	45	6789

- The following problems may occur;
  - Primary key is changed as {SSN, phones}.
  - Redundancy: Same information(SSN, name, age) is repeated.
  - Anomalies may occur.

## Convert into 1NF : Method B

- Method B; If maximum number of phone values(say, 3) is known, replace phone attribute by 3 atomic attributes.

PERSON      PK = {SSN}					
<u>SSN</u>	name	age	phone1	phone2	phone3
100	abe	57	1234	2345	3456
200	bob	27	4567	null	null
300	eve	45	5678	6789	null

- The following problems may occur;
  - Many null values exist.
  - Spurious semantics about ordering among phone values.
  - Hard to query.
  - Hard to redesign ; What if few more phones are added?

## Convert into 1NF : Method C

- Method C; Remove phones attribute that violates 1NF and place it in a separate relation; Decompose it!

PERSON

<u>SSN</u>	name	age
100	abe	57
200	bob	27
300	eve	45

PK =  
{SSN}

PHONE

<u>SSN</u>	<u>phones</u>
100	1234
100	2345
100	3456
200	4567
300	5678
300	6789

PK = {SSN,  
phone}

- Method C is the best; Completely general!
  - No redundancy
  - No anomalies
  - Easy to query.
  - Easy to redesign

## More than 1 Multivalued Attributes

- Consider a PERSON relation; It has 2 multivalued attributes;

PERSON(SSN, {hobbies}, {phones})

- Suppose we use method A;

PERSON(SSN, hobbies, phones)

- Key is a {all attributes};
- Lots of redundancy! All possible of combination of values for every ID.

- We recommend Method C;

PERSON1(SSN, hobbies)

PERSON2(SSN, phones)

# Nested Relations

- Consider STUDENT relation; This is not in 1NF because {transcript} is multi-valued/composite attribute; That means "Nested Relation (= Relation within relation)"!

STUDENT(ID, name, {transcript(course, year, grade)})

STUDENT PK: SSN

<u>ID</u>	name	course	year	grade
100	abe	DB	2019	A
		OS	2018	B
		DS	2018	B
200	bob	OS	2019	C
		network	2017	A
		. . .	. . .	. . .

"course" is a partial key; Within each tuple, nested relation has unique course value.

- Remove nested relation transcript and put it in separated relation;

STUDENT(ID, name)

TRANSCRIPT(ID, course, year, grade) PK: {ID, course}



# Multilevel Nested Relations

- Decomposition can be applied repeatedly to a relation with multilevel nesting to get 1NF.

- Consider a CANDIDATE relation;

CANDIDATE(ID, Name, {Job-Hist(Company, Position, {Sal-Hist(Year, Sal))})})

- Using decomposition repeatedly, we convert CANDIDATE into;

CANDIDATE1 (ID, Name), PK = {ID}

CANDIDATE-Job-Hist (ID, Company, Position),

PK = {ID, Company}, Company is a partial key

CANDIDATE-Sal-Hist (ID, Company, Year, Sal),

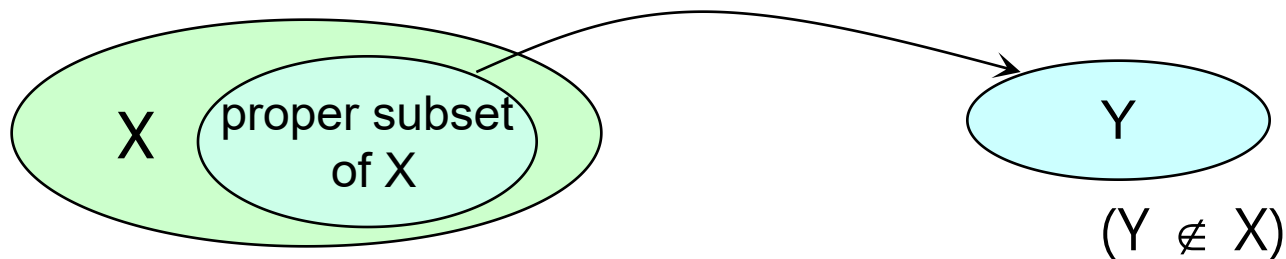
PK = {ID, Company, Year}, Year is a partial key

