9. Relational Design

Main Issues:

- What are guidelines for "decomposing" a relation?
- What is a "lossless join (information preservation)"?
- What is a "dependency(FD) preservation"?
- What is a "BCNF decomposition" ?
- What are benefits of BCNF and 3NF?
- What is a "Multi-Valued Dependency(MVD)"?
- What is a "Fourth Normal Form(4NF)"?
- What is a "denormalization"?

Decomposition

- Decomposition (분해):
  (정규화 단계에서) relation R을 여러 개의 작은 relation $\{R_1, R_2, \ldots, R_m\}$ 들로 분해하는 과정

- 주의 사항:

  (1) **Information 보존 (= Lossless Join)**

  - 분해된 relation들로부터 원래 R의 정보가 정확히 복구되어야함.
  - 필수사항.

  (2) **Functional Dependency 보존**

  - Relation R의 FD들이 분해된 각 relation들에서 보존되어야 함.
  - 권장 사항

잘못 된 분해 : 예

- 다음의 분해는 올바른가? 야기되는 문제점은?

PERSON

SSN	name	age
11111	abe	28
22222	bob	31
33333	eve	50
44444	mat	38
55555	eve	19

π SSN, name

π name, age

PERSON1

SSN	name
11111	abe
22222	bob
33333	eve
44444	mat
55555	eve

PERSON2

name	age
abe	28
bob	31
eve	50
mat	38
eve	19

잘못 된 분해 : 예

- 다음 relation들로부터 원래의 정보가 정확히 복구되는지?

PERSON1

SSN	name
11111	abe
22222	bob
33333	eve
44444	mat
55555	eve

PERSON2

name	age
abe	28
bob	31
eve	50
mat	38
eve	19

PERSON1 * PERSON2

PERSON

SSN	name	age
11111	abe	28
22222	bob	31
33333	eve	50
33333	eve	19
44444	mat	38
55555	eve	50
55555	eve	19

Spurious Tuples!

Lossless Join (무손실 조인)

- Relation R을 $\{R_1, R_2, \ldots, R_m\}$들로 decompose한 경우,
  분해된 relation들을 natural join하여 복구하면 다음이 성립함.
  (단, join attribute들은 null 값이 없다는 전제조건)

$$R_1 * R_2 * \ldots * R_m \supseteq R$$

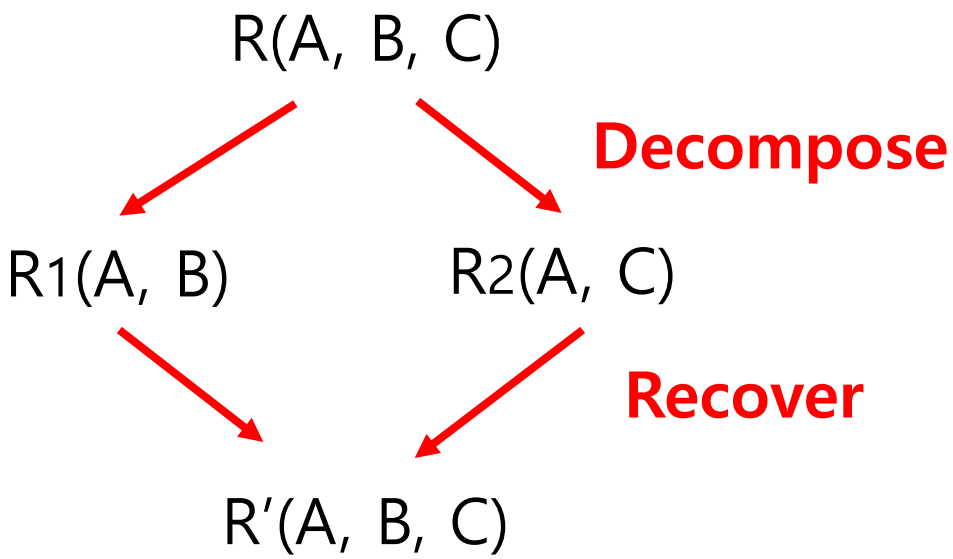
- 여기서 $\supset$ 인 경우, 원래 relation R에 없는 가짜(spurious) tuple들이 추가로 생성됨. 이를 손실 조인(**Lossy Join**) 이라 함.
  ("손실" 은 "tuple"의 손실이 아닌, "정보"의 손실을 의미)

- 다음 조건을 만족할 경우, 이를 **Lossless Join** 이라고 함.

$$\mathbf{R_1 * R_2 * \ldots * R_m = R}$$

Lossless Decomposition

- A decomposition is **lossless** if we can recover:



- R1(A, B) * R2(A, C) = R'(A, B, C) = R(A, B, C)

Lossless Join Testing

- **R**을 **R1**과 **R2**로 decompose 한다고 하자. 단,

  F : **R**에 주어진 FD들

  F$^+$: **F** 로부터 유도되는 모든 FD들

- 다음 FD들 중 하나가 **F**$^+$ 에 속한다면, Lossless Join이 보장됨.

