

8장. 관계형 데이터베이스 설계

이 동 호

데이터베이스 연구실
소프트웨어학부

목차

- 8.1 좋은 관형형 데이터베이스 설계의 특징
- 8.2 원자적 도메인과 제1정규형
- 8.3 함수 종속을 이용한 분해
- 8.4 함수 종속 이론
- 8.5 함수 종속을 위한 알고리즘
- 8.6 다중값 종속을 이용한 분해
- 8.7 더 많은 정규형
- 8.8 데이터베이스 설계 과정

8.1 좋은 데이터베이스 설계

- 만약 *instructor*와 *department*를 합쳐서 *inst_dept* 만들면?
- 많은 정보의 중복이 발생 !

inst_dept

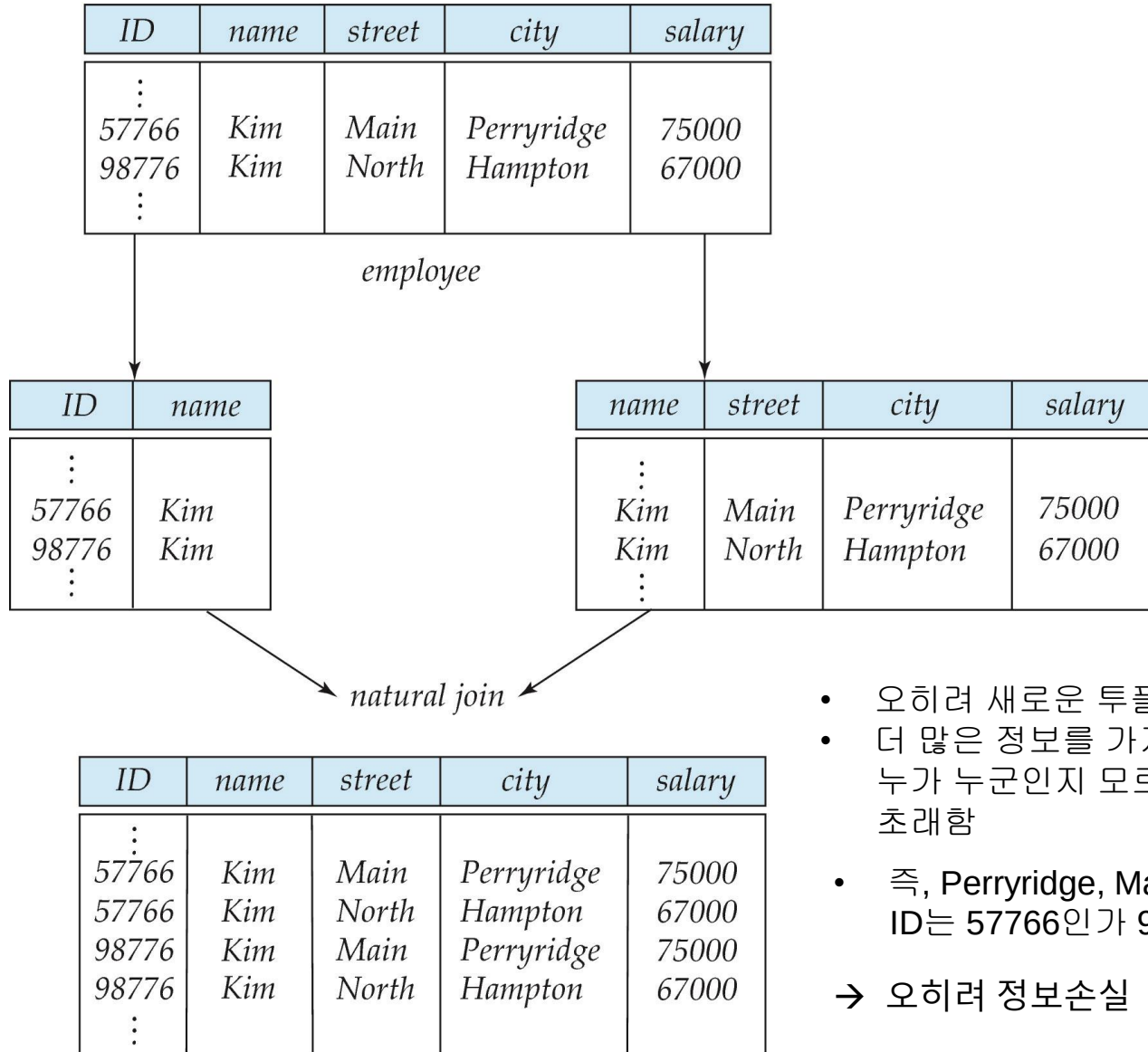
<i>ID</i>	<i>name</i>	<i>salary</i>	<i>dept_name</i>	<i>building</i>	<i>budget</i>
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	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

좀 더 작은 스키마로 분해하면 ?

- *inst_dept*를 어떻게 *instructor* 와 *department* 로 분해할 것인가?
- 함수 종속(**functional dependency**)을 이용
 - 만약 속성 *dept_name*와 *building*, *budget*간의 종속 관계가 존재한다는 사실이 주어진면? 즉,
 - *dept_name* → *building*, *budget*
 - “*dept_name*에 따라 *building*과 *budget*이 결정된다는 의미”
 - (*dept_name*, *building*, *budget*)으로 구성된 스키마 생성
- 모든 분해가 바람직하는지 모름
 - *employee*(*ID*, *name*, *street*, *city*, *salary*)
 - employee1* (*ID*, *name*)
 - employee2* (*name*, *street*, *city*, *salary*)
- 잘 못 된 분해는 정보를 손실할 수 있음
 - 손실 분해(**lossy decomposition**)

함수적 종속: 주어지는 것인가? 찾아야 하는 것인가?

