7장: 정규화

데이터베이스 시스템 개념, 7 판.

개요

§ 좋은 관계 설계의 특징 § 기능적 종속성 § 기능적 종
속성을 사용한 분해

§ 일반 형식

§ 기능적 종속성 이론 § 기능적 종속성을 사용한
분해 알고리즘 § 다중값 종속성을 사용한 분해

§ 더 정상적인 형태

§ 원자 도메인 및 제1정규형

§ 데이터베이스 설계 프로세스

§ 임시 데이터 모델링

좋은 관계 설계의 특징

§ 강사와 부서를 in_dep로 결합한다고 가정합니다.
강사와 학과의 자연스러운 결합을 나타냅니다.

ID	name	salary	dept_name	building	budget
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

classroom(*building*, *room_number*, *capacity*)
department(*dept_name*, *building*, *budget*)
course(*course_id*, *title*, *dept_name*, *credits*)
instructor(*ID*, *name*, *dept_name*, *salary*)
section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)
teaches(*ID*, *course_id*, *sec_id*, *semester*, *year*)
student(*ID*, *name*, *dept_name*, *tot_cred*)
takes(*ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*)
advisor(*s_ID*, *i_ID*)
time_slot(*time_slot_id*, *day*, *start_time*, *end_time*)
prereq(*course_id*, *prereq_id*)

Figure 7.1 Database schema for the university example.

§ 정보의 중복이 있음 § null 값을 사용해야
함(강사가 없는 새로운 부서를 추가하는 경우)

분해

§ in_dep 스키마에서 정보 반복 문제를 피하는 유일한 방법은 이를 강사 스키마와 부서 스키마의
두 스키마로 분해하는 것입니다.

§ 모든 분해가 좋은 것은 아닙니다. 분해한다고 하자

직원(ID, 이름, 거리, 도시, 급여)

~ 안으로

employee1(ID, 이름)
employee2(이름, 거리, 도시, 급여)

같은 이름을 가진 두 명의 직원이 있을 때 문제가 발생합니다. § 다음 슬라이드는 정보 손실
방법을 보여줍니다. 원래 직원 관계를 재구성할 수 없으므로 이는 손실 분해입니다.

손실 분해 Lossy Decomposition

F+ = {ID -> 이름, 거리, 도시, 급여}

ID	name	street	city	salary
57766	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000

employee

R1

ID	name
57766	Kim
98776	Kim

R2

name	street	city	salary
Kim	Main	Perryridge	75000
Kim	North	Hampton	67000

natural join

ID	name	street	city	salary
57766	Kim	Main	Perryridge	75000
57766	Kim	North	Hampton	67000
98776	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000

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

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

무손실 분해 Lossless Decomposition

§ R을 관계 스키마라고 하고 R1 과 R2가 R의 분해를 형성하도록 합니다.
즉 R = R1 U R2

§ R을 2개의 관계 스키마 R1 로 대체하여 정보의 손실이 없다면 무손실 분해 라고 한다.

유 R2

§ 공식적으로,

그 R1 (r) ⋈ R2 (r) = r

§ 그리고 반대로 분해는 다음과 같은 경우 손실이 있습니다.

r ì õ R1 (r) ⋈ R2 (r) = r

무손실 분해의 예 Example of Lossless Decomposition

§ R의 분해 = (A, B, C)

R1 = (A, B) R2 = (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

정규화 이론 Normalization Theory

§ 특정 관계 R이 "양호한" 형태인지 여부를 결정합니다. § 관계 R이 "양호한" 형태가 아닌 경우 다음과 같이 관계 {R1, R2, ..., Rn} 집합으로 분해합니다. • 각 관계가 양호함 • 분해는 무손실 분해입니다.

§ 우리의 이론은 다음을 기반으로 합니다.

- 기능 종속성
- 다중값 종속성

기능 종속성

§ 일반적으로 실제 데이터에는 다양한 제약(규칙)이 있습니다.
세계.

§ 예를 들어, a에서 유지될 것으로 예상되는 제약 조건 중 일부는
대학 데이터베이스는 다음과 같습니다.

- 학생과 강사는 ID로 고유하게 식별됩니다.
- 각 학생과 강사는 이름이 하나만 있습니다.
- 각 강사와 학생은 (주로) 오직 한 사람과만 연결됩니다.
  부서.
- 각 부서는 예산에 대한 값이 하나뿐이고 관련 건물도 하나뿐입니다.

