5&6번째 스카마 정제와 정규형(Nomal form) RDBMS

16시간 전

https://blog.naver.com/lsm980515/222690102879

Normal form

- RDBMS에서 **데이터 중복이 받생하면 릴레이션에 대한 데이터 삽입, 수정, 삭제 연산 등은 수행할 때** 부작용이 발생할 수 있는데 이러한 **부작용**을 **이상현상(anomaly)**이라 함.
anomaly현상은 제거하면서 RDBMS을 올바르게 설계해 나가는 과정이 **정규화(Nomalization)이다.**
- ****즉. RDBMS에서 중복은 최소화하게 데이터를 구조화하는 프로세스를 정규화라고 함.****
- 정규화 된 정도를 정규형(Normal Form)으로 표현하며, FD에 기반한 정규형들로 1NF, 2NF. 3NF, BCNF 이 있고, 그 외에도 4NF,5NF, 6NF까지 있다.
- **3NF, BCNF는 함수 종속성(functional dependencies)을 이용하여 이상현상을 방지함. ****
- 정규형들은 점차적으로 뒤의 정규형으로 갈수록 제약조건들이 많아짐.
- BCNF에 속하는 모든 relation들은 역시 3NF에 속하고, 3NF에 속하는 모든 relation 역시 2NF에 속한다.

/entry.naver?docId=3431240&cid=58430&categoryId=58430
https://ko.wikipedia.org/wiki/%EB%8D%B0%EC%9D%B4%ED%84%B0%EB%B2%A0%EC%9D%B4%EC%8A%A4_%EC%A0%95%EA%B7%9C%ED%99%94

제1정규형(First normal form: 1NF)

-**1NF**의 은 "**어떤 relation에서 모든 각 attribute에 원자 값(atomic value)만 반드시 허용이 되는 경우** 이다.
- 제1정규화의 문제점은 중복된 값을 많이 가질 수 있는데 RUDI 문제가 발생해도 대처할 수 없음.

Student	Courses	id
Mary	{CS145, CS229}	1593

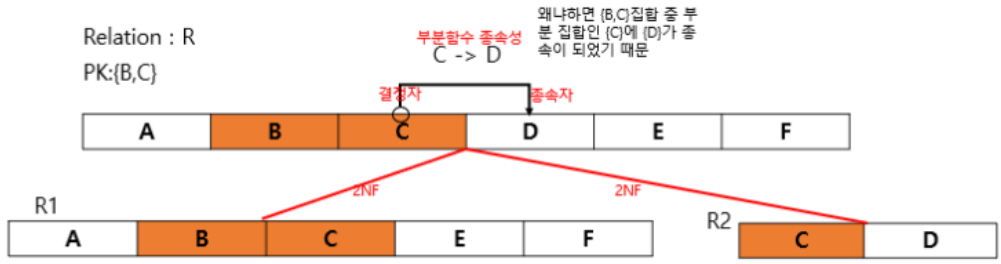
- Courses value값이 한칸에 두개가 있기 때문에 원자성을 충족시키지 못했기 때문에 이 relation은 제1정규형을 만족하지 안늪다.
- 만족할려면 Courses value에 값이 한개여야 한다. e.g. CS145

Student	Courses	id
Mary	CS145	1593
Mary	CS229	1593

지금 같은 경우가 모든 value가 원자 값을 가지기 때문에 제1정규형을 만족했다라고 말할 수 있으며, 제 1정규형을 만족해야 relation이 될 자격이 있다.

제2정규형(Second normal form : 2NF)

- Relation이 1NF에 속하고, Primary key가 아닌 모든 attribute가 기본키에 완전 함수 종속되면 2NF이다.
- 1NF에 속하는 relatino이 2NF을 만족하게 하려면, 부분 함수 종속은 제거하고 모든 속성이 Primary key 에 완전 함수 종속되도록 relation은 분해하는 정규화 과정을 거쳐야 함.
- 정규화 과정에서 relation을 분해할 때 주의할 점은, 분해된 relation들은 natural join하여 분해 전의 relation으로 다시 복원 가능해야 됨. 즉, 정규화 과정 중에 수행되는 relation의 분해는 무손실 분해여야 함. **



위 그림에서 이 부분함수 종속을 제거하는 것이 2NF라고 한다. !

※natural join
▷natural join하게 되면 동일한 column명이 합쳐짐

완전함수 종속성

▷ 하나의 relation을 구성하는 attribute들의 부분 집합을 X와 Y라 할 때, 어느 시점에서든 Relation 내의 모든 tuple은 대상으로 한 X값(결정자)에 대한 Y(종속자)각각의 attribute의 값이 항상 하나면 "Y가 X에 함수적으로 종속되어 있다" or "X가 Y를 함수적으로 결정한다.
즉 X attribute의 값에 대응되는 Y 각각의 attribute의 값이 단 하나여야 된다.
함수 종속 관계는 X -> Y로 표현, X를 결정자, Y는 종속자라고 함.
유일하게 결정하는 attribute가 결정자 이다. 무조건 primary key랑 candiate key만 되는건 아님.
즉. 속성 Y값을 유일하게 결정하는 속성 X는 함수 종속 관계에서 모두 결정자가 될 수 있다.

Student_id	event_num	name	당첨여부
aaa1	E001	김	Y
aaa1	E002	김	Y
aaa1	E005	김	N
bbb1	E010	최	Y
bbb1	E002	최	N
ccc1	E006	모	Y
ccc2	E007	모	Y

교내 이벤트참여 relaion

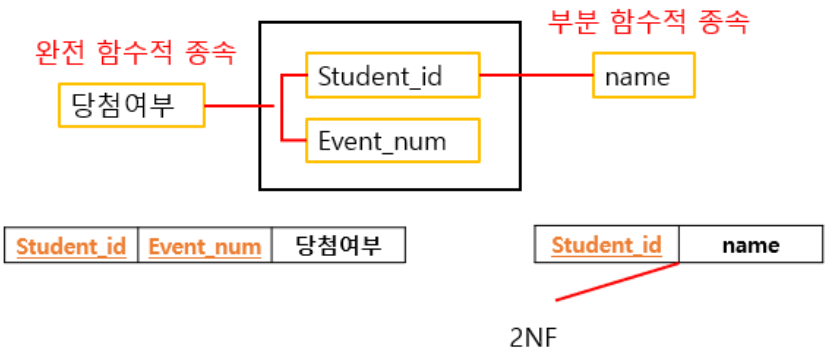
여기서는 결정자 X = {student_id, evnet_num}는 속성 집합 이다.
왜냐하면 이 X가 모든 tuple을 구분할 수 있기 때문
따라서 X{student_id, event_num} -> Y{name, 당첨여부} or
X{student_id, event_num} -> Y{name} and
X{student_id, event_num} -> Y{당첨여부} 로 나타낼 수 있음.