손실 분해



- 오히려 새로운 튜플이 추가됨!
 - 더 많은 정보를 가지게 되었지만, 누가 누군인지 모르게 되는 결과를 초래함
 - 즉, Perryridge, Main에 사는 kim의 ID는 57766인가 98776인가?
- 오히려 정보손실

무손실 조인 분해

- 무손실 조인 분해(Lossless join decomposition)
- $R = (A, B, C)$ 을 다음의 두 스키마로 분해할 경우
 $R_1 = (A, B) \quad R_2 = (B, C)$

A	B	C
α	1	A
β	2	B

r

A	B
α	1
β	2

$\Pi_{A,B}(r)$

B	C
1	A
2	B

$\Pi_{B,C}(r)$

$\Pi_A(r) \bowtie \Pi_B(r)$

A	B	C
α	1	A
β	2	B

8.2 원자적 도메인과 제1정규형

- 속성의 도메인이 더 이상 분해되지 않을 경우 원자적(atomic)이라고 함
 - 비원자적 도메인의 예:
 - name과 같은 복합 속성
 - CS101과 같은 식별번호도 분해될 수 있음 (CS+101)
 - (CS101에서 CS와 101을 분리하려는 시도를 하지 않으면 atomic)
- **제1정규형(first normal form)**
 - 릴레이션 스키마 R의 모든 속성들의 도메인이 원자적일 때 R은 제1정규형을 만족함
- 비원자적 속성은 데이터의 중복과 저장시 여러 문제를 발생
 - 예제: 복합 속성을 그대로 테이블로 만들어 저장할 때 여러 문제 발생
 - 모든 릴레이션은 기본적으로 제1정규형을 만족해야 함

원자적 도메인과 제1정규형 (Cont.)

- 원자성은 도메인을 해석하는데 중요
 - 예제: 문자열은 일반적으로 분해 불가능한데 만약 학생에게 *CS0012* 혹은 *EE1127*같이 roll number를 부여하고, 첫 두 문자가 소속학과를 의미한다면 ? → roll number의 도메인은 더이상 원자적이지 않음
 - 이런 식의 해석은 좋은 방법이 아님
- 즉, DB 차원이 아니라 application차원에서 roll number를 분해하여 소속학과를 알아내야 한다. 만약 그렇게 되면 학생이 소속학과를 변경하게 되면 application 프로그램을 다시 작성해 한다. (바람직하지 않다)
- 그러면, course 스키마에서 course_id(e.g. CS-101, BIO-301 등)는 atomic한가?
- DB application 프로그램에서 course_id를 나누려는 시도를 하지 않거나, 이를 학과의 축약형으로 해석하지 않는 한 atomic하다고 볼 수 있다.

릴레이션 **R**이 좋은 설계인지를 판별하기 위한 방법론

- 릴레이션 **R**이 좋은 설계인지를 판별하자!
- 만약 릴레이션 **R**이 좋은 설계가 아니라면 **R**을 다음 조건을 만족하도록 더 작은 스키마 $\{R_1, R_2, \dots, R_n\}$ 로 분해해야 함
 - 각 릴레이션 R_i 은 좋은 설계이어야 함
 - 분해는 무손실 분해이어야 함
- 이런 방법론은 다음의 두 이론이 기반함
 - 함수 종속
 - 다중값 종속

}

어떤 스키마를 분해할지 말지를 평가하는 방법론을 제시!

8.3 함수 종속을 이용한 분해

- 함수 종속
 - 릴레이션 집합에 존재하는 제약조건
 - 어떤 속성 집합의 값이 다른 속성 집합을 값을 유일무이 (uniquely)하게 결정하는 성질
 - 함수 종속은 키로 표현될 수 있음
 - (수퍼)키 → 튜플을 유니크하게 결정할 수 있는 속성 집합

함수 종속

- R : 릴레이션 스키마

$$\alpha \subseteq R \text{ and } \beta \subseteq R$$

- 함수 종속(**functional dependency**)

$$\alpha \rightarrow \beta \quad \alpha : \text{결정자 (determinant)}, \beta : \text{종속자 (dependent)}$$

릴레이션 $r(R)$ 에 존재하는 두개의 튜플 t_1 과 t_2 에 대해서, 속성 α 의 값이 서로 같다면 속성 β 의 값도 서로 같다. 즉,

$$t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$$

- 예제: $r(A, B)$

A	B
1	4
1	5
3	7

$$B \rightarrow A$$

$$\frac{t_1[\alpha] \neq t_2[\alpha]}{\downarrow} \vee \frac{t_1[\beta] = t_2[\beta]}{\downarrow}$$

$$(\text{항상 True}) \vee (\text{true or false}) = \text{T}$$

- $A \rightarrow B$: 함수 종속 존재? 아니요!! 그럼, $B \rightarrow A$?

$$(t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]) \equiv (t_1[\alpha] \neq t_2[\alpha] \vee t_1[\beta] = t_2[\beta])$$

키와 함수 종속

- K 가 릴레이션 스키마 R 의 수퍼키이면 $K \rightarrow R$ 로 표현 가능
- 함수 종속은 수퍼키를 사용해서 표현할 수 없는 제약 조건을 표현할 수 있음

- 예제:

inst_dept (ID , *name*, *salary*, $dept_name$, *building*, *budget*).