기능 종속성(계속)

§ 이러한 모든 실제 제약 조건을 충족하는 관계의 인스턴스는 다음과 같습니다.
관계의 법적 인스턴스 라고 합니다 .

§ 데이터베이스의 법적 인스턴스는 모든 관계 인스턴스가 법적 인스턴스인 경우입니다. § 법적
관계 집합에 대한
제약. § 특정 속성 집합에 대한 값이 고유하게 결정하
도록 요구
다른 속성 세트의 값.

§ 기능 종속성은 키 개념의 일반화입니다.

기능 종속성 정의 Functional Dependencies Definition

§ R을 관계 스키마라고 하자

a Í R 및 § 기 비 R에서

능 종속성

® 비

는 R을 유지하는 경우에만 법적 관계 r(R)에 대해 r의 두 튜플 t1 과 t2가 속성 a에 동의할 때마다 속성 에도 동의합니다. 그건,

비

t1[a] = t2 [a] Þ t1[비] = t2 [비]

§ 예: 다음 r 인스턴스가 있는 r(A,B)를 고려하십시오.

1	4
5	3
1	7

§ 이 경우 B ® A 홀드; A ® B는 유지되지 않습니다 .

기능 종속성 집합의 폐쇄 Closure of Functional Dependencies

§ 기능적 종속성 세트 F가 주어지면 F에 의해 논리적으로 암시되는 특정 다른 기능적 종속성이 있습니다.

- A ® B이고 B ® C이면 A ® C라고 추론할 수 있습니다.
- 등

§ F가 논리적으로 암시하는 모든 기능 종속성 집합은 F의 클로저 입니다.

§ 우리는 F에 의한 F의 폐쇄를 F+ 로 나타냅니다 .

키 및 기능 종속성

§ K는 K ® R의 경우에만 관계 스키마 R의 수퍼키입니다. § K는 다음과 같은 경우에만 R의 후보 키입니다.

- K ® R, 그리고
- a가 없는 경우 Ì K, a ® R

§ 기능 종속성을 통해 슈퍼키를 사용하여 표현할 수 없는 제약 조건을 표현할 수 있습니다. 스키마를 고려하십시오.

in_dep (ID, 이름, 급여, 부서명, 건물, 예산).

다음과 같은 기능 종속성이 유지될 것으로 예상합니다.

dept_name® 건물

건물 ID

그러나 다음이 유지될 것으로 기대하지 않습니다.

dept_name ® 급여

기능 종속성의 사용

§ 기능 종속성을 사용하여 다음을 수행합니다.

- 관계를 테스트하여 주어진 기능 종속성 세트에서 합법적인지 확인합니다.
  - § 관계 r이 기능적 종속성 집합 F에서 합법적이면 r이 F를 만족한다고 말합니다. • 법적 관계 집합에 대한 제약을 지정

하기 위해

§ R에 대한 모든 법적 관계가 기능적 종속 F 집합을 만족하는 경우 F가 R 을 유지한다고 말합니다.

§ 참고: 관계 스키마의 특정 인스턴스는 기능을 충족할 수 있습니다.
기능적 종속성이 모든 법적 인스턴스에 적용되지 않는 경우에도 종속성이 있습니다.

- 예를 들어 강사의 특정 인스턴스는 우연히 이름 ® ID를 충족할 수 있습니다.

사소한 기능 종속성

§ 일부 유형의 기능 종속성(예: A ® A, AB ® A)은 모든 관계에서 충족되는 경우 사소 하다고 합니다.

§ 예:

- 이름 ® 이름
- 아이디, 이름 ® 아이디

§ 일반적으로 a ® 비 사소한 경우 비 안에

무손실 분해

§ 기능 종속성을 사용하여 특정 분해가 무손실인 경우를 표시할 수 있습니다.

§ R = (R1, R2) 의 경우 모든 가능한 관계 r에 대해
스키마 R

r = ÕR1 (r) ÕR2 (r) § 다

음 종속성 중 적어도 하나가 F+ 에 있는 경우 R을 R1 및 R2 로 이진 분해하는 것은 무손실 분해입니다.

- R1 Ç R2 ® R1
- R1 Ç R2 ® R2

§ 위의 기능 종속성은 무손실 조인 분해를 위한 충분 조건입니다. 종속성은 모든 제약 조건이 기능적 종속인 경우에만 필수 조건입니다.