  (1) **R1 ∩ R2 → R1 – R2**

  (2) **R1 ∩ R2 → R2 – R1**

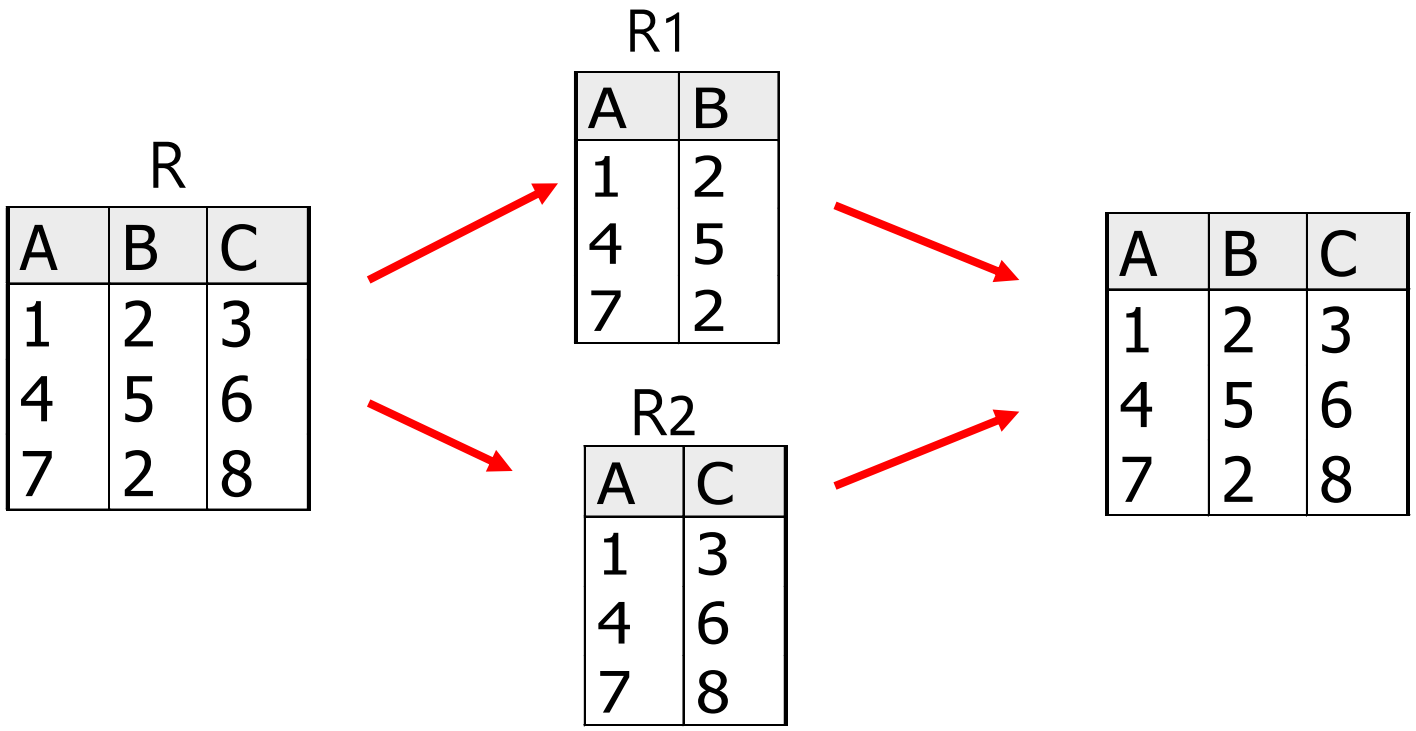
- 위에 대한 역도 성립됨. 즉, 이 조건을 만족하지 않으면 Lossless Join이 위반됨(Lossy Join!)

- Corollary: 만약 **R1 ∩ R2** 이 (**R1** 혹은 **R2**의) **key**를 포함하면, 이는 역시 무손실 조인이 보장됨.