# Normalization : 기본 개념

- (지금부터) 모든 relation은 1NF라고 가정함.
- Redundancy와 Anomaly 현상은 "bad" FD들이 존재할 때 발생함;
- 이러한 "bad" FD들을 단계별로 찾아내서 소거하는 것이 정규화의 기본 개념. 다음은 전형적인 "bad" FD들;
  - Key에 부분 종속되는 FD들을 소거 : 2NF
  - Key에 이행 종속되는 FD 들을 소거 : 3NF
  - Super Key가 아닌 것에 종속되는 FD들을 소거 : BCNF

# Partial Dependency (부분 종속)

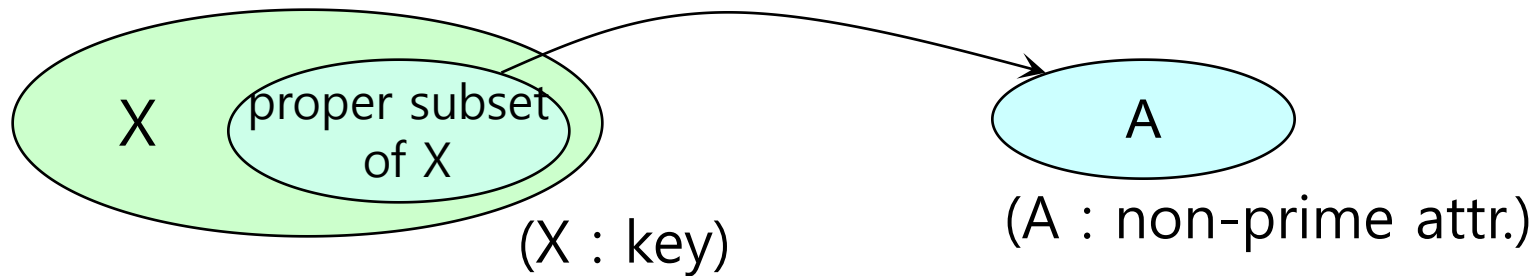
- An attribute that is a **member** of some key is called a **prime** attribute. An attribute that is **not a member** of any key is **non-prime** attribute.
- Given FD  $X \rightarrow Y$ ,  $Y$  is **partially dependent** on  $X$  if there exists FD such that  $Y$  is dependent of a **proper subset** of  $X$ .



$Y$  is partially dependent on  $X$

## Second Normal Form (2NF)

- A relation **R** is **2NF** if any non-prime attribute is not partially dependent of any key.



- This is not in 2NF because non-prime attribute **A** is partially dependent on key **X**.

## 2NF : Example

<u>SSN</u>	<u>PNO</u>	ename	pname	location	hours
------------	------------	-------	-------	----------	-------

Key = {SSN, PNO}

- F:
- 1) {SSN, PNO} → hours
  - 2) SSN → ename
  - 3) PNO → {pname, location}

- Is this relation in 2NF?

- No! Why? **Non-prime** attribute 'ename' is **partially** dependent

on **key** '{SSN, PNO}' because {SSN, PNO} → ename also holds.  
and 2) SSN → ename is given,

Each of **non-prime** attributes {pname, location} is **partially** dependent on **key** {SSN, PNO} because {SSN, PNO} → pname, {SSN, PNO} → location also holds and 3) PNO → {pname, location} is given.