존재하는 함수 종속:

$dept_name \rightarrow building$ 혹은 $ID \rightarrow building$

그러나, 아래 같은 함수 종속은 존재하지 않음:

$dept_name \rightarrow salary$

즉, $dept_name$ 이 *inst_dept*의 superkey는 아님 $\rightarrow$ 함수적 종속은 superkey를 표현할 수 없는 제한조건까지 표현할 수 있다.

함수 종속의 활용

- 함수 종속이 주어졌을 때 어떤 릴레이션이 적법한(**legal**)한지를 판별할 때,
 - 릴레이션 r 이 주어진 함수 종속을 모두 만족할 때, r 은 적법한 릴레이션이다.
- 적법한 릴레이션의 제약 조건을 명세할 때,
 - R 상의 정의된 모든 적법한 릴레이션들이 함수 종속 집합 F 를 만족할 때, F 는 R 상에서 항상 지켜진다(**hold on**).
- 주의: 비록 모든 함수 종속이 모든 릴레이션 인스턴스에 대해서 지켜질 수 없지만 우연히 어떤 인스턴스는 함수 종속을 모두 만족할 수도 있다..
 - 예를 들어, 어떤 *instructor* 인스턴스에 대해서 우연히 함수 종속 $name \rightarrow ID$ 이 만족할 수도 있음
동명이인이 없는 경우 우연히 $name \rightarrow ID$ 만족될 수도 있음.

함수 종속의 활용 (Cont.)

- 어떤 함수 종속이 릴레이션의 모든 인스턴스들에 대해서 만족된다면 해당 함수 종속은 자명(**trivial**)하다고 말한다.
 - 예제:
 - $ID, name \rightarrow ID$
 - $name \rightarrow name$
 - 일반적으로,

$$\alpha \rightarrow \beta \text{ is trivial if } \beta \subseteq \alpha$$

함수 종속 집합의 폐포

닫혀 있는 집합(Closure)

- 함수 종속의 집합 F 에 존재하는 함수 종속들을 이용하여 논리적 유추 가능한 함수 종속이 존재
 - 예를 들어, $A \rightarrow B$ 와 $B \rightarrow C$ 를 통해 $A \rightarrow C$ 추론할 수 있음
- F 의 폐포(**closure** of F) : F^+
 - 함수 종속 집합 F 에 존재하는 함수 종속들을 이용해 유추할 수 있는 모든 함수 종속의 집합
 - F^+ 는 F 의 모집합(superset)

[“닫혀있다”의 개념]

제약조건: “ $A \odot B = C$ ”, A, B, C 는 모두 정수이다”

“정수 A, B, C 에 대해, $A \odot B = C$ 이다.”

→ 이 제약조건은 $+$, $-$, $\times$ 에 닫혀있다. $/$ 에는 닫혀있지 않다.

보이스-코드(Boyce-Codd) 정규형

- 릴레이션 스키마 R 이 보이스-코드 정규형(BCNF)일 조건:
 - 함수 종속 집합 F 와 F^+ 에 존재하는 모든 함수 종속 $\alpha \rightarrow \beta$ ($\alpha \subseteq R, \beta \subseteq R$) 에 대해서 다음 두 조건 중에 하나라도 만족해야 함

- 1 $\alpha \rightarrow \beta$ 는 자명함 (i.e., $\beta \subseteq \alpha$)
- 2 α 가 R 의 수퍼키

- 예제:
 - $instr_dept (ID, name, salary, dept_name, building, budget)$
 - 함수 종속: $dept_name \rightarrow building, budget$

$dept_name \rightarrow building$ trivial 한가?
 Superkey 인가?

BCNF로 분해하기

- 스키마 R 이 자명하지 않은 함수 종속 $\alpha \rightarrow \beta$ 에 의해 BCNF가 아닐 경우
 - 아래와 같은 두개의 스키마로 분해
 - $(\alpha \cup \beta)$
 - $(R - (\beta - \alpha))$
- 예제:
 - $\alpha = dept_name$
 - $\beta = building, budget$
 - $inst_dept$ 는 아래와 같이 분해
 - $(\alpha \cup \beta) = (dept_name, building, budget)$
 - $(R - (\beta - \alpha)) = (ID, name, salary, dept_name)$

BCNF와 종속성 보전

- 종속성 보전(dependency preservation)
 - 원래 스키마 R 을 R_1 과 R_2 로 분해한 후에도 R 상에 존재하는 함수 종속들이 R_1 및 R_2 에 대해서 그대로 유지가 된다면 종속성 보존이라 함
 - 함수 종속 집합에 존재하는 모든 함수 종속들이 R_1 과 R_2 에 대해서 만족할 필요가 없고, R_1 에 존재하는 속성들만으로 구성된 함수 종속들에 대해서만 테스트하면 됨. R_2 도 마찬가지
- BCNF와 종속성 보전을 모두 유지하기 어려움
 - 좀 더 약한 정규형 필요 → 제3정규형