또 student_id가 학생name을 유일하게 결정하므로 X = {student_id} -> Y = {name} 도 된다. 왜냐하면 student_id가 같으면 모든 tuple에서 name이 반드시 같은 값을 가지기 때문이다.

따라서 위 relatino에서 존재하는 함수 종속 관계

X (결정자) -> Y (종속자)

1. student_id -> name
2. {student_id, event_num} -> 당첨여부
3. {student_id, event_num} -> name



여기서 Y = {당첨여부}는 X={student_id, event_num}의 일부분이 아닌 전체에 종속되어 있는데 이런 경우는 당첨여부 attribute가 {student_id, event_num} 속성 집합에 완전 종속되었다고 함.

반대로

여기서 Y={name}는 X={studnet_id}에도 종속되어 있고, X = {student_id, event_num}에도 종속되 어 있다.

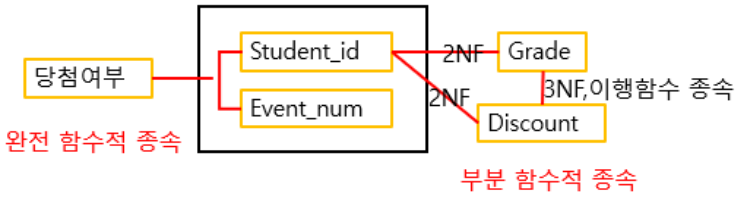
이런 경우는 name attribute가 {student_id, event_num} attribute 집합에 부분 함수 종속되었다고 함.

https://terms.naver.com/entry.naver?docId=3431246&cid=58430&categoryId=58430

Student_id	event_num	grade	당첨여부	discount
aaa1	E001	vip	Y	20%
aaa1	E002	vip	Y	20%
aaa1	E005	vip	N	20%
bbb1	E010	gold	Y	10%
bbb1	E002	gold	N	10%
ccc1	E006	silver	Y	5%
ccc2	E007	silver	Y	5%

이 relation을 2차 정규화 하기 위해서는

Relation 내의 모든 tuple은 대상으로 한 X값(결정자)에 대한 Y(종속자)각각의 attribute의 값이 항상 하나인 부분을 골라 내야됨.



그 결과 X{student_id}일때 Y{grade} , 그리고 X{student_id}일때 Y{discount}값이 항상 하나 이다.

즉 student_id -> grade, student_id -> Discount 이고 두개의 부분 함수적 종속성을 가지고 있다.
이를 제거해 주는것이 제2 정규화이다.
이행함수 종속성을 제거해주는것이 3NF이다. <- 아래 그림은 2NF까지만 진행했음.
따라서 부분함수적 종속성을 합친 student_id, grade, Discount를 속성으로 가지는 Relation1을 만들다.
또 X{student_id, evnet_num}일때 Y{당첨여부}가 항상 하나로 해당 tuple은 항상 유일하다. <- Relation2
e.f. {aaa1, E001, Y}, {aaa1, E001, Y} ...

R1

Student id	Grade	Discount
aaa1	vip	20%
bbb1	gold	10%
ccc1	silver	5%

R2

Student id	Event num	당첨여부
aaa1	vip	Y
aaa1	vip	Y
aaa1	vip	N
bbb1	gold	Y
bbb1	gold	N
ccc1	silver	Y
ccc1	silver	Y

relation이 함수 종속성(Y)을 하나만 포함하면 이상현상 발생하지 않지만,
Y(grade,당첨여부)와 같이 두 개이상이 포함이 된다면 이상 현상이 발생할 수 있다 **

이러한 중복성으로 야기되는 문제점

1. 중복 저장(Redundant Storage)
   ▷ 어떤 정보가 반복적으로 저장됨.
   ▷ 여기서는 grade값 vip는 dicount 값 20%과 대응라는데, 이 연관성이 세 번 반복 됨.
   ▷ vip 와 할인율은 항상 일치할 수 밖에 없는 특징을 가졌고, 특정 기간동안 할인율을 올린다고 했을때 바꿀려는 row수가 많아지게 되서 일부가 갱신이 안되는 경우가 발생할 수 있어서 정확도에 문제가 생김.
   ▷ 따라서 갱신하게 되는 row수를 줄이는 것이 중요함.

2. 갱신 이상(Update Anomaly):
   ▷ 만약 반복되어 저장된 데이터 중 한 사본만을 갱신하는 경우, 모든 사본이 함께 갱신되지 않으면 데이터간의 불일치가 발생된다.
   ▷ 위의 relation을 예시로 특정 id의 vip 할인율을 변경하게 되면, 어느 값이 진짜 값인지 알기 어렵게 된다.

3. 삽입 이상(Insertion Anomaly):
   ▷ 어떤 새로운 정보를 저장하기 위해 이와 관련이 없는 다든 정보도 함께 저장하여야 한다.
   ▷ 새로운 grade인 bronze 등급이 생기는 경우 이 등급에 해당하는 id가 없어 해당 attribute 값이 NULL이 되므로 삽일할 수 없다.