## 2NF : Example

- We decompose as follows:

SSN	PNO	ename	pname	location	hours
-----	-----	-------	-------	----------	-------

- F:
- 1)  $\{SSN, PNO\} \rightarrow hours$
  - 2)  $SSN \rightarrow ename$
  - 3)  $PNO \rightarrow pname, location$

(2NF) ↘

SSN	PNO	hours
-----	-----	-------

PK = {SSN, PNO}

(2NF) ↘

SSN	ename
-----	-------

PK = {SSN}

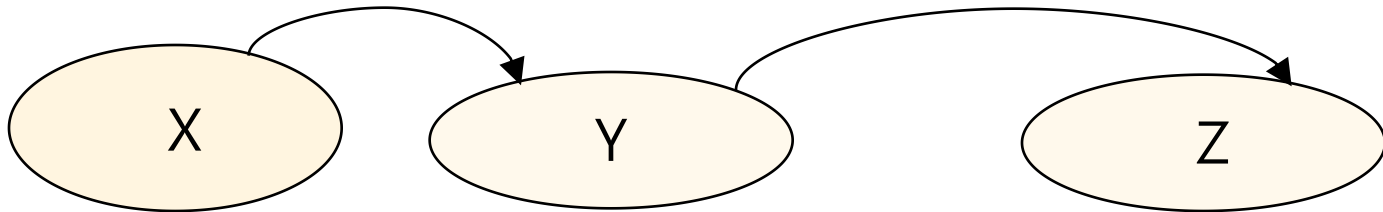
(2NF) ↘

PNO	pname	location
-----	-------	----------

PK = {PNO}

# Transitive Dependency (이행종속)

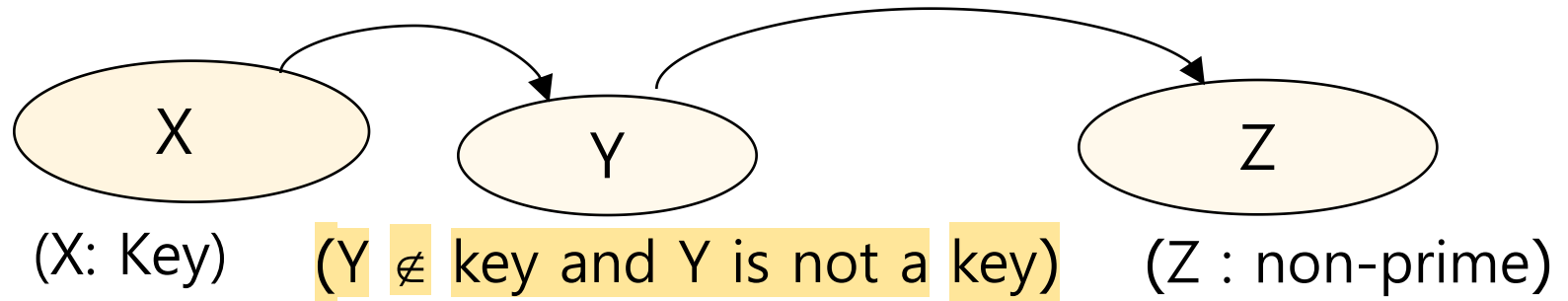
- Given FD  $X \rightarrow Z$ ,  $Z$  is **transitive dependent** on  $X$  if there exist FDs such that  $X \rightarrow Y$  and  $Y \rightarrow Z$



$Z$  is transitive dependent on  $X$  (via  $Y$ )

# Third Normal Form (3NF)

- A relation R is 3NF if **any non-prime** attribute is not transitive dependent of **any key**.



- This is not in 3NF because non-prime attribute **Z** is transitive dependent on key **X**



# Third Normal Form (3NF)

- (Another Definition)

A relation R is 3NF if for every FD :  $X \rightarrow A$ ,

(1) X is a **super key**,

(= X contains any key)

or

(2) A is a **prime attribute**

(= A is a member of some key)