Lossless Join : 예

- R = {A, B, C}

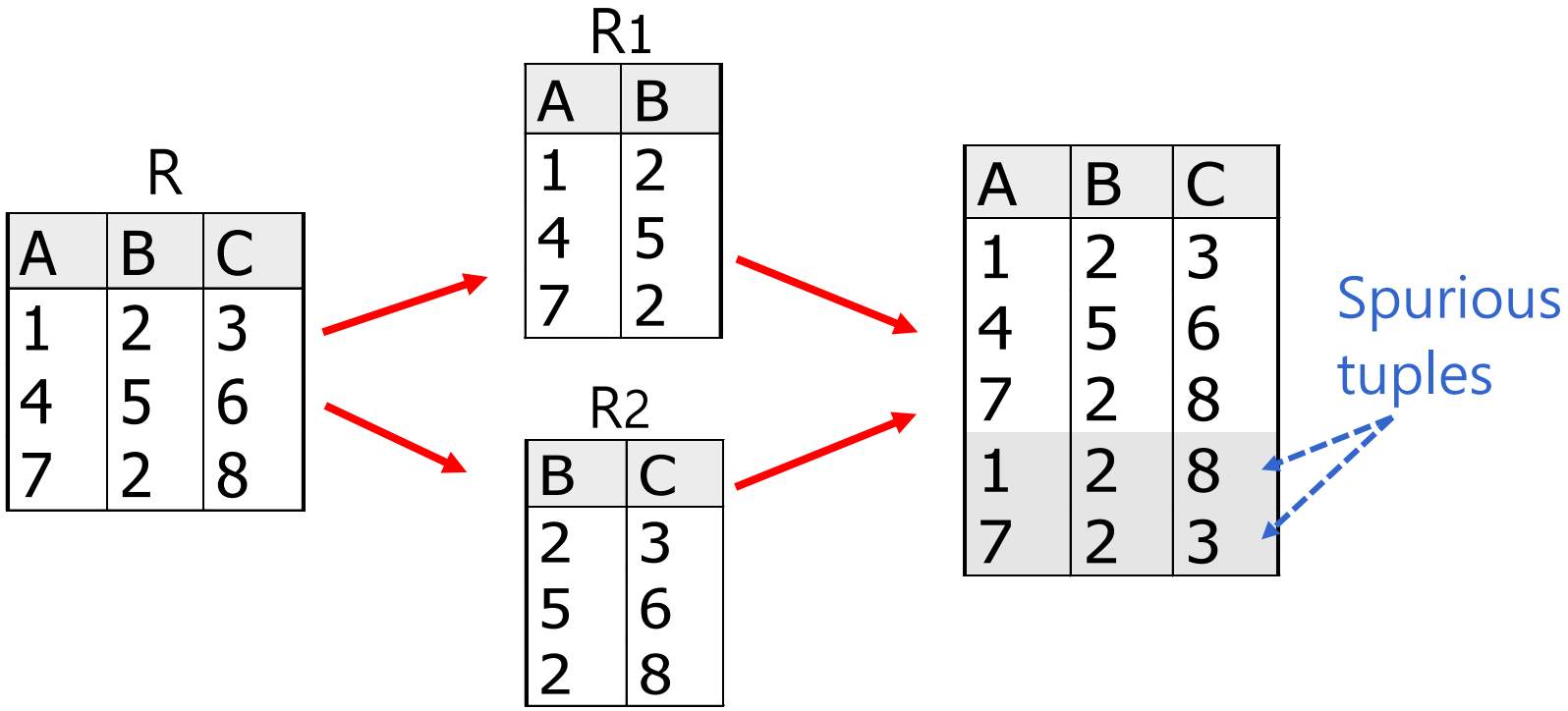
  F : A → B
- R1 = {A, B}, R2 = {A, C}로 분해;
- R1 ∩ R2 (= A) → R1 – R2 (= B) ∴ Lossless Join



Lossy Join : 예

- R = {A, B, C}

  F : A → B
- R1 = {A, B}, R2 = {B, C}로 분해;
- R1 ∩ R2 (= B) → ??? . . ∴ Lossy Join



Lossless Join : Exercise

- BANK = (branch#, city, assets, customer#, loan#, amount)
  F : 1) branch# → {city, assets}
  2) loan# → {amount, branch#}

- Suppose we decompose as follows;
  - BRANCH = (branch#, city, assets)
  - LOAN = (branch#, customer#, loan#, amount)

- Since BRANCH ∩ LOAN (= branch#) → {city, assets},
  this decomposition is a lossless join.

- Thus, Branch * LOAN = BANK

FD 보존의 의미

- FD 보존이 안 된다는 의미는? 어떤 문제점이 발생?
  (즉, Relation R에 주어진 어떤 FD가 분해된 relation 에서 보존이 안 된 경우)

- 분해된 relation에서 update 혹은 insert를 할 경우, FD 위반을 검증하여야 함. 이때 이 FD가 누락되었으므로 이 single relation 자체만으로 이를 검증할 수가 없음.

- 이 경우 매번 (2 개 이상의 relation들을) Join 연산을 하여야 함. 이는 매우 비효율적.

- 따라서 FD 보존이 보장되면, FD 위반의 대한 검증 비용을 줄 일 수 있음.

FD 보존 위반 : 예

ADDRESS

street	city	zip
16th	NY	23508
17th	NY	23509

F : 1) {street, city} → zip
2) zip → city

- 다음과 같이 decompose 했다고 가정하자.

street	zip
16th	23508
17th	23509

city	zip
NY	23508
NY	23509

- {street, city} → zip 이 보존이 안 됨.

- 이 경우 새로운 tuple (예: <16th, NY, 23510>)이 insert 될 때마다 (이 FD 위반 여부의 검증을 위해) 매번 join 연산이 필요함.