- ((R1 Ç R2 ® R1) (R1 Ç R2 ® R2))인 경우 R에서 R1 및 R2 로의 이진 분해는 무손실 분해입니다 (True).
- R의 R1 및 R2 로의 이진 분해 가 무손실 분해이면 ((R1 Ç R2 ® R1) (R1 Ç R2 ® R2)) (???)

예kample

§ R = (A, B, C)
F = {A ® B, B ® C). § R1
= (A, B), R2 = (B, C)

- 무손실 분해:

R1 Ç R2 = {B} 및 B ® BC

§ R1 = (A, B), R2 = (A, C)

- 무손실 분해:

R1 Ç R2 = {A} 및 A ® AB

§ 참고:

- B ® BC

에 대한 단축 표기법입니다.

- B ® {B, C}

종속성 보존Dependency Preservation

§ 데이터베이스가 업데이트될 때마다 기능 종속성 제약 조건 테스트
업데이트는 비용이 많이 들 수

§ 있습니다. 제약 조건을 효율적으로 테스트할 수 있는 방식으로 데이터베이스를 설계하는 것
이 유용합니다.

§ 하나의 관계만 고려하여 기능적 종속성 을 테스트할 수 있는 경우 이 제약 조건을 테스트하는 비
용이 낮습니다 .

§ 계산적으로 적용하기 어렵게 만드는 분해
기능적 종속성은 NOT 종속성 보존 이라고 합니다 .

종속성 보존 예 Dependency Preservation Example

§ 스키마 고려:

dept_advisor(s_ID, i_ID, dept_name) §

기능 종속성: i_ID ® dept_name s_ID,

dept_name ® i_ID

§ 위의 설계에서는 강사가 dept_advisor 관계에 참여할 때마다 부서 이름을 한 번씩 반복해야 합니다.

§ 이 문제를 해결하려면 dept_advisor를 분해해야 합니다 .

s_ID, 부서명 ® i_ID

§ 따라서 컴포지션은 종속성을 보존하지 않습니다.

§ R1 = {s_ID, i_ID}. R2 = {i_ID, 부서_이름}

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

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

Boyce-Codd 정규형 Boyce-Codd Normal Form

§ 관계 스키마 R은 F+ 의 모든 기능적 종속성이 다음과 같은 형식 이면 기능적 종속성 집합 F에 대해 BCNF에 있습니다.

® 비

여기서 a í R 및 비í R, 다음 보류 중 적어도 하나: í a)

- ® 비 사소합니다(즉, 비
- a는 R의 수퍼키입니다.

Boyce-Codd 정규형(계속) Boyce-Codd Normal Form (Cont.)

§ BCNF에 없는 예제 스키마 :

in_dep (ID, 이름, 급여, 부서명, 건물, 예산)

왜냐하면 :

- dept_name® 건물, 예산 § 보류 in_dep

§ 그러나

- dept_name은 수퍼키가 아닙니다.

§ in_dept를 강사와 부서로 분해할 때

- 강사가 BCNF 소속
  - § 강사 (ID, 이름, 부서명, 급여) F+ = {ID -> 이름, 부서명, 급여}
- 부서는 BCNF에 있음 § 부서(부서
  _이름, 건물, 예산) F+ = {부서_이름® 건물, 예산 }

- $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
- α is a superkey for R

스키마를 BCNF로 분해 Decomposing Schema into BCNF

§ R을 BCNF에 없는 스키마 R이라고 하자. 하자 ® BCNF를 위반합니다. 비 FD가 되십시오

§ 우리는 R을 다음과 같이 분해합니

다. • (a U b) dept_name® 건물, 예산

• (R - (- a비) §

in_dep의 예에서,

- in_dep (ID, 이름, 급여, 부서명, 건물, 예산) a = 부서명 = 건물, 예산 및 in_dep
  - 는 다음으로 대체 •
  - 비(a U b) = (부서명, 건

물, 예산) - a)) = (ID, 이름 ,

dept_name, 급여)

- (R - (비

예kample

§ R = (A, B, C)
F = {A ® B, B ® C} F+ = {A ® B, B ® C, A ® C}

§ R1 = (A, B), R2 = (B, C) • 무손실

조인 분해:

R1 Ç R2 = {B} 및 B ® BC

• 의존성 보존

§ R1 = (A, B), R2 = (A, C) • 무손

실 조인 분해:

R1 Ç R2 = {A} 및 A ® AB • 종속성

보존 아님(R1 R2를 계산하지 않고 B ®
C를 확인할 수 없음) ⋈

BCNF 및 종속성 보존ency Preservation

§ BCNF와 종속성 보존을 모두 달성하는 것이 항상 가능한 것은 아닙니다.