# 3NF : Example

SSN	ename	age	DNO	dname
-----	-------	-----	-----	-------

Key = {SSN}

F: 1)  $SSN \rightarrow DNO$   
2)  $DNO \rightarrow dname$

- Is this relation 3NF?
- No! Why? **Non-prime** attribute '**dname**' is transitive dependent on **key** '**SSN**' because  $SSN \rightarrow dname$  also holds.
- In other words, in 2)  $DNO \rightarrow dname$ , '**DNO**' is not a **super key** and also, '**dname**' is not a **prime** attribute;

# 3NF : Example

- We decompose as follows:

SSN	ename	age	DNO	dname
-----	-------	-----	-----	-------

- F: 1)  $SSN \rightarrow DNO$   
2)  $DNO \rightarrow dname$

(3NF)



SSN	ename	age	DNO
-----	-------	-----	-----

PK = {SSN}

(3NF)



DNO	dname
-----	-------

PK = {DNO}

# Normalization : Exercise (1)

## STUDENT

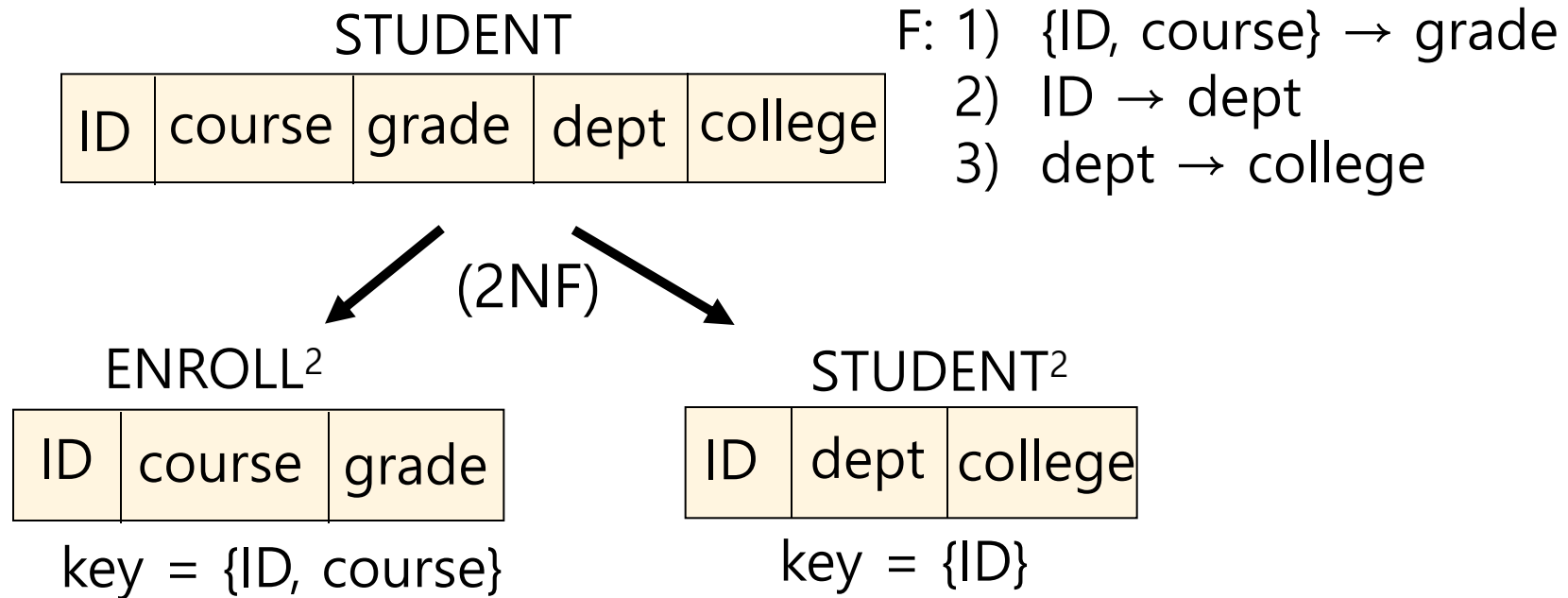
ID	course	grade	dept	college
----	--------	-------	------	---------