FD 보존 Testing

- R을 $\{R_1, R_2, \ldots, R_m\}$으로 분해한다고 하자. 단,

  F : R에 주어진 FD들의 집합

  F^+ : F로부터 유도되는 모든 FD들의 집합

  Fi : 각 R_i 의 attribute들만 포함하는 F^+의 모든 FD들의 집합

- 다음을 만족하면 모든 FD 보존이 보장됨..

  $$(F1 \cup F2 \cup \ldots \cup Fm)^+ = F^+$$

- 즉, R에서 성립하는 모든 FD들의 집합 F^+와 분해된 각 relation 에서 성립하는 FD들을 모두 합한 것의 closure가 서로 같음.

FD 보존 : 예

- R = {A, B, C}

  F : { 1) A → B, 2) B → C, 3) C → A }

- R_1 = {A, B}, R_2 = {B, C}으로 분해 시, 이는 FD 보존인가?

  (즉, C → A 가 보존되는지?)

  - $F^+ = F \cup \{A \rightarrow C, B \rightarrow A, C \rightarrow B\}$
  - F1 = {A → B, B → A}
  - F2 = {B → C, C → B}
  - $(F1 \cup F2)^+$

    $= \{\{A \rightarrow B, B \rightarrow A\} \cup \{B \rightarrow C, C \rightarrow B\}\}^+$

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

    $= F^+$

- 즉, C → A 도 (간접적으로) 보존되므로, FD 보존임.

Lossless Join / FD 보존 : 예

- R = (A, B, C)

  F : { 1) $A \rightarrow B$, 2) $B \rightarrow C$}

- 예 1 : $R_1 = \{A, B\}$ $R_2 = \{B, C\}$ 로 분해
  - $R_1 \cap R_2$ (= B) $\rightarrow R_2 - R_1$ (= C) ∴ Lossless Join
  - $(F1 \cup F2)^+ = F^+$ ∴ FD 보존

- 예 2 : $R_1 = \{A, B\}$ $R_2 = \{A, C\}$ 로 분해
  - $R_1 \cap R_2$ (= A) $\rightarrow R_1 - R_2$ (= B) ∴ Lossless Join
  - $F^+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$
  - $F1 = \{A \rightarrow B\}$, $F2 = \{A \rightarrow C\}$
  - $(F1 \cup F2)^+ = \{A \rightarrow B, A \rightarrow C\} \neq F^+$

  ∴ $B \rightarrow C$ 가 보존이 안 됨.

Lossless Join with No Redundancy

TEACH

student	course	prof

F: 1) {student, course} → prof
2) prof → course

Keys = {student, course}, {student, prof}

- This relation is not in BCNF, because FD 2) is violated.
- We have redundancy problem; <course> is repeated many times, because 'prof' is not a super key.

student	course	prof
ann	network	lee
bob	network	lee
eve	network	lee
ann	database	kim
bob	database	kim
eve	database	kim
dick	network	park
joe	network	park

Lossless Join with No Redundancy

TEACH

student	course	prof

F: 1) {student, course} → prof
2) prof → course

Keys = {student, course}, {student, prof}

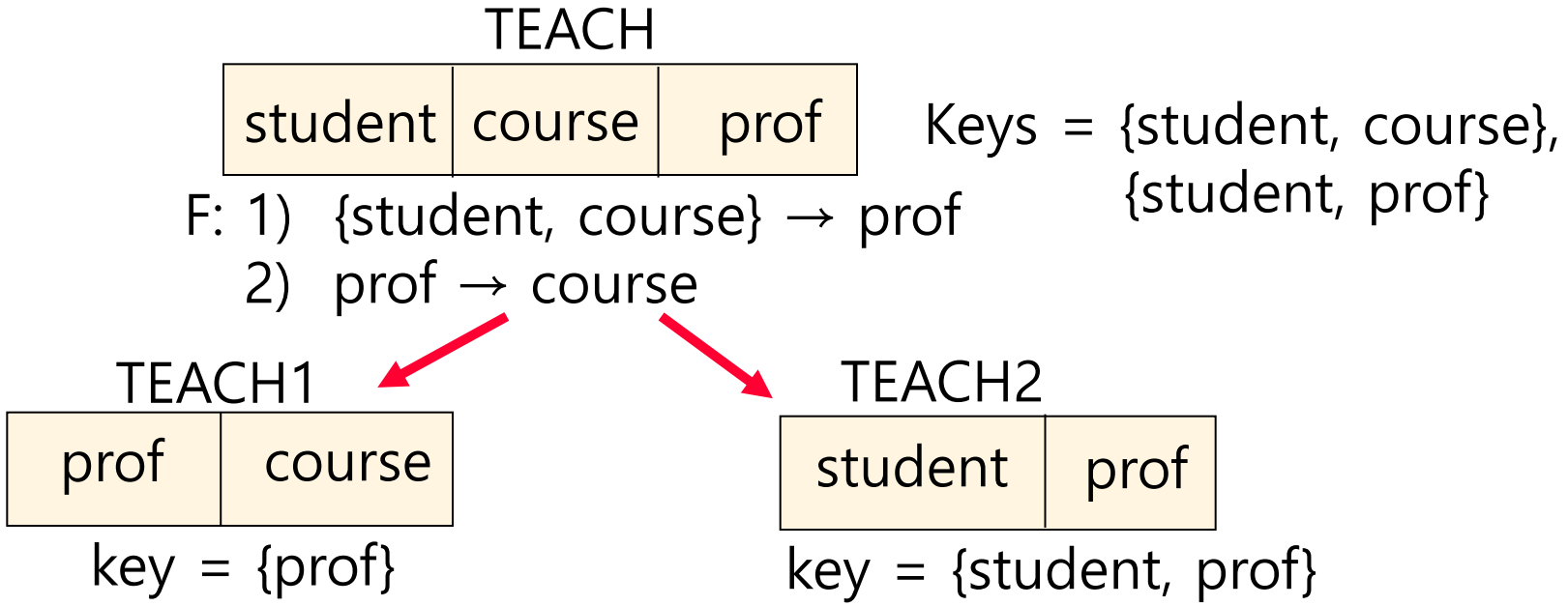
- 다음들 중 lossless 이고, redundancy가 발생하지 않는 분해는? (즉, lossless join과 BCNF를 모두 만족하는 분해) Answer: (3)