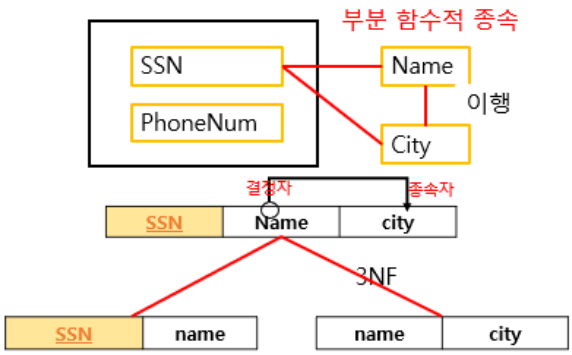
4. 삭제 이상(Deletion Anomaly):
   ▷ 만약 어떤 정보를 삭제하는 경우, 이와 관련이 없는 다든 정보도 함께 상실될 수가 있다.
   ▷ vip등급이 사라진다면 해당 등급의 discount가 얼마가 되는지 정보를 잃게 된다. 즉, 연관성을 상실

▷ 테이블에서 찾은 FD중 RUDI 문제가 있는 FD듣은 해당 테이블에서 분리하여 독립된 테이블로 만들어 줘야 된다.

https://blog.naver.com/seungp916/220067635834
https://yaboong.github.io/database/2018/03/09/database-normalization-1/
https://terms.naver.com/entry.naver?docId=3431250&cid=58430&categoryId=58430

제 3 정규형(Third Normal Form:3NF)***

- 제 2NF에 속하면서, 기본키가 아닌 모든 attribute들이 기본키에 이행적 함수 종속이 되지 않으면 제3정규형 이다.
- 3NF로 릴레이션을 분해하더라도 해당 릴레이션들 중에 기본키가 될 수 있는 후보키가 여러개 라면 이상현상이 발생할 수 있는데 이를 해결하는 것이 BCNF이다.
- 모든 BCNF 릴레이션은 3NF에 속한다.
- 2NF에서 이행함수 종속성을 제거하면 3NF를 만족하게 된다.
- 기본키 집합에 종속되지 않고, 일반 속성에 종속되는 속성을 분해 한다.



위의 table(ssn,name,city)은 3NF 대상 릴레이션 이므로 3NF를 하게 되면
결정자와 종속자끼리 table1, 그리고 결정자와 나머지 attribute를 해서 table2를 구성하게 되면
3정규화된 것이다.

함수 종속성(functional dependency: FD)

▷ 함수 종속성은 일종의 무결성 제약조건(integrity constranit:IC)*으로서, 키의 개념을 일반화한 것이다.
▷ FD는 X -> Y와 같은 형식으로 표현됨.
▷ 좋은 FD는 X -> Y와 같은 형식에서 X가 super key인 경우가 좋은 FD이다. 나머지 경우는 Bad FD
▷ X - > Y에서 Y는 모든 attribute들의 집하이며, X의 진부분 집합 V가 존재하여 V ->Y가 만족된다면, X는 슈퍼키 이다.
▷ 즉 Y ⊆ X이고 [V ⊂ X, V≠X]를 만족하는 X의 진부분 집합V가 존재하여 V -> Y가 만족된다면 X는 super key* 이다.

※integrity constranit
1. entity 무결성 제약조건
▷ 어떠한 primary key 값도 null이 될 수 없다.
2. reference 무결성 제약 조건
▷ 한 relation에 있는 attribute가 다든 relation의 attribute을 참조하려면 반드시 참조되는 attribute의 value가 그 relation 내에 있어야 한다.

※super key ***
▷ 슈퍼키는 A,B,C,D,E 5개의 attribute가 있을때 A,B가 중복값이 없고, null값이 없어 유일함을 만족할때 A,B 두개가 각각의 키가 될 수 있고, 또 여러 attribute를 합한 A+B+E, B+E 등 도 슈퍼키가 될수 있다.
▷이렇게 테이블에서 각 행은 유일하게 식별할 수 있는 하나 or 그 이상의 attribute의 집합이다.
▷ 이런 super key중 가장 minial한 key를 key라고 부른다.

A	B	C	D
a1	b1	c1	d1

a1	b1	c1	d2
a1	b2	c2	d1
a2	b1	c3	d1

위 테이블은 AB -> C 라는 함수 종속성을 만족하는 인스턴스(instence)를 보여줌.
즉. a1,b1 일때 C의 value는 무조건 c1인것 처럼 위의 함수 종속성을 만족함.
하지만 마지막 행에 <a1,b1,c2,d1>을 추가하게 되면 위의 FD를 위반하게 된다.
왜냐하면 a1,b1일때 C는 c1인데 c2가 나오게 되면서 무결성 제약조건을 만족하지 못함.

ACTIVITY

A	B	C	D	E
1	2	4	3	6
3	2	5	1	8
1	4	4	5	7
1	2	4	3	6
3	2	5	1	8

Find at least *three* FDs which are violated on this instance:

{ } → { }
{ } → { }
{ } → { }

여기서 FD는
{B,C} -> {D},
{C,D} -> {E},
{A,B} -> {C,D,E}

Example:

Products

Name	Color	Category	Dep	Price
Gizmo	Green	Gadget	Toys	49
Widget	Black	Gadget	Toys	59
Gizmo	Green	Whatsit	Garden	99

Provided FDs:
1. {Name} → {Color}
2. {Category} → {Department}
3. {Color, Category} → {Price}

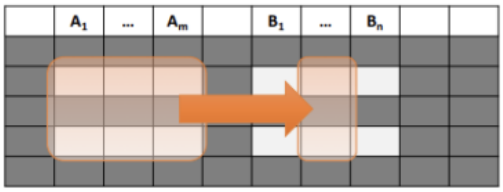
우리는 위 식에 주어진 FD들을 보고,
{Name,Category} -> {Price} 관계가 성립함을 알 수 있다.

FD를 찾아야 하는 이유

1. 테이블에서 RUID를 제거하기 위해
2. 남아 있는 FD가 해당 테이블의 key가 되도록 하기 위해.

Armstrong's Rules

1.Split /Combine



Combine $A_1, \ldots, A_m \rightarrow B_1,\ldots,B_n$

Split $A_1,\ldots,A_m \rightarrow B_i$ for i=1,...,n

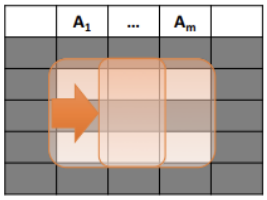
▷ Split은X $\{A_1,..,A_m\}$ -> Y $\{B_1,..,B_m\}$ 관계가 성립 했을 때