제3정규형

- 릴레이션 스키마 R 이 제3정규형(**3NF**)일 조건
 - 함수 종속 집합 F^+ 에 존재하는 모든 함수 종속 $\alpha \rightarrow \beta$ 에 대해서 아래의 3가지 조건 중에 하나라도 만족해야 함

- 1 $\alpha \rightarrow \beta$ 는 자명함 (i.e., $\beta \in \alpha$)
- 2 α 가 R 의 수퍼키
- 3 $\beta - \alpha$ 존재하는 각 속성들이 R 의 후보키에 포함됨
(주의: 각 속성들이 다른 후보키에 포함되어도 됨)

- 세번째 조건은 종속성 보전을 위한 조건임 ($\rightarrow$ 의미는 강의노트 42~43 페이지에서 다시 설명)
- BCNF이면 반드시 3NF이다 ?

BCNF는 과연 좋은 설계인가?

- 예제:

inst_info (ID, child_name, phone)

– instructor 는 여러 명의 자녀와 전화를 가질 수 있다.

inst_info

<i>ID</i>	<i>child_name</i>	<i>phone</i>
99999	David	512-555-1234
99999	David	512-555-4321
99999	William	512-555-1234
99999	Willian	512-555-4321

존재하는 FD 찾아보기? ID → child_name or phone? 존재하지 않음

ID, child_name → phone? 존재하지 않음

결국, 존재하는 모든 FD는 (ID, child_name, phone) → ID or child_name or phone 형태임.
So, inst_info에 존재하는 함수적 종속은 모두 trivial 하다. (BCNF임)

BCNF는 과연 좋은 설계인가?(Cont.)

- Inst_info는 BCNF 임
 - 과연 좋은 릴레이션 스키마인가?
 -
- 삽입 이상현상
 - 새로운 전화번호 981-992-3443 를 ID 99999에 삽입하고자 할 때, 두 개의 튜플을 삽입해야 함 → 하나의 번호를 추가하는데 왜 두 번을 삽입해야 하는가?
 - (99999, David, 981-992-3443)
 - (99999, William, 981-992-3443)

BCNF는 과연 좋은 설계인가?(Cont.)

- 만약 *inst_info*를 아래와 같이 분해

<i>inst_child</i>	<i>ID</i>	<i>child_name</i>
	99999	David
	99999	David
	99999	William
	99999	Willian

<i>inst_phone</i>	<i>ID</i>	<i>phone</i>
	99999	512-555-1234
	99999	512-555-4321
	99999	512-555-1234
	99999	512-555-4321

더 높은 차수의 정규형이 필요 → 제4정규형(4NF), 뒷 장에서 다시 언급

정규화의 목표

- 릴레이션 R 과 R 상에 존재하는 함수 종속 집합 F 에 대해서,
 - ① 릴레이션 스키마 R 이 좋은 설계인지를 결정하는 것!!
 - ② 만약, R 이 좋은 설계가 아니라면 아래 조건을 만족하는 더 작은 릴레이션 스키마의 집합 $\{R_1, R_2, \dots, R_n\}$ 으로 분해하는 것!!
 - R_i 는 좋은 설계 each relation scheme is in good form
 - 분해는 무손실 조인 분해
 - 가능하면 분해시 종속성 보존

8.4 함수 종속 이론

- 함수 종속 집합의 폐포(F^+)를 구하는 방법
- 속성 집합의 폐포를 구하는 방법과 활용
- BCNF와 3NF으로 무손실 조인 분해하는 방법
- 분해가 종속성 보전인지 여부를 판별하는 방법

함수 종속 집합의 폐포

- 함수 종속의 집합 F 에 존재하는 함수 종속들을 이용하여 논리적 유추 가능한 함수 종속이 존재
 - 예를 들어, $A \rightarrow B$ 와 $B \rightarrow C$ 를 통해 $A \rightarrow C$ 추론할 수 있음
- F 의 폐포(**closure** of F) : F^+
 - 함수 종속 집합 F 에 존재하는 함수 종속들을 이용해 유추할 수 있는 모든 함수 종속의 집합
 - F^+ 는 F 의 모집합(superset)

함수 종속 집합의 폐포 (Cont.)

- 암스트롱의 공리(**Armstrong's Axioms**)를 반복 적용하여 F^+ 구
수 있음
 - if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ 재귀법칙 (**reflexivity**)
 - if $\alpha \rightarrow \beta$, then $\gamma \alpha \rightarrow \gamma \beta$ 증가법칙 (**augmentation**)
 - if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ 이행법칙 (**transitivity**)

예제

- $R = (A, B, C, G, H, I)$
 $F = \{$
 - $A \rightarrow B$
 - $A \rightarrow C$
 - $CG \rightarrow H$
 - $CG \rightarrow I$
 - $B \rightarrow H\}$
- 추론 가능한 새로운 함수 종속: F^+
 - $A \rightarrow H$
 - $A \rightarrow B, B \rightarrow H$ (이행법칙)
 - $AG \rightarrow I$
 - $AG \rightarrow CG$ ($A \rightarrow C$ 양쪽에 G 추가 : 증가법칙)
 - $CG \rightarrow I$ (이행법칙)
 - $CG \rightarrow HI$
 - $CG \rightarrow CGI$ ($CG \rightarrow I$ 양쪽에 CG 추가: 증가법칙)
 - $CGI \rightarrow HI$ ($CG \rightarrow H$ 양쪽에 I 추가: 증가법칙)
 - 그리고, 이행법칙 적용

함수 종속 집합의 폐포 (Cont.)