(1) R1 = (student, course), R2 = (prof, course): No/Yes

(2) R1 = (student, prof), R2 = (course, student): No/Yes

(3) R1 = (prof, course), R2 = (student, prof): Yes/Yes

Lossless Decomposition into BCNF



(1) BCNF를 위반하는 FD 2)의 attribute들을 모두 한 relation으로 합침; (이때, TEACH1에서 {prof}는 key가 됨; 따라서 BCNF 만족)

(2) 다른 relation은 TEACH에서 FD 2)의 우변 attribute를 제거한 나머지 attribute들로 구성 (이때, TEACH2에서 key는 {student, prof}이고, 역시 BCNF 만족. 또한 양쪽 relation의 공통인 {prof}는 Key가 되면서 lossless 만족)

- This decomposition satisfies both BCNF and lossless join;

Decomposition into BCNF

- **Lossless BCNF Decomposition Algorithm**

Input : A relation **R** and a set of FDs **F**

Output : **D** = {R_1, R_2, . . ., R_m} where each R_i : **BCNF**

Let **D** = {**R**};

While (there is a relation **Q** in **D** that is not in **BCNF**) do

{

Choose **Q** in **D** that is not in **BCNF**;

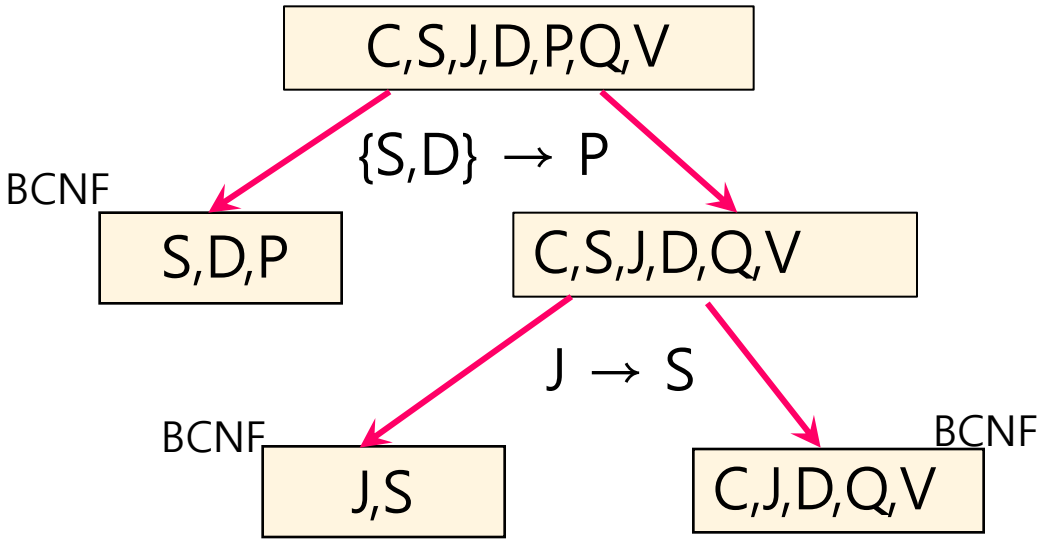
Find a FD **X** → **Y** in Q that violates **BCNF**;

Replace Q by **{X ∪ Y}** and **{Q – Y}**;