- F: 1)  $\{ID, \text{course}\} \rightarrow \text{grade}$   
2)  $ID \rightarrow \text{dept}$   
3)  $\text{dept} \rightarrow \text{college}$

- What is key?
- This relation is not in 2NF because non-prime attribute {dept} is partially dependent on key {ID, course}
- We have the following problems: Explain!
  - Redundancy
  - Insert Anomaly
  - Delete Anomaly
  - Update Anomaly

## Normalization : Exercise (2)

- We decompose STUDENT as follows:



- In both relations, no more problems in 2NF; Explain!
  - No Redundancy
  - No Insert Anomaly
  - No Delete Anomaly
  - No Update Anomaly

# Normalization : Exercise (3)

STUDENT<sup>2</sup>

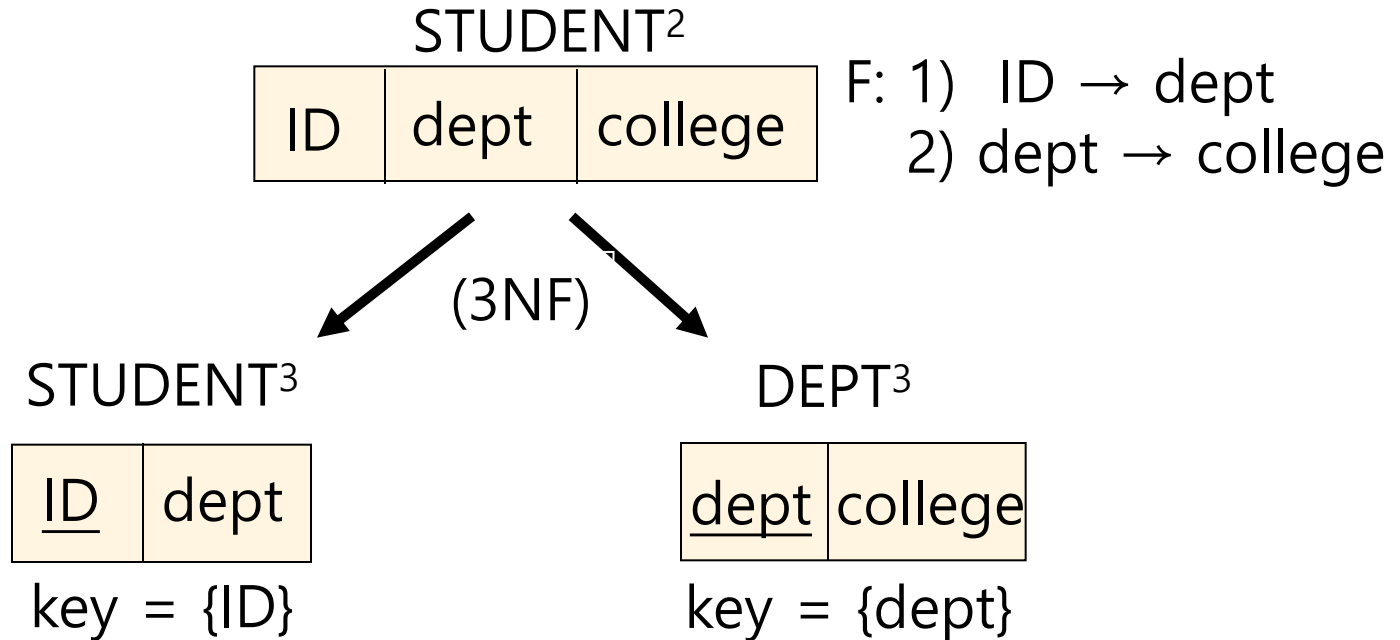
ID	dept	college
----	------	---------

F: 1)  $ID \rightarrow dept$   
2)  $dept \rightarrow college$

- This relation is not in 3NF because non-prime attribute 'college' is transitive dependent on key = {ID};
- In other words, in 2)  $dept \rightarrow college$ , 'dept' is not a super key; also, 'college' is not a prime attribute.
- We still have the following problems: Explain!
  - Redundancy
  - Insert Anomaly
  - Delete Anomaly
  - Update Anomaly