§ 스키마 고려:

dept_advisor(s_ID, i_ID, dept_name)

§ 함수 의존성 포함: R1(i_ID, dept_name), R2(s_ID, i_ID) i_ID ® dept_name s_ID,

dept_name ® i_ID §

dept_advisor는 BCNF에 없음

• i_ID는 수퍼키가 아닙니다. §

dept_advisor의 모든 분해는

dept_advisor의 모든 속성을 포함하지 않습니다.

s_ID, dept_name § ® i_ID

따라서 분해는 종속성 보존이 아닙니다.

• 제 3 정규형 의 동기

제3정규형Normal Form

§ 관계 스키마 R은 다음과 같은 경우 제3정규형(3NF) 입니다.

® 비F+ 에서

다음 홀드 중 적어도 하나는 사소합니다(즉,

- ® 비a는 R의 수퍼키 비 ㄱ)
- 입니다 – a는 R의 후보 키
- 각 속성 A는 베 포함되어 있습니다.

(참고: 각 속성은 다른 후보 키에 있을 수 있음)

§ 릴레이션이 BCNF에 있으면 3NF에 있습니다 (BCNF에서는 위의 처음 두 조건 중 하나가 충족되어야 하므로).

§ 세 번째 조건은 종속성 보존을 보장하기 위해 BCNF를 최소한으로 완화하는 것입니다(이 유는 나중에 참조).

§ BCNF -> 3NF(O) 3NF -> BCNF(X)

3NF 예제3NF Example

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

$\alpha \rightarrow \beta$ in F^+

at least one of the following holds:

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

(**NOTE**: each attribute may be in a different candidate key)

§ 스키마 고려:

dept_advisor(s_ID, i_ID, dept_name) § 기능 종속성 포함: i_ID ® dept_name

s_ID, dept_name ®

i_ID § 두 개의 후보 키 =

{s_ID, dept_name}, {s_ID, i_ID } § 이전에 dept_advisor가 BCNF에 없음 을 확인했습니다. § dept_advisor , 그러나 3NF에 있습니다

- s_ID, dept_name은 수퍼키입니다. • i_ID ® dept_name 및 i_ID는 수퍼키가 아니지만 :
  - § {dept_name} – {i_ID } = {dept_name } 및 §
  - dept_name은 후보 키 {s_ID, dept_name} 에 포함되어 있습니다.

3NF의 이중화

§ 3NF에 있는 아래 스키마 R을 고려하십시오.

- R = (J, K, L) • F = {JK ® L, L ® K } • 인스턴스 테이블:

J	L	K
j_1	l_1	k_1
j_2	l_1	k_1
j_3	l_1	k_1
null	l_2	k_2

§ 테이블에 어떤 문제가 있습니까?

- 정보의 반복 • null 값을 사용해야 함(예: 관계 l2, k2를 나타내기 위해 J에 해당하는 값이 없는 경우)

BCNF와 3NF의 비교

§ BCNF보다 3NF의 이점. 무손실 또는 종속성 보존을 희생하지 않고 3NF 설계를 얻는 것은 항상 가능합니다.

§ 3NF의 단점.

- 데이터 항목 간에 가능한 의미 있는 관계 중 일부를 나타내기 위해 null 값을 사용해야 할 수도 있습니다 .
- 정보의 중복 문제가 있습니다 .

정규화의 목표

§ R을 기능적 종속성 집합 F를 갖는 관계 체계라고 합니다. § 관계 체계 R이 "좋은" 형식인지 결정합니다. § 릴레이션 스킴 R이 "좋은" 형태가 아닌 경우,

다음과 같이 관계 체계 {R1, R2, ..., Rn} 세트로 분해합니다 .

- 각 관계 체계가 양호한 형태입니다 . • 분해가 무손실 분해입니다. • 바람직하게는 분해는 종속성을 보존해야 합니다.

BCNF는 얼마나 좋은가요?