}

- 위의 분해는 **BCNF**와 **lossless join** 을 모두 만족함

Lossless BCNF로의 분해 : 예



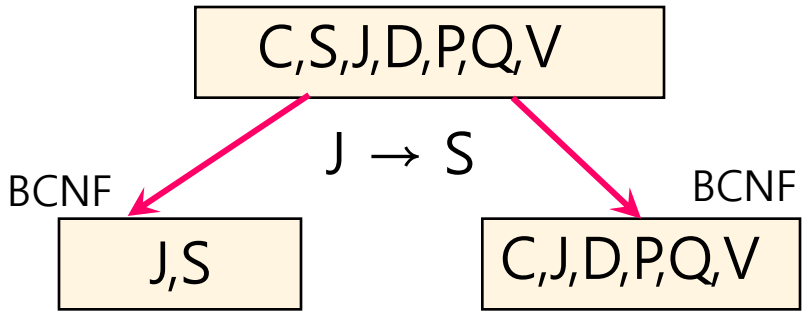
R = {C,S,J,D,P,Q,V}

Keys = {C}, {J,P}

F : 1) {J,P} → C
2) {S,D} → P
3) J → S

- {S,D} → P 와 J → S 가 BCNF 위반;
- 이들 중 {S,D} → P 를 선택; {S,D,P}와 {C,S,J,D,Q,V} 로 분해
- {S,D,P}는 BCNF; {C,S,J,D,Q,V}는 아직 아님.
- 다음으로 J → S 를 선택; {J, S}와 {C,J,D,Q,V} 로 분해
- {J,S}는 BCNF; {C,J,D,Q,V}도 역시 BCNF
  (그 이유는 C → {J,D,Q,V})
- 위의 분해는 역시 모두 lossless join을 만족함. 확인!

Lossless BCNF로의 분해 : 예 (다른 방법)



R = {C,S,J,D,P,Q,V}

Key = {C}, {J,P}

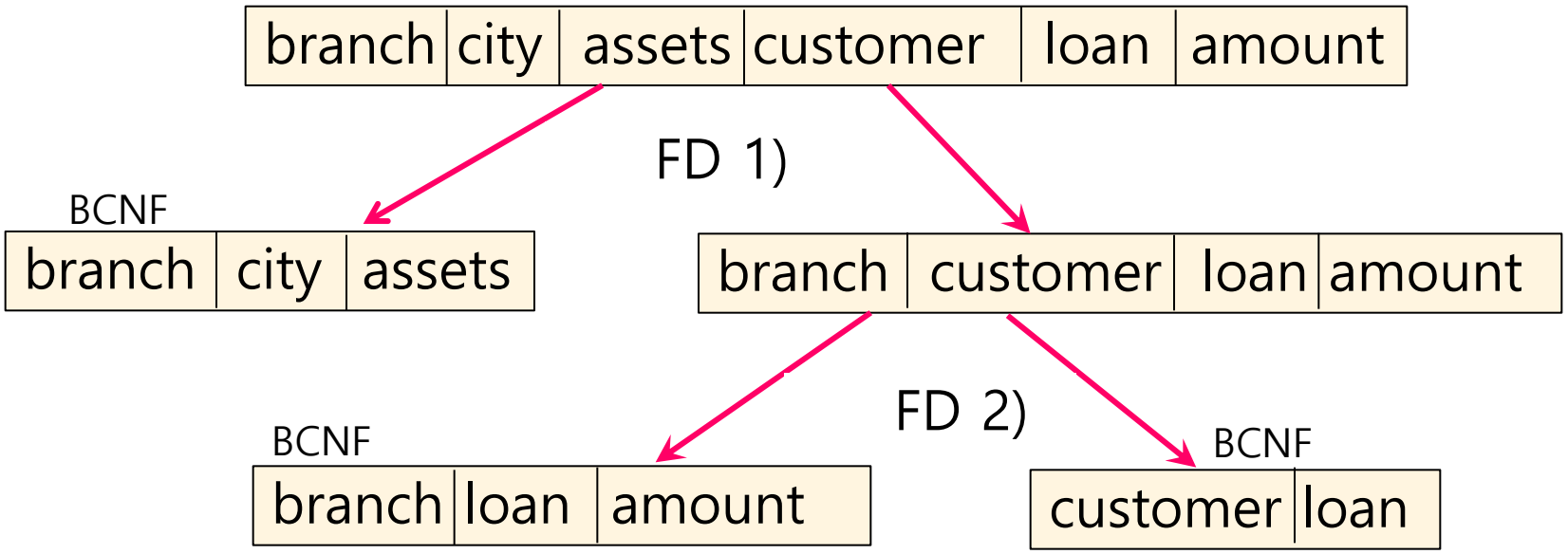
F : 1) {J,P} → C
2) {S,D} → P
3) J → S

- {S,D} → P 와 J → S 가 BCNF 위반.
- 이번에는 J → S 를 선택; {J,S}와 {C,J,D,P,Q,V} 로 분해
- {J,S}는 BCNF; {C,J,D,P,Q,V} 도 역시 BCNF;
  (그 이유는 {J,P} 도 역시 key 이기 때문)
- 위의 분해는 역시 모두 lossless join을 만족함. 확인!

- BCNF를 위반하는 FD 들 중에서 어떤 것을 선택하냐에 따라, 서로 다른 BCNF들이 생성될 수 있음.