- X $\{A_1,..,A_m\}$ -> Y $\{B_i\}$ for i = 1,....,n 관계가 성립

▷ Combine은 X $\{A_1,..,A_m\}$ -> Y $\{B_i\}$ for i = 1,....,n 관계가 성립 했을 때

- X $\{A_1,..,A_m\}$ -> Y $\{B_1,..,B_m\}$ 관계가 성립

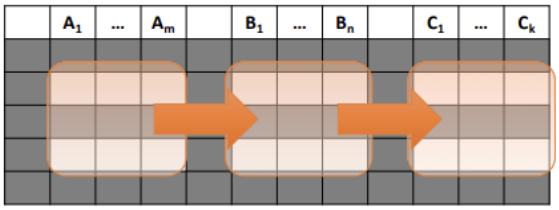
2. Reduction / Trivial



▷ X $\{A_1,..,A_m\}$ 있을 때

- X $\{A_1,..,A_m\}$ -> Y $\{A_j\}$ for j = 1,....,n 관계가 성 립 한다는 뜻

3. Transitive Closure



▷ X $\{A_1,..,A_m\}$ -> Y$\{B_1,..,B_m\}$ 관계가 성립하고 Y $\{B_1,..,B_m\}$ -> Z$\{C_1,..,C_m\}$ 관계가 성립할 때

- X $\{A_1,..,A_m\}$ -> Z$\{C_1,..,C_m\}$가 성립한다.

문제를 한번 풀어보자

Inferred FDs:

Inferred FD	Rule used
4. {Name, Category} -> {Name}	?
5. {Name, Category} -> {Color}	?
6. {Name, Category} -> {Category}	?
7. {Name, Category -> {Color, Category}	?
8. {Name, Category} -> {Price}	?

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Dept.}
3. {Color, Category} → {Price}

4. {Name, Category} -> {Name} 관계는 X $\{A_1,..,A_m\}$ -> Y $\{A_j\}$ for j = 1,....,n과 같으므로 Trivial 이다.
5. {Name, Category} -> {Color} 관계는 {Name} -> {Color} 관계가 성립하므로 X{Name, Category} -> Y{color,Category} 관계가 성립하고 Y {Color, Category} -> Z{Color} 성립하므로 X{Name, Category} -> Z{Color} 와 같으므로 Transitive이다.
6. {Name, Category} -> {Category} 관계는 X {Name, Category} -> Y {Category}와 같으므로

Trivial 이다.

7. {Name, Category} -> {Color, Category} 관계는 {Name, Category} -> {Color} 관계도 성립하고, {Name, Category} -> {Category} 관계도 성립하므로 이 두 관계의 결합은 {Name, Category} -> {Color, Category}으로 동등하므로 Combine 이다.
8. {Name, Category} -> {Price} 관계는 주어진 {Color, Category} -> {Price}관계로부터
   X{Name, Category} -> Y {Color, Category} -> Z{Price} 가 되며
   이는 X{Name, Category} -> Z{Price} 와 같으므로 Transitive 이다.

Closure(페포), 종속성 추돈 규칙(Inference Rules)***

▷ FD듣의 집합 F에 의해 추돈되는 모든 FD듣의 집합은 F 의 Closure라고 하며, 이를 F^+로 표기함.

▷ Armstrong's Axioms(암스트롱의 공리)라고 불리는 세 가지 법칙들을 반복 적용하면 FD들의 집합 F에 의해 내포되는 모든 FD들을 추론해 낼 수 있다.

▷ Closure를 통해 해당 table에서 key로 쓸 수 있는 속성을 알아 낼 수 있다.

IR1(reflexive rule:재귀) X ⊇ Y이면 X → Y이다. - 암스트롱의 공리1
IR2(augmentatino rule:부가) X → Y이면, XZ → YZ이다. - 암스트롱의 공리2
IR3(transitive rule:이행) X → Y이고 Y → Z이면, X → Z 이다. - 암스트롱의 공리3
IR4(Union:결합) X → Y이고 X → Z이면, X → YZ 이다.
IR5(Decomposition:분해) X → YZ이면, X → Y이고 X → Z 이다.
IR6(pseudo transitive rule:유사 이행) X → Y이고 WY → Z이면, WX → Z 이다.

https://blog.naver.com/parkdntir/221983132898

▷attribute의집합 $\{A_1,..,A_n\}$ 과 FD 집합 $F=\{f_1,..,f_n\}$이 주어 졌을때 Closure,$\{A_1,..,A_n\}^+$는 attribute의 집합 B 이다. 즉, $\{A_1,..,A_n\}$ -> B

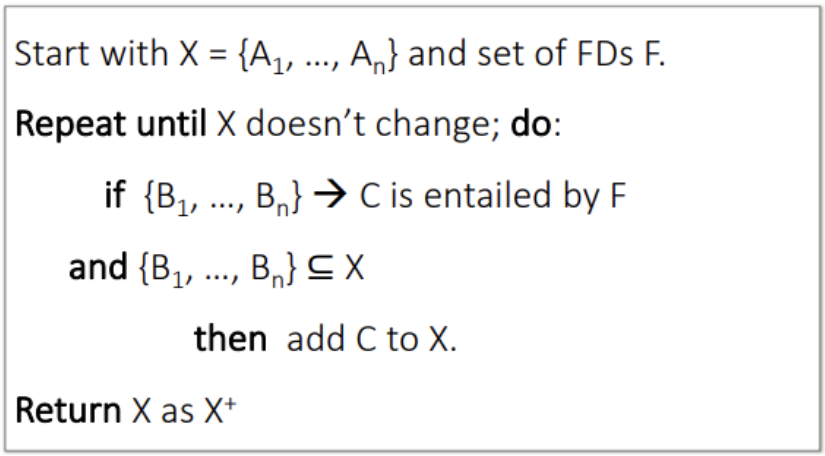
위와 같이 FD의 조건인 F의 집합이 위와 같을 때

$\{name\}^+$는 자기자신 Name과, FD의 조건에서 나온 Color를 가진다.

$\{name, category\}^+$는 자기자신 Name, category과 FD의 조건에서 {name} →{color}, {category}->{dept}이고, 여기서 얻어진 color와 category를 집합으로 해서 price값을 종속시켜

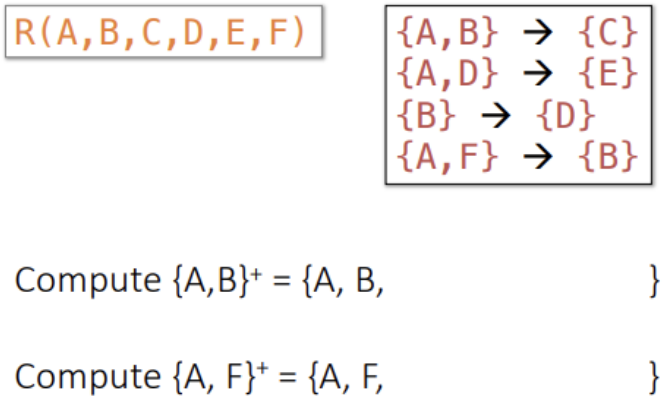
최종적으로 $\{name, category\}^+$ = {name, category, color, dept, price}이다.