- 추가적인 규칙:
 - 결합법칙 (**union**)
 - If $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$, then $\alpha \rightarrow \beta \gamma$
 - 분해법칙 (**decomposition**)
 - If $\alpha \rightarrow \beta \gamma$, then $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$
 - 가이행법칙 (**pseudotransitivity**)
 - If $\alpha \rightarrow \beta$ and $\gamma \beta \rightarrow \delta$, then $\alpha \gamma \rightarrow \delta$

암스트롱의 공리로부터 추론 가능한 추가적인 규칙

속성 집합의 폐포

- 속성 집합 α 의 폐포 : α^+
 - α 가 주어졌을 때 함수 종속 집합 F 를 적용하여 결정되는 모든 속성들의 집합 (즉, $\alpha \rightarrow \alpha^+$)
- F 가 주어졌을 때 α^+ 구하는 알고리즘

```

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

```

속성 집합의 폐포 예제

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

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

- $(AG)^+$

1. $result = AG$

2. $result = ABCG \quad (A \rightarrow C \text{ and } A \rightarrow B)$

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

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

- Is AG a candidate key?

1. Is AG a super key?

1. Does $AG \rightarrow R?$ == Is $(AG)^+ \supseteq R$

2. Is any subset of AG a superkey?

1. Does $A \rightarrow R?$ == Is $(A)^+ \supseteq R$

2. Does $G \rightarrow R?$ == Is $(G)^+ \supseteq R$

$$(A)^+$$

1. Result = A

2. Result = ABC

3. Result = ABCH

$$A \rightarrow R ? \text{ No}$$

$$(G)^+$$

1. Result = G

$$G \rightarrow R ? \text{ No}$$

속성 집합 폐포의 활용

① 수퍼키 판별:

- α 이 수퍼티인지 판별하기 위해서 α^+ 구한 후, α^+ 가 R 의 모든 속성을 포함하는지 확인

② 함수 종속 테스트:

- 함수 종속 $\alpha \rightarrow \beta$ 가 지켜지는 테스트하기 위해서 α^+ 구한 후, $\beta \subseteq \alpha^+$ 인지 확인
- ($\alpha \rightarrow \beta$ 가 F 에 존재하지 않을 경우) F^+ 를 찾아서 $\alpha \rightarrow \beta$ 가 존재하는지를 확인하는 것보다 훨씬 계산이 단순하고 효율적임

③ F 의 폐포 계산

- R 에 존재하는 모든 속성 집합 γ 에 대해(즉, $\gamma \subseteq R$), γ^+ 을 계산한 후에 $S \subseteq \gamma^+$ 를 만족하는 S 에 대해서 함수 종속 $\gamma \rightarrow S$ 출력

무손실 (조인) 분해

- $R = (R_1, R_2)$ 에 대해서,

$$r = \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) \rightarrow \text{무손실 분해}$$
- R 을 R_1 과 R_2 로 분해한 것이 무손실이기 위해서는 적어도 다음 두 개의 함수 종속 중에 하나가 F^+ 에 존재해야 함
 - $R_1 \cap R_2 \rightarrow R_1$
 - $R_1 \cap R_2 \rightarrow R_2$ } 결국 $R_1 \cap R_2$ 가 R_1 혹은 R_2 의 superkey를 구성한다.
- $employee(ID, name, street, city, salary)$
 - $employee1(ID, name)$
 - $employee2(name, street, city, salary)$ } name은 분해된 두 릴레이션에서 어느쪽에서도 superkey가 되지 못함
- $inst_dept(ID, name, salary, dept_name, building, budget)$
 - $instructor(ID, name, dept_name, salary)$
 - $department(dept_name, building, budget)$ } dept_name은 department의 superkey를 형성함

↓
무손실 분해

예제

- $R = (A, B, C)$
 $F = \{A \rightarrow B, B \rightarrow C\}$
 - 두가지 방법으로 분해 가능!!
- $R_1 = (A, B), R_2 = (B, C)$
 - 무손실 분해:

$$R_1 \cap R_2 = \{B\} \text{ and } B \rightarrow BC \text{ (} B \text{는 } R_2 \text{의 슈퍼키)}$$
 - 종속성 보존 (본래 $A \rightarrow B, B \rightarrow C$ 는 분해 이후에도 보존됨)
- $R_1 = (A, B), R_2 = (A, C)$
 - 무손실 분해:

$$R_1 \cap R_2 = \{A\} \text{ and } A \rightarrow AB \text{ (} A \text{는 } R_1 \text{의 슈퍼키)}$$
 - 종속성 보존 안됨 ($B \rightarrow C$ 는 분해된 이후 보존되지 않음)
 $(R_1 \text{과 } R_2 \text{를 자연조인하지 않으면 종속성 } B \rightarrow C \text{를 확인할 수 없음})$

결국, $B \rightarrow C$ 라는 FD가 분해 이후에도 지켜지고 있는지를 확인하기 위해선 $R_1 \bowtie R_2$ 한 후에 check해 봐야 한다.