Lossless BCNF로의 분해 : 예

F : 1) branch → {city, assets}
2) loan → {amount, branch}
Key = {loan, customer}



Lossless/ FD 보존 BCNF로의 분해 : 예

- R = (A, B, C)
  F = 1) $A \rightarrow B$
  2) $B \rightarrow C$
  Key = {A}

- R is not in BCNF

- Decomposition R_1 = (A, B), R_2 = (B, C)
  - R_1 and R_2 in BCNF
  - Lossless-join decomposition
  - Dependency preserving

Problems : BCNF는 FD 보존 보장 못 함

ADDRESS

street	city	zip
16th	NY	23508
17th	NY	23509

F: 1) {street, city} → zip
2) zip → city

- zip → city 가 BCNF 위반; 따라서 다음과 같이 분해;

street	zip
16th	23508
17th	23509

city	zip
NY	23508
NY	23509

- {street, city} → zip 이 보존이 안 됨.

Comparison: 3NF vs. BCNF

- It is possible to decompose a relation into BCNF such that:
  - Lossless Join
  - No redundancy

- It is possible to decompose a relation into 3NF such that:
  - Lossless Join
  - Dependency Preservation

- Goal for a relational database design is:
  - BCNF
  - Lossless join.
  - Dependency preservation.

- If we can not achieve this, we accept one of
  - Lack of dependency preservation
  - Redundancy (due to use of 3NF)

Normalization : Exercise

- Consider the following CAR relation;

CAR

model	cylinder	origin	tax	fee

F: 1) {model, cylinder} → origin
2) {model, cylinder} → tax
3) cylinder → fee
4) origin → tax

- Decompose the above CAR relation with
  - no redundancy
  - lossless join
  - no insert anomaly
  - no delete anomaly
  - no update anomaly

Multi-Valued Dependency (MVD)

- 많은 경우에 FD로 표현이 어려운 제약조건이 존재함; 이는 **Multi-Valued Dependency (MVD)**로 표현함.

- Relation **R**에서 **X**의 값에 대해 **Y**의 값이 여러 개 있을 경우, 이를 **MVD** : **X** →→ **Y** 로 표기함. (단, X, Y : attribute(s))

- 즉, MVD : **X** →→ **Y** 는 FD : **X** → **Y** 를 확장한 것

- 예 : PERSON(name, addr, phone, hobby)

  (1) A person has many phones; **name** →→ **phone**

  (2) A person has many hobbies; **name** →→ **hobby**

MVD : 예

- 한 relation에 여러 개의 multi-valued attribute들이 존재;
- A person's phones are independent of the hobby.
- In this case, each phones must appear with each of the hobbies in all combinations.

MVD : 1) **name →→ phone**
2) **name →→ hobby**

PERSON

name	addr	phone	hobby
eve	a	p1	h1
eve	a	p2	h2
eve	a	p2	h1
eve	a	p1	h2

If these tuples are here,

then another tuples must also be in the relation

Property : MVD

- Let **X**, **Y**, **Z** : set of attributes in relation **R**;

- 만약 **X→→Y** 이 성립하면, **X→→Z** 가 역시 성립
  (즉, 양쪽 MVD가 서로 대칭으로 항상 성립함)

- **Y** 와 **Z**의 값들은 서로 독립적으로 무관한 관계임.

- **R**에서 MVD : **X →→ Y** 가 다음 조건 중 하나를 만족하면 **Trivial** MVD 라 함. 이 외는 **Non-Trivial** 이라 함.
  1) **Y ⊆ X**
  2) **X ∪ Y = R**

- MVD가 Trivial인 경우는 (자연스럽게) 문제가 발생 하지 않음
  : 만약 **Non-Trivial MVD** 인 경우는?

Problems : MVD

SELLS

product	company	country
A	LG	USA
A	LG	UK
A	LG	Japan
B	LG	USA
B	LG	UK
B	LG	Japan
A	Samsung	USA
. . .	. . .	. . .

MVD :

1) company →→ product
2) company →→ country

- 위의 MVD는 **Non-Trivial** 인가?
- 위의 relation의 **Key**는?
- 위의 relation은 **BCNF**를 만족하는가?
- 위의 relation은 **FD**로 인한 **Redundancy** 문제가 발생하는가?
  만약 발생 안 한다면, 또 다른 문제들은?

Non-Trivial MVD : 예

- 어떤 Relation **R**에 **Non-Trivial MVD** 가 존재하는 경우;
  - **R**의 **key**는 (일반적으로) "**All-Attributes**" 가 됨.
  - **R**에는 어떠한 **FD**도 발생하지 않음.
  - 따라서 **BCNF**를 만족하여 **FD**로 인한 중복은 발생하지 않음.
  - 그러나 **BCNF**로 해결할 수 없는 또 다른 문제점들이 발생함.

- 앞의 SELLS relation에서 다음 문제들이 발생함; Explain!
  - Redundancy
  - Insert Anomaly
  - Delete Anomaly
  - Update Anomaly

Fourth Normal Form (4NF)

- A relational R is **4NF** if

  1) **X** →→ **Y** is **trivial** or

  2) for every **non-trivial MVD**: **X** →→ **Y**, **X** is a **super-key**.