$\{color\}^+$는 자기자신 color만 가짐 name에 종속되어 있는 FD조건이 있지만 종속일뿐 결속자가 아니기 때문이다.



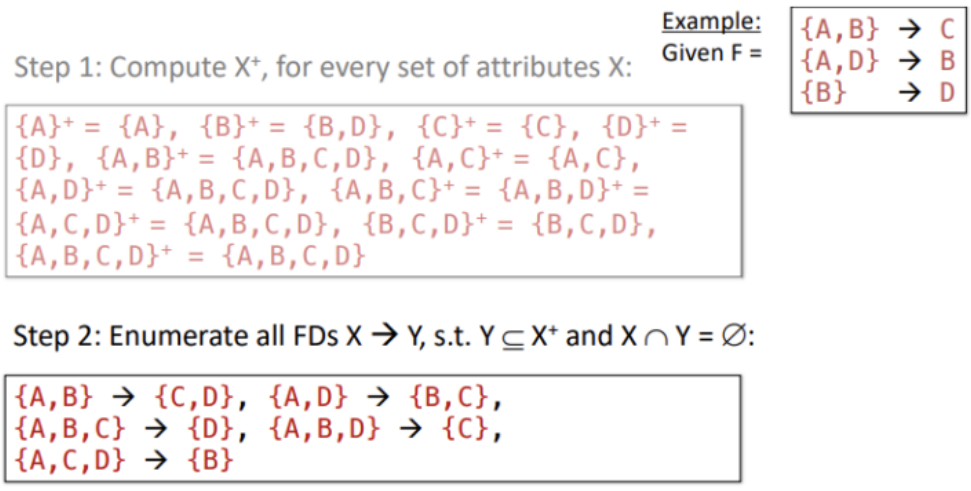
Closure Algorithm

위와 같은 Closure 알고리즘을 보면
Attribute의 집합 X와 FD의 집합 F가 있을때 X가 변하지 않을때 까지 반복해서 아래와 같은 식을 F를 기준으로 검사함. 해당 식의 조건이 참이면 집합 C의 값을 X에 더한다.



예제

A,B의 closure인 $\{A,B\}^+$ 는 자기자신 A,B를 가지고, FD의 조건을 보면 $\{C\} \subseteq \{A,B\}$이므로 이므로 C를 가지고, $\{D\} \subseteq \{B\}$ 이고, $\{E\} \subseteq \{A,D\}$이므로
$\{A,B\}^+ = \{A,B,C,D,E\}$
A,F의 closure 인$\{A,F\}^+$는 자기자신 A,F를 가지고, FD의 조건을 보면 $\{B\} \subseteq \{A,F\}$이므로 B를 가지고
또 $\{D\} \subseteq \{B\}$이므로 D를 가지고 또 $\{C\} \subseteq \{A,B\}$와 $\{E\} \subseteq \{A,D\}$의 조건을 가지므로 C,E를 가진다.
따라서 $\{A,F\}^+ = \{A,B,C,D,E,F\}$



step1은 모든X의 attribute의 집합을 closure한 것인데 이렇게 까지 할 필요는 없고

step 2은 모든 FD의 조건이 X → Y일때 $Y \subseteq X^+$와 $X \cap Y = \emptyset$을 만족하는 것을 나열해라
즉 closure에 포함이 되는 것들중 자기자신을 빼고 나열을 해도 된다.
e.f. FD의 조건중 X={A,B} → Y={C}를 보면 {C} ⊆ $\{A,B\}^+$이고 {A,B} ∩ {C} = ∅을 만족하는 것을 나열하는데
우리는 step1에서 $\{A,B\}^+$ → {A,B,C,D}인 관계를 만족하는 것을 알기 때문에
자기자신을 빼게 되면 {A,B} -> {C,D}이 나온다.

또 다른 예제로 스키마 Contracts(contractid, supplierid, projectid, deptid, partid, qty, value)가 있고
이 스키마에 속한 투플의 의미는 "어떤 공급자 S가 부품 P를 Q수량만큼, 어떤 부서D가 수행하는 프로젝트J에게 공급하기로 한 계약번호가 C인 계약을 체결하였으며, 이 계약의 금액 V는 가격 attribute value와 같다.
Contracts의 릴레이션에서 다음의 FD들이 성립된다고 가정하자
1. C → CSJDPQV. 계약번호 C는 key이다.
2. JP → C. 한 프로젝트에서 하나의 부품을 구매하는 것은 단 하나의 계약을 통해서만 이루어진다.
3. SD → P. 각 부서는 각 공급자로부터 최대 하나의 부품만 구매할 수 있다.

이렇게 주어진 FD들의 집합에 대한 closure에는 다음과 같은 FD들이 추가적으로 추론될 수 있다.
▷ JP → C와 C → CSJDPQV로부터, 이행 법칙에 의해, JP → CSJDPQV가 추론됨.
▷ SD → P로부터, 부가 법칙에 의해 SDJ → JP가 추론 됨.
▷ SDJ → JP와 JP → CSJDPQV로부터, 이행 법칙에 의해 SDJ → CSJDPQV가 추론됨.
▷C → CSJDPQV로부터, 분배 법칙을 이용하면 C → C, C → S, C → J,.....들도 추론 가능하다.