단순한 경우는 종속보존 여부를 쉽게 살펴볼 수 있지만, FD가 많고 복잡하면 종속보존을 Check하기 매우 어렵다.

종속성 보존

- $F_i: R_i$ 에 존재하는 속성으로만 구성된 함수 종속 집합 ($F_i \subseteq F^+$)
 - 아래 조건이 만족되면 종속성 보존(**dependency preserving**)

$$(F_1 \cup F_2 \cup \dots \cup F_n)^+ = F^+$$

- 종속성이 보존되지 않으면 함수 종속성을 체크하기 위해서 다시 조인을 해야 하면 이는 매우 비싼 비용을 요구함

- R_i 에 있는 속성들로만 구성된 FD를 F_i 라고 하자
- $F' = F_1 \cup F_2 \cup \dots \cup F_n$ 하자
- 일반적으로 $F' \neq F$ 일 수 있지만, 결국 $F'^+ = F^+$ 이다
- 다시 말해, F 의 모든 종속은 F' 를 논리적으로 내포하고 있다.

R 상에 정의된 FD의 집합 F : 총 10개

R을 R1과 R2로 분해했을 때,

F중에서 R1에 속한 속성만으로 구성된 FD의 집합 F_1 : 5개

F중에서 R2에 속한 속성만으로 구성된 FD의 집합 F_2 : 2개

→ 비록 $F \neq F_1 \cup F_2$ 이지만 만약 $F^+ = (F_1 \cup F_2)^+$ 이라면, 이 분해는 종속성 보존 분해

예제

- $R = (A, B, C)$
 $F = \{A \rightarrow B$
 $B \rightarrow C\}$
 $\text{Key} = \{A\}$
- R 은 BCNF인가? $\rightarrow$ No (Why?)
- 분해 $R_1 = (A, B), R_2 = (B, C)$
 - R_1 과 R_2 는 모두 BCNF임
(why? $\rightarrow A$ 와 B 가 각각 R_1 과 R_2 의 슈퍼키이므로...)
 - 무손실 조인 분해
(why? $\rightarrow R_1 \cap R_2 = B$ and $B \rightarrow BC$ 이므로...)
 - 종속성 보존
(why? $\rightarrow F_1: A \rightarrow B, F_2: B \rightarrow C, (F_1 \cup F_2)^+ = F^+$ 이므로...)
- 그럼 분해 $R_1 = (A, B), R_2 = (A, C)$ 어떻게 될까?

8.5 BCNF 분해 알고리즘

- BCNF 분해
 - 어떤 릴레이션 R 이 BCNF인지 아니지를 판별하는 것은 BCNF 정의를 이용하여 바로 판별 가능
 - 그러나 F^+ 를 구하는 문제는 매우 복잡함
 - 간단히 검사할 수 있는 단순화된 검사 방법 존재
 - 릴레이션 R 이 BCNF가 아닐 때 BCNF인 더 작은 릴레이션으로 분해할 수 있음

BCNF 판별

- 자명하지 않은 함수 종속 $\alpha \rightarrow \beta$ 가 BCNF를 위반하게 만드는지를 검사
 1. 먼저, α^+ 계산
 2. α^+ 가 R의 모든 속성을 포함하는지(즉, α 가 수퍼키인지)를 검사
- **단순화된 검사:** R이 BCNF인지를 검사하기 위해서 모든 함수 종속, 즉 F^+ 를 계산할 필요 없이 F 에 존재하는 함수 종속들만 검사해도 충분함
 - F 에 존재하는 어떤 함수 종속도 BCNF를 위반하게 만들지 않는다면 F^+ 에 존재하는 어떤 함수 종속도 BCNF를 위반하게 만들지 않음

BCNF 판별 (Cont.)

- **[주의]** F 에 존재하는 함수 종속만을 가지고 하는 **단순화된 검사**는 R 을 분해해서 만들어지 릴레이션에 대해서 검사할 때는 항상 옳지 않을 수도 있음
 - $R = (A, B, C, D, E), F = \{ A \rightarrow B, BC \rightarrow D \}$
 - R 을 분해: $R_1 = (A, B), R_2 = (A, C, D, E)$
 - R_1 은 BCNF, R_2 는? $\rightarrow F$ 에 R_2 의 속성 (A, C, D, E) 로만 구성된 함수 종속이 없기 때문에 R_2 도 BCNF이다!!
 - $\rightarrow$ 즉, (A, C, D, E) 상에서 정의된 FD는 없음. 결국 자명하지 않은 함수 종속이 하나도 없음으로 BCNF 이다.!
 - **그러나, 사실 함수 종속 $AC \rightarrow D$ 가 F^+ 에 존재함**
 - $\rightarrow$ 즉, $AC \rightarrow D$ 는 non-trivial 하며, AC 가 R_2 의 수파도 아님.