- 'SELL' relation is not in 4NF; Why?

  1) company →→ product

  2) company →→ country

  – These are non-trivial MVDs and 'company' is not a super key.

- If **X** →→ **Y** in R violates 4NF, we decompose **R** using the same technique as for BCNF. (See next page)

Decomposition into 4NF

- **Lossless 4NF Decomposition Algorithm**

Input : A relation **R** and a set of MVDs **F**

Output : **D** = {$\mathbf{R_1}$, $\mathbf{R_2}$, . . ., $\mathbf{R_m}$} where each $\mathbf{R_i}$: **4NF**

Let **D** = {**R**};

While (there is a relation **Q** in **D** that is not in **4NF**) do

{

Choose **Q** in **D** that is not in **4NF**;

Find a FD **X** →→ **Y** in Q that violates **4NF**;

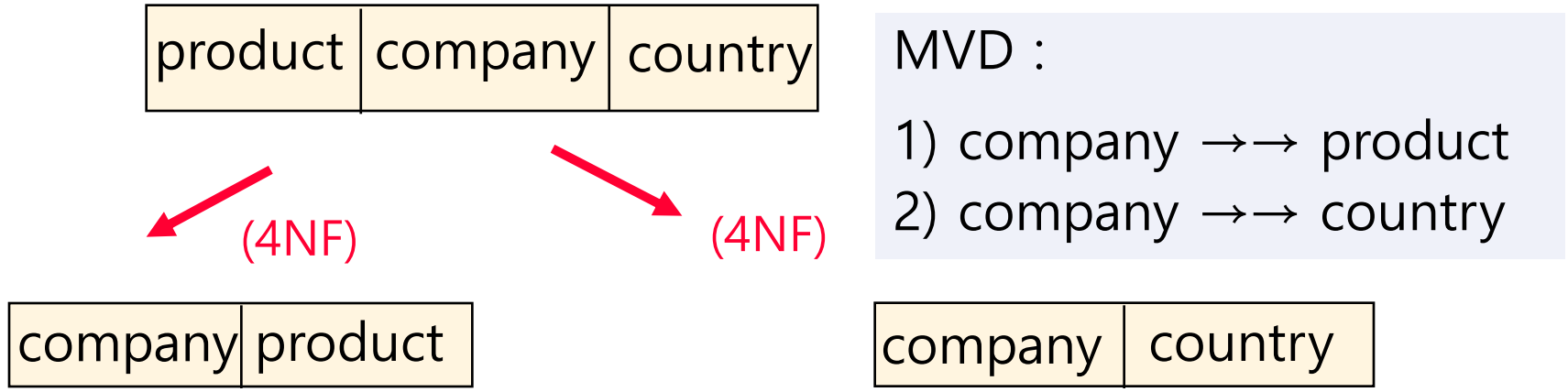
Replace **Q** by **{X ∪ Y}** and **{Q – Y}**;

}

- 위의 분해는 **4NF**와 **lossless join** 을 모두 만족함

Decomposition into 4NF

- We decompose SELL relation as follows;



MVD :

1) company →→ product
2) company →→ country

- 위의 분해된 relation에서 다음 문제들이 발생하지 않음; 설명!
  - Redundancy
  - Insert Anomaly
  - Delete Anomaly
  - Update Anomaly

4NF : Exercise

- Consider the relation 'BASEBALL' and MVDs;

players	team	couches

MVD :

1) team →→ players
2) team →→ couches

1) 위의 relation은 BCNF인가? 4NF인가?

2) 위의 relation에서 발생하는 다음 문제점들을 설명!

- Redundancy
- Insert Anomaly
- Delete Anomaly
- Update Anomaly

3) 위의 문제점들이 발생하지 않도록 분해하라.

4NF : Exercise

- Consider the following relation;

  Person (<u>name, phone, hobby,</u> addr)

  FD : name → addr

  MVD : 1) name →→ phone

  2) name →→ hobby

  Key = {name, phone, hobby}

- We decompose 'Person' using the above FD;

  Person1 (<u>name</u>, addr)

  Person2 (<u>name, phone, hobby</u>)

- We decompose 'Person2' using the above MVD;

  Person3 (<u>name, phone</u>)

  Person4 (<u>name, hobby</u>)

Comparisons : Normal Forms

문제점	2NF	3NF	BCNF	4NF
FD로 인한 중복 소거?	some	most	yes	yes
MVD로 인한 중복 소거?	no	no	no	yes
FD 보존하는 분해?	yes	yes	maybe	maybe
MVD 보존하는 분해?	maybe	maybe	maybe	maybe
Lossless Join 분해?	yes	yes	yes	yes

De-Normalization

- **De-Normalization** is a process of composing(merging) relations to intentionally increase redundancy.

- Advantages of De-normalization
  - Speed up by minimizing the number of joins
  - Precompute aggregate values
  - Reduce the number of tables (in some cases)

- De-normalization Techniques
  - Setting weaker normal forms
  - Adding Columns
  - Collapsing Tables
  - Duplicating Tables