Super key와 key를 찾는 방법 **

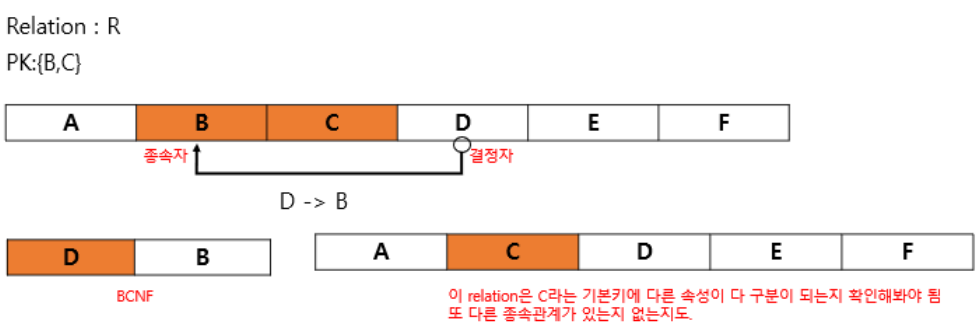
Attribute의 집합 X = $\{A_1,..,A_n\}$
1. Compute X^+
2. X^+ 중에서 모든 attribute의 집합들을 구분짓을 수 있는 X^+가 있다면 그 X가 super key가 됨
3. 그중 X가 가장 짧은 것이 key가 됨.

```
Product(name, price, category, color)
```

```
{name, category} → price
{category} → color
```

여기서 FD의 조건으로 부터 {naem, category} → {price}와 {category} → {color}이므로
$\{naem, category\}^+$= {naem, category, price, color}로 모든 attribute를 나타 낼 수 있으므로

보이스-코드 정규형(Boyce-code normal from: BCNF)*****



결정자가 candidate key가 아니면서 종속자가 기본키의 or 기본키의 부분집합일 경우 BCNF라고 함.

▷ 릴레이션의 함수 종속 관계에서 "모든 결정자"가 candidate key이면 BCNF에 속한다.

▷ 후보키를 여러 개 가지고 있는 릴레이션에 발생할 수 있는 이상 현상을 해결하기 위해 제 3정규형보다 엄격한 제약조건을 가짐

▷ 보이스-코드 정규형에 속하는 모든 릴레이션은 제 3정규형에 속함. 하지만 제 3정규형이라고 모두 BCNF는 아니다.

▷ 결정자가 Candidate key가 아닌 함수를 제거하면 BCNF를 만족함.

▷ 결정자가 후보키가 아니고 종속자가 기본키의 부분집합인 속성은 분해하면 BCNF이다.

▷ BCNF에 속하는 모든 릴레이션은 FD 정보와 관련하여 중복성이 발생하지 않는다.

▷어떤 Relation 스키마를 R이라고 하고, R에 속하는 attribute들의 부분집합을 X라 하고, R에 속하는 어떤 attribute를 A라고 하면, 이때 R에 주어진 함수 종속성들의 집합 F에 속하는 모든 FD X → A에 대해,

다음 조건들 중 하나만 만족하는 경우, R은 BCNF라고 함.

1. A ∈ X (X가 {ssd}라면 A도 {ssd}인 경우 or [X{cc,bb}일때 A{bb} or A{cc} or A{bb,cc} 인 경우])
2. X는 super key이다.

여기서 1번 조건을 확인하는 것은 closer F^+에 속하는 각 종속성 X → A를 모두 살펴야 되기 때문에
2번 조건인 F에 속학는 각 FD의 좌측(결속자)에 key가 포함되어 있는지를 검사하는 것만으로 충분함.

X	Y	A
x	y1	a
x	y2	?

BCNF에 대한 한 예

이 인스턴스가 FD {X} → {A}를 만족한다고 가정하면
두 번째 tuple의 A attribute의 value가 a인것을 추론해 낼 수 있다.
하지만 이런 상황은 중복성의 한 예이다. 실제로 a라는 값이 두 번 저장되었기 때문.
이런 상황은 BCNF에 속하는 릴레이션에서 발생할 수 없음.
왜냐하면 BCNF를 만족한다면, A와 X는 서로 다든 attribute이므로 X가 key가 되어야 하는데 X가 key가 되버리면 y1 = y2가 되버리므로 두 개의 tuple이 모든 attribute에 대해 서로 동일한 값을 가지게 되므로, 한 릴레이션에서 동일한 두 개의 투플은 허용이 안된다는 것을 알기 때문에 위와 같은 상황은 BCNF가 아니다.
따라서 X는 key가 될 수 없으며 위와 같은 상황으 BCNF에서 발생 불가능.

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield
Joe	987-65-4321	908-555-1234	Westfield

{SSN} → {Name,City}

This FD is *bad* because it is **not** a superkey

⟹ **Not** in BCNF

What is the key?
{SSN, PhoneNumber}

이 예시의 FD를 보면 {ssn} → {name,city}관계를 보여주는데
이 테이블에서 결정자{name,city}가 기본키의 부분집합이 아니므로 BCNF아니다.
이 테이블에서 SSN이 superkey의 역할을 하지 못하므로 BCNF가 아니다.
그래서 위의 1,2번 조건중 둘다 만족못하므로 BCNF가 아님
BCNF를 할때는 좋은 fd조건을 가지고 해야됨!

Name	SSN	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Madison

SSN	PhoneNumber
123-45-6789	206-555-1234
123-45-6789	206-555-6543
987-65-4321	908-555-2121
987-65-4321	908-555-1234

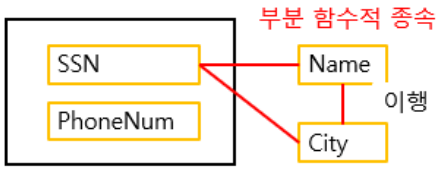
{SSN} → {Name,City}

This FD is now *good* because it is the key

Let's check anomalies:
- Redundancy ?
- Update ?
- Delete ?

Now in BCNF!

11



이번 그림을 보면 SSN이 위에 테이블에서 super key역할을 하므로 BCNF를 만족 <- 2번 조건
아래 테이블도 {SSN,PhoneNumber}가 합쳐져 key를 형성하고 중복된 값이 없으므로 BCNF를 만족함. <- 1번조건
따라서 각 테이블이 하나의 조건만을 만족하므로 BCNF이다.
항상 이렇게 테이블을 분해(decompositon)할때 데이터가 손실되는거가 있으면 안됨.