BCNF 분해 (알고리즘)

- $R = (A, B, C)$
 $F = \{A \rightarrow B$
 $B \rightarrow C\}$
 $\text{Key} = \{A\}$
 - R 은 BCNF가 아님
 - (두 FD 모두 non-trivial함, 그러나 $A \rightarrow B$ 에서 A 가 슈퍼키임)
 - ($B \rightarrow C$, B 는 R 의 슈퍼키가 아님)
 - 분해 방법
 1. 먼저 BCNF를 위반하는 FD를 이용해 분해함
 $R_1 = (B, C)$
 2. 먼저 분해된 R_1 의 슈퍼키와 그 이외의 속성들 새로운 릴레이션 구성
 $R_2 = (A, B)$
- 1~2 과정을 F 에 존재하는 모든 함수 종속에 대해서 반복적으로 수행 :
 간략화된 알고리즘

BCNF 분해 예제

- *class* (*course_id*, *title*, *dept_name*, *credits*, *sec_id*, *semester*, *year*, *building*, *room_number*, *capacity*, *time_slot_id*)
- 함수 종속:
 - *course_id* → *title*, *dept_name*, *credits*
 - *building*, *room_number* → *capacity*
 - *course_id*, *sec_id*, *semester*, *year* → *building*, *room_number*, *time_slot_id*
- 후보키: {*course_id*, *sec_id*, *semester*, *year*}.
- *class*는 BCNF인가? 만약 아니라면 BCNF로 분해하시오.
 - *course_id* → *title*, *dept_name*, *credits* holds
 - but *course_id* is not a superkey.
 - We replace *class* by:
 - *course*(*course_id*, *title*, *dept_name*, *credits*)
 - *class-1* (*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *capacity*, *time_slot_id*)

BCNF 분해 예제 (Cont.)

- *course* is in BCNF
 - How do we know this?
- *building, room_number* → *capacity* holds on *class-1*
 - but {*building, room_number*} is not a superkey for *class-1*.
 - We replace *class-1* by:
 - *classroom* (*building, room_number, capacity*)
 - *section* (*course_id, sec_id, semester, year, building, room_number, time_slot_id*)
- *classroom* and *section* are in BCNF.

BCNF와 종속성 보존

- $R = (J, K, L)$
 $F = \{JK \rightarrow L \Rightarrow \text{non-trivial, but JK is superkey}$
 $L \rightarrow K \} \Rightarrow \text{non-trivial, and L is not superkey}$
 두개의 후보키 = JK 그리고 JL
- R 은 BCNF가 아님
- BCNF 분해:
 - 먼저 R_1 은 (L, K) 로 분해하고, 나머지 R_2 는 (J, L) 로 분해된다.
 - 그러면, 본래 있던 함수 종속인 $JK \rightarrow L$ 은 분해된 릴레이션에서 보존되지 않음
 - 보존하려면 ?
 - R_2 에 K (후보키 집합에 포함되지만 R_2 에는 없는 속성)을 추가함 $\rightarrow R_2: (J, K, L)$? 또 다른 문제 중복 발생!

종속성이 보존되는 **BCNF** 분해가 항상 가능하지 않을 수 있음

제3정규형: 동기

- 다음과 같은 상황 존재:
 - 종속성이 보존되지 않은 BCNF 이고,
 - 함수 종속성 위반 여부를 검사하는 것이 중요한 경우
- 방법: BCNF보다 약한 정규형 필요 → 제3정규형(3NF)
 - 약간의 중복을 허용하는 대신 각 릴레이션에 대해서 자연조인 없이 함수 종속을 검사할 수 있어야 함
 - 무손실 분해이면서 종속성이 보존되는 분해 → 3NF.

[3NF조건] 다음 3조건 중에 하나라도 만족

1. $\alpha \rightarrow \beta$ 는 자명함 (i.e., $\beta \in \alpha$)
2. α 가 R 의 수퍼키
3. $(\beta - \alpha)$ 에 존재하는 각 속성이 R 의 후보키에 포함됨
(주의: 각 속성이 다른 후보키에 포함되어도 됨)

$\alpha \rightarrow \beta$ 를 non-trivial하게
만드는 속성들

앞 슬라이드 예제에서 BCNF를 위반하게 만드는
함수 종속 $L \rightarrow K$ 에 대해서 3번 조건을 검사: 3NF

3NF 예제

- Relation *dept_advisor*:
 - *dept_advisor* (*s_ID*, *i_ID*, *dept_name*)
 - $F = \{s_ID, dept_name \rightarrow i_ID, i_ID \rightarrow dept_name\}$
 - Two candidate keys: (*s_ID*, *dept_name*) and (*i_ID*, *s_ID*)
 - *R* is in 3NF
 - $s_ID, dept_name \rightarrow i_ID$
 - non-trivial
 - *s_ID*, *dept_name* is a superkey
 - $i_ID \rightarrow dept_name$
 - non-trivial
 - *i_ID* is not superkey
 - *dept_name* is contained in a candidate key
($dept_name - i_ID = dept_name$)

BCNF와 3NF 비교

- 3NF:
 - 분해는 무손실
 - 종속성 보존
- BCNF
 - 분해는 무손실
 - 종속성 보존이 가능하지 않을 수도 있음

설계의 목표

- 관계형 데이터베이스 설계의 목표
 - BCNF
 - 무손실 조인 분해
 - 종속성 보전
- 3가지 목표를 동시에 달성하기 어려울 때
 - 종속성 보전을 포기하고 **3NF** 형태로 설계
- 참고: SQL은 수퍼키 이외에 함수 종속을 명세하기 위한 방법을 제공하지 않음

8.6 다중값 종속을 이용한 분해

- *inst_info* 릴레이션 스키마
 - *inst_info*(*ID*, *child_name*, *phone_number*)
 - 예제 데이터:
 - (99999, David, 512-555-1234)
 - (99999, David, 512-555-4321)
 - (99999, William, 512-555-1234)
 - (99999, William, 512-555-4321)
- 릴레이션 *inst_info*는 BCNF 임
 - 왜 ? → non-trivial dependency가 하나도 없음
즉, (*ID*, *child_name*, *phone_number*) → any attribute
 - 과연 좋은 설계인가 ? → If your answer is NO, why ?