# Normalization : Exercise (4)

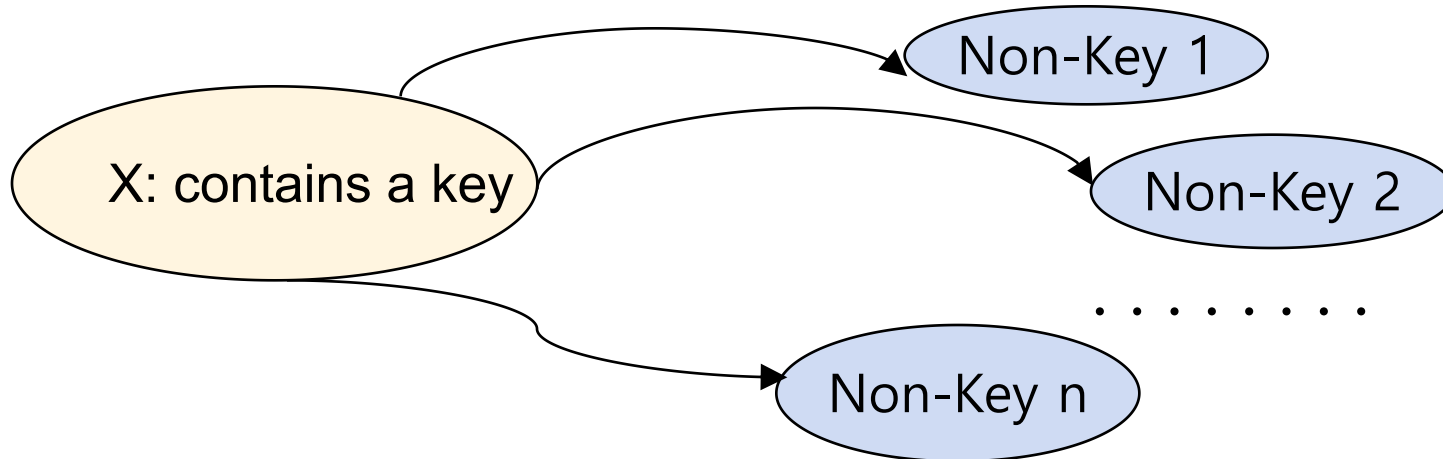
- We decompose STUDENT<sup>2</sup> as follows:



- In the both relations, no more problems; Explain!
  - No Redundancy
  - No Insert Anomaly
  - No Delete Anomaly
  - No Update Anomaly

# Boyce-Codd Normal Form (BCNF)

- A relation **R** is in **BCNF** if for every **FD** :  **$X \rightarrow A$** , **X** is a **super key**
- In other words, **all FDs** that **hold over R** **satisfy super key constraints**