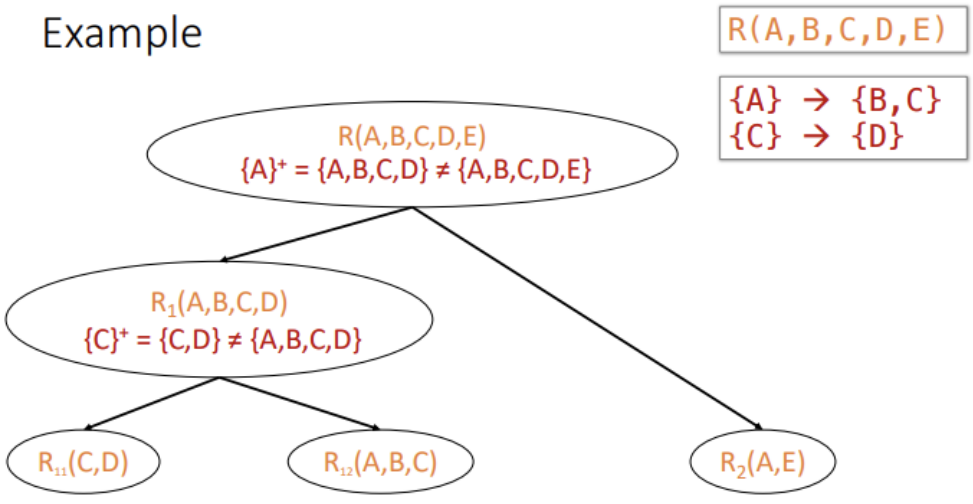
BCNFDecomp(R):
Find a *set of attributes* X s.t.: $X^+ \neq X$ and $X^+ \neq$ [all attributes]

if (not found) **then Return** R

let $Y = X^+ - X$, $Z = (X^+)^C$
decompose R into $\mathbf{R_1(X \cup Y)}$ and $\mathbf{R_2(X \cup Z)}$

Return BCNFDecomp(R_1), BCNFDecomp(R_2)

BCNF 릴레이션으로 분해하는 알고리즘

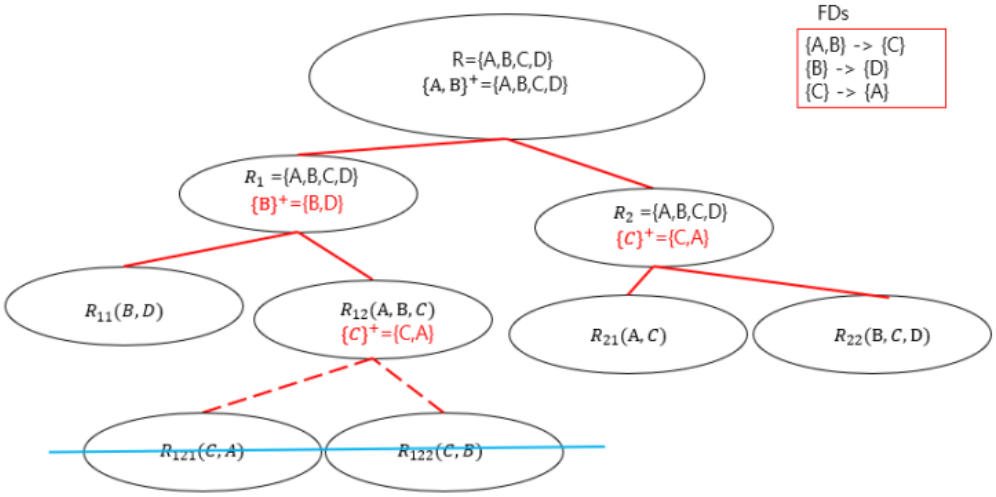


R 릴레이션을 BCNF로 Decomposition하는 방법은 우선 FD를 보고
A^+을 구하는데 A^+={A,B,C,D}이고 Y = A^+-A = {B,C,D}, Z={A^+}^{C}={E}이므로
R_1=(A∪Y)=(A,B,C,D), R_2=(A∪Z) = (A,E)가 되고
R_1에서 C가 D를 결정지을 수 있기 때문에 마저 BCNF로 Decomposition을 하면 위 와 같이

C^+을 구하는데 C^+={C,D}이고 Y = C^+-C = {D}, Z={C^+}^{C}={A,B,}이므로

R_{11}=(CUY)=(C,D), R_{12}=(CUZ) = (A,B,C)가 된다.

R_{11}의 superkey는 C가 되고, R_{12}의 superkey는 A가 되고 최종적으로 3개의 테이블을 구하게 됨.



R_{12}의 테이블을 BCNF화 하게 되면 FD중 {A,B} -> {C} 라는 관계가 사라지기 때문에 더이상 분해하지 않는다.

최종적인 table은 R_{11},R_{12},R_{21},R_{22}이며, super kye는 R_{11}에는 B,R_{12}에는 AB,R_{21}과R_{22}에는 C 이다.

https://yaboong.github.io/database/2018/03/10/database-normalization-2/

Decompositon

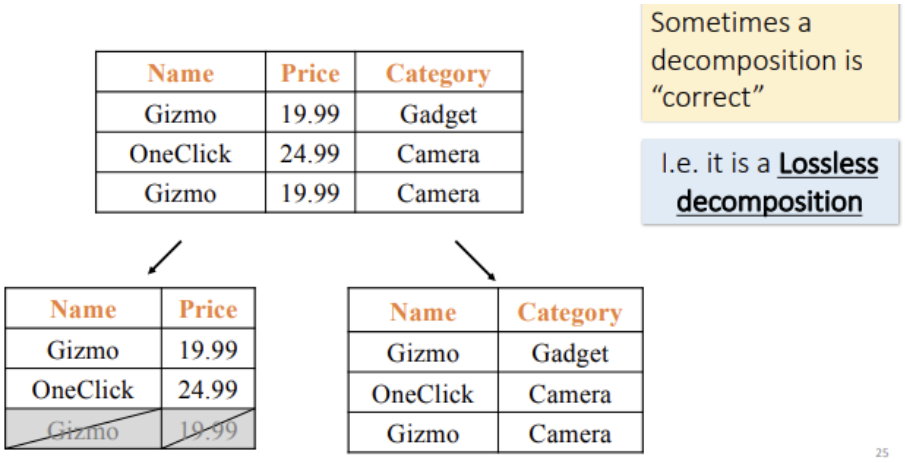
▷ 분해를 할때는 정보의 손실이 있으면 안된다. 즉,무손실 분해여야 함.

▷ natural join은 할때 정보의 손실이 없어야 함.