다중값 종속 (MVDs)

- 릴레이션 스키마 R 그리고 $\alpha \subseteq R$, $\beta \subseteq R$ 일 경우, 만약 적법한 릴레이션 $r(R)$ 에 존재하는 모든 튜플 t_1 과 t_2 에 대해, 다음과 같은 t_3 와 t_4 가 존재한다면,

$$\begin{aligned} t_1[\alpha] &= t_2[\alpha] = t_3[\alpha] = t_4[\alpha] \\ t_3[\beta] &= t_1[\beta] \\ t_3[R - \beta] &= t_2[R - \beta] \\ t_4[\beta] &= t_2[\beta] \\ t_4[R - \beta] &= t_1[R - \beta] \end{aligned}$$

α 와 β 사이에 아래와 같은 **다중값 종속(multivalued dependency)** 존재

$$\alpha \twoheadrightarrow \beta \text{ (}\alpha \text{ multidetermines } \beta \text{)}$$

- ❖ α 는 β 를 다중으로 결정한다.
- ❖ 즉, 동일한 α 값에 대해서 서로 다른 β 값이 존재한다.

다중값 속성 (Cont.)

- $\alpha \twoheadrightarrow \beta$

	ID	child_name	phone_number
	α	β	$R - \alpha - \beta$
t_1	$a_1 \dots a_i$	$a_{i+1} \dots a_j$	$a_{j+1} \dots a_n$
t_2	$a_1 \dots a_i$	$b_{i+1} \dots b_j$	$b_{j+1} \dots b_n$
t_3	$a_1 \dots a_i$	$a_{i+1} \dots a_j$	$b_{j+1} \dots b_n$
t_4	$a_1 \dots a_i$	$b_{i+1} \dots b_j$	$a_{j+1} \dots a_n$

- 1) $\alpha \rightarrow \beta$: 동일한 α 값에 대해 서로 다른 β 값은 존재할 수 없다.
- 2) $\alpha \twoheadrightarrow \beta$: 동일한 α 값에 대해 서로 다른 β 값이 존재할 수 있다.

결국, 다중값 종속은 특정 튜플의 존재를 금지하는 것이 아니라, 릴레이션에서 특정 형태를 갖는 튜플들의 존재를 허용하는 것이다.

So, 동일성 생성 종속(equality generating dependency) 혹은 튜플 생성 종속(tuple generating dependency)라고도 부른다.

제4정규형

- 릴레이션 R , $\alpha \subseteq R$, $\beta \subseteq R$, 그리고 R 상의 모든 함수 종속 및 다중값 종속 집합 D 에 대해서 D^+ 집합에 존재하는 모든 다중값 종속 (함수 종속 포함) $\alpha \twoheadrightarrow \beta$ 에 대해서 아래의 두 조건 중에 하나라도 만족한다면 R 은 제4정규형임

1 $\alpha \twoheadrightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$ or $\alpha \cup \beta = R$)

2 α is a superkey for schema R

- 첫 번째 조건이 함수 종속에서 다중값 종속으로 변한 것 이외에 BCNF와 동일한 Condition 임
- 모든 함수 종속은 다중값 종속임 ($FD \subseteq MVD$)

예제

- 예제:

inst_info(*ID*, *child_name*, *phone_number*)

$ID \twoheadrightarrow child_name$

$ID \twoheadrightarrow phone_number$

- *inst_info* 는 4NF인가 ?
 - 아니다. 왜 ?
 - 다중값 종속이 4NF 조건을 만족하지 못함
 - 분해! (BCNF 분해와 유사한 방법으로 분해)
 - *r1*(*ID*, *child_name*)
 - *r2*(*ID*, *phone_number*)

8.7 추가 정규형

- 제5정규형 혹은 추출-조인 정규형(project-join normal form, PJNF)
 - 다중값 종속성을 일반화 시킨 조인 종속성 고려
 - 조인 종속성을 추론하기 위한 규칙이 존재하지 않음
 - 거의 사용되지 않은 정규형
- 제2정규형
 - 역사적으로만 의미 있음
 - refer to Practice Exercise 8.17

8.7 데이터베이스 설계 과정

- 좋은 스키마 R 을 설계하는 과정
 1. E-R 다이어그램을 테이블로 만드는 과정에서 릴레이션 스키마 R 생성
 2. R 이 좋은 설계인지 판별
 - 함수 종속성이 이용하여 R 이 BCNF 혹은 3NF인지 판별
 3. 만약 R 이 BCNF 혹은 3NF 조건을 만족한다면 좋은 설계라고 말할 수 있음. 아니면, R 을 더 작은 스키마로 분해
 - BCNF 분해 혹은 3NF 분해 방법
 4. 데이터베이스를 구성하는 모든 스키마 R 에 대해서 2~3의 과정을 반복함