# BCNF : Example

## TEACH

student	course	prof
---------	--------	------

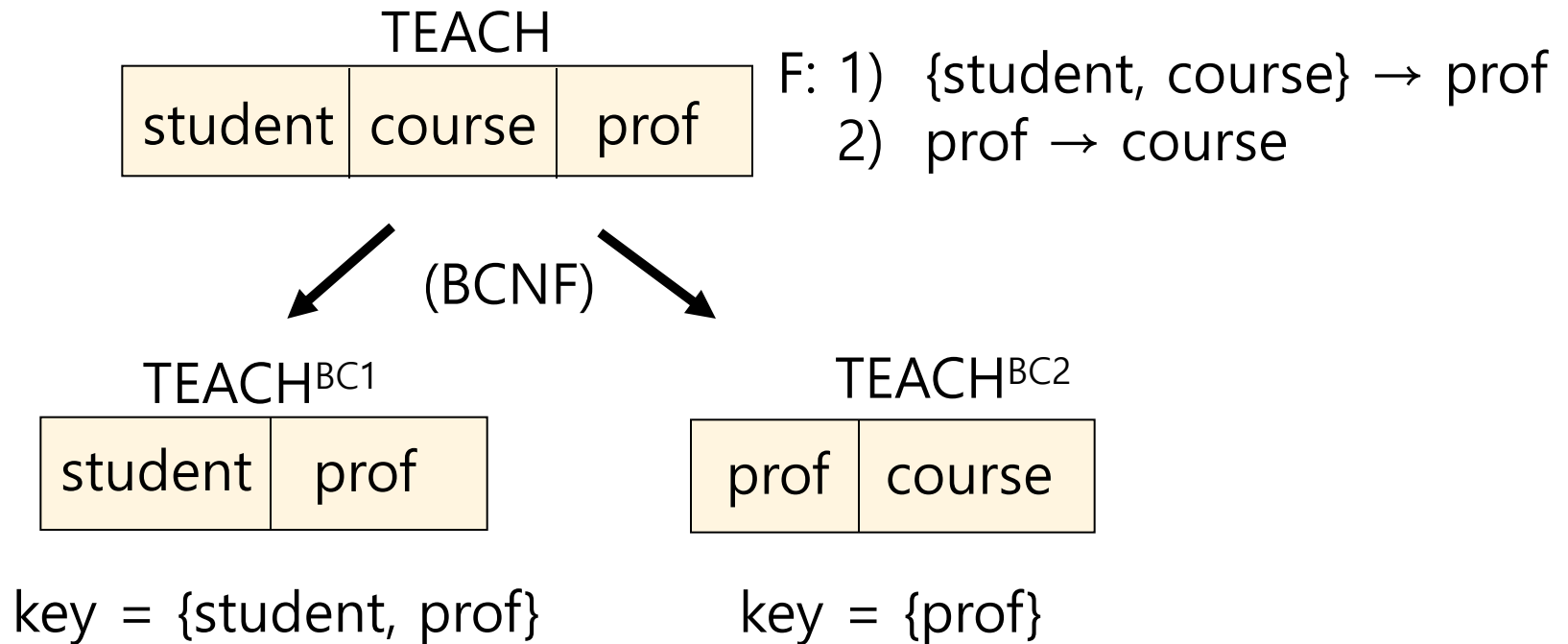
F: 1) {student, course}  $\rightarrow$  prof  
2) prof  $\rightarrow$  course

- There are 2 keys; {student, course}, {student, prof}
- This relation is in 3NF, but not BCNF; Why?  
In 2) prof  $\rightarrow$  course, 'prof' is not a super key;
- We still have redundancy problem: The same 'course' name is repeated many times. And, also any anomalies?

student	course	prof
ann	network	lee
bob	network	lee
eve	network	lee
ann	database	kim
bob	database	kim
eve	database	kim

# BCNF : Example

- We decompose as follows:



- In both BCNF relations, there exists no more any redundancy and no more anomaly problems.

# Why BCNF?

- Every BCNF guarantees no redundancy and no anomalies caused by FDs.
  - Since X is a super key, the two tuples must be identical ( $y_1 = y_2, z_1 = z_2$ ); But this is not allowed. Why?

X	Y	Z
x	$y_1$	$z_1$
x	$y_2$	$z_2$

- Suppose R is a binary relation;  $R = (A, B)$ 
  - Does R satisfy 2NF?
  - Does R satisfy 3NF?
  - Does R satisfy BCNF?

# Normal Forms

- Each normal form is strictly stronger than the previous one.
  - Every 2NF relation is in 1NF
  - Every 3NF relation is in 2NF
  - Every BCNF relation is in 3NF
- Higher Normal Forms give us;
  - Less Redundancy,
  - Less Anomalies,
  - Lower performance.
- Our goal is to get every relation in BCNF.
- But, typically, we trade off: BCNF vs 3NF; Why?