▷ BCNF를 위반하는 릴레이션은 분해할려고 하면

1. BCNF를 위반하는 nontrivial FD X -> Y를 찾는다.
   즉. 결정자가 슈퍼키가 아닌 fd의 경우를 찾으면 된다.
2. 이렇게 되면 R1(결정자,종속자) 릴레이션과 R2(결정자,a_1,..,a_n) 릴레이션으로 분해
   이렇게 되면 최종적으로 BCNF를 위반하는 릴레이션 되고,
   만약 결정자가 super key이면 BCNF로 분해가 된 것이다.

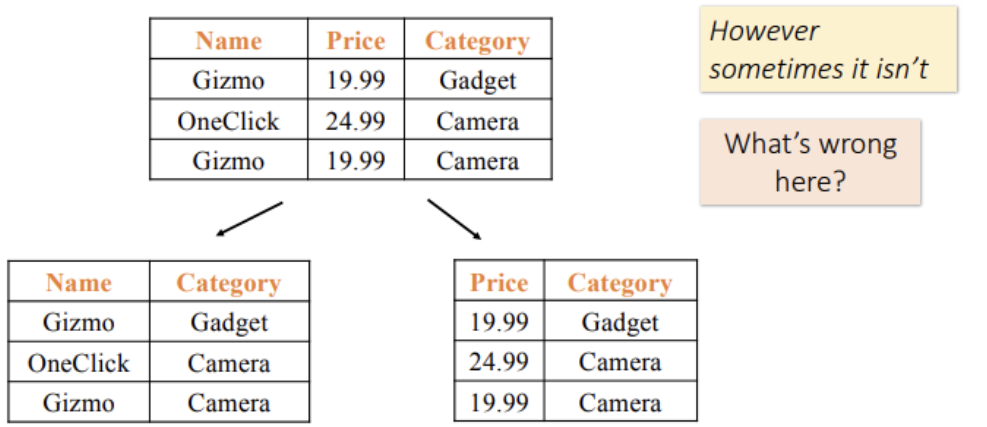
▷ 분해할때 주의해야 될 점은 새로운 정보를 삽입하였을때 주어진 local FD의 조건을 만족해야됨



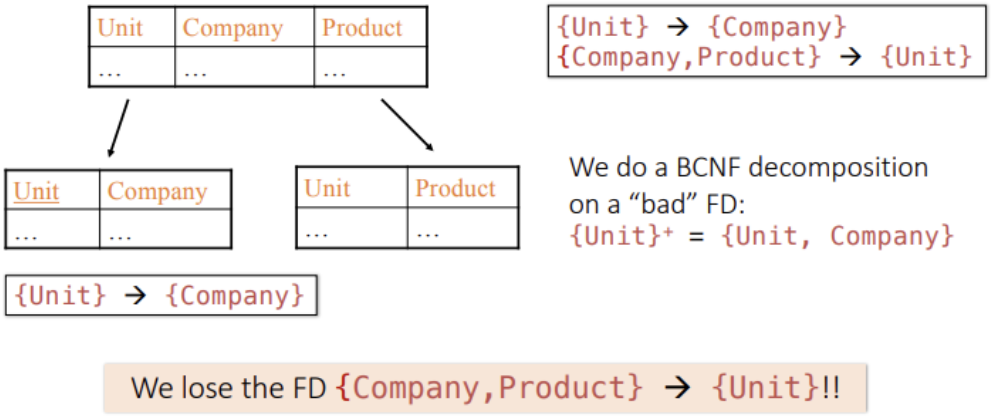
지금과 같은 예시는 정보의 손실이 없는 decomposition이다.

왜냐하면 FD 조건을 {name} -> {price}로 주었고, 이 조건을 기준으로 (결정자,종속자), (결정자,나머지 속성들)로 테이블들을 구성했는데 이 두 테이블을 natural join했을때 사라지는 정보가 없기 때문이다.

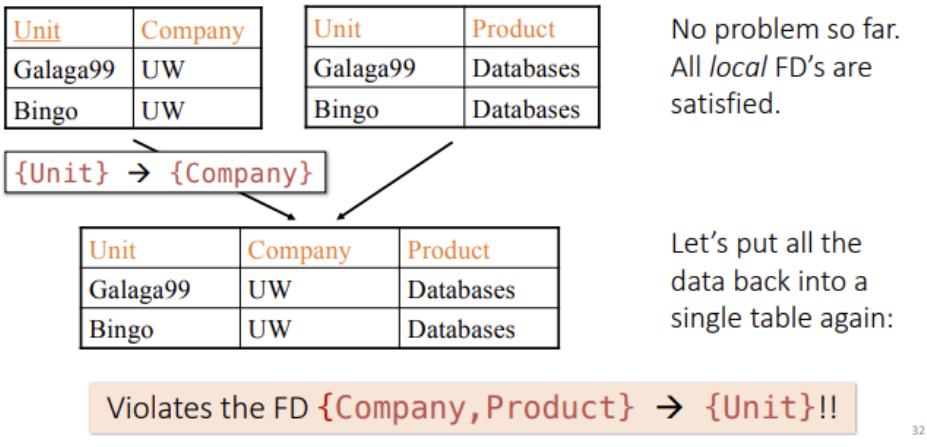
Lossy Decomposition



지금 예시는 정보의 손실이 있는 분해방법이다.
FD를 보면 {Category} -> {Name} 이고, 결정자(Category)를 기준으로 테이블을 분해하면 위와 같이 두 개의 테이블로 분해가되고, 결정자(Category)를 기준으로 natural join을 하게 됐을때 camera부분에서 tuple이 각각 2개씩 더나와 최종적으로 5개의 tuple을 가진 table이 생성되기 때문에 잘못된 분해 방법이다 .
즉 좋지않은 FD로 BCNF분해를 했기 때문에 문제가 발생한 것이다.



위와 같은 예시에서 주어진 FDs 중 unit을 결정자로 하여 테이블을 분해하면 {Company,Product} -> {Unit}라는 fd를 잃게 된다.
따라서 이런 경우는 다시 join을 했을때 정보의 손실을 얻게 된다.
그 예시가 아래와 같다.



Company,Product에 의해서 unit이 결정되어야 되는데 join한 테이블에서 company,product가 동일함에도 불구하고 unit이 다 다르기 때문에 정보의 손실이 발생하였음을 확인 할 수 있다.

Prof. Lee Seungjin님이 제공해준 Data Processing Systems의 자료를 활용하여 제작하였습니다.