§ BCNF에는 보이지 않는 데이터베이스 스키마가 있습니다.
충분히 정규화

§ 관계를 고려

inst_info (ID, child_name, phone) •

강사가 둘 이상의 전화를 가질 수 있고
여러 자녀

- inst_info의 인스턴스

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

BCNF는 얼마나 좋은가요? (계속)

§ 사소하지 않은 기능 종속성이 없으므로 관계가 BCNF에 있습니다.

§ 삽입 이상 현상 – 즉, 전화 981-992-3443을 99999에 추가하면 두 개의 튜플(99999, David, 981-992-3443)을 추가해야

합니다.
(99999, 윌리엄, 981-992-3443)

고등 정규형

§ inst_info를 다음과 같이 분해하는 것이 좋습니다.

- inst_child:

ID	*child_name*
99999	David
99999	William

- inst_phone:

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

§ 이것은 Fourth와 같은 더 높은 일반 형식의 필요성을 시사합니다.
정규형(4NF), 나중에 살펴보겠습니다.

기능 의존성 이론 로드맵

§ 이제 우리는 주어진 기능적 종속성 집합에 의해 논리적으로 어떤 기능적 종속성이 내포되어 있는지 알려주는 형식 이론을 고려합니다. § 그런 다음 BCNF 및 3NF로 무손실 분해를 생성하는 알고리즘을 개발합

니다.

§ 그런 다음 분해가 종속성인지 테스트하는 알고리즘을 개발합니다.
보존

기능 종속성 집합의 폐쇄

§ 기능적 종속성 세트 F가 주어지면 F에 의해 논리적으로 암시되는 특정 다른 기능적 종속성이 있습니다.

- A ® B이고 B ® C이면 A ® C라고 추론할 수 있습니다.
- 등

§ F가 논리적으로 암시하는 모든 기능 종속성 의 집합은 클로저 입니다.
끄다.

§ 우리는 F에 의한 F의 폐쇄를 F+ 로 나타냅니다 .

기능 종속성 집합의 폐쇄

§ 암스트롱의 식을 반복적으로 적용하여 F+, F의 폐쇄를 계산할 수 있습니다.
공리:

- 재귀 규칙: if 비 Í a, then a ® • 증강 비
  규칙: if a ® then ga ® g 비, 비
- 전이성 규칙: ® § 이 규칙은 비, 그리고 비 ® g 다음에 a ® g

- 사운드 -- 실제로 유지되는 기능적 종속성만 생성합니다.
  그리고
- 완료 -- 유지되는 모든 기능 종속성을 생성합니다.

F+의 예

§ R = (A, B, C, G, H, I)
F = { A ® B
A ® C
CG®에이치
CG® 나
B ® H}

- **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 $\alpha \rightarrow \gamma$

§ F+ 의 일부 구성원

- A ® H
  § A ® B 및 B ® H의 전이성에 의해
- AG ® I
  § A ® C를 G로 증가시켜 AG ® CG를 얻은 다음 CG ® I로 전이성을 얻습니다.
- CG ® HI
  § CG ® CGI를 추론하기 위해 CG ® I를 증가시킴으로써 ,
  CGI ® HI를 추론하기 위한 CG ® H의 증대 ,
  그런 다음 전이성 (CG ® CGI 및 CGI ® HI , 그런 다음 CG ® HI)

기능 종속성 종료(계속)

- **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 $\alpha \rightarrow \gamma$

§ 추가 규칙:

• 합집합 규칙: ®인 경우 비 보류 및 a ® g 보류, 그런 다음 a ® 비 g 보유.

§ 에서 ® 비 증강에 의한 g

§ 증강에 의한 a ® ag

§ ® 비 전이성에 의한 g

• 분해 규칙: a ® g가 성립하는 경우. • 비 g가 유지되면 a ® 비 보류 및 ®

Pseudotransitivity 규칙: a ® g ® d가 유지되는 경 비 보류 및 g 비 ® d가 유지되면 a 우.

§ ® 비 에게 ®에서 비 증강에 의한 g

§ 에서 ® 비이행성에 의한 g 와 g ® d à ag ® d • 위의 규칙은 암스트롱 의 공리에서 유추할 수 있습니다.

F+ 계산 절차

§ 기능 종속성 F 집합의 종료를 계산하려면:

에프 + = F

F+ 의 각

기능 종속성 f 에 대해 반복합니다 . f에 반사성 및 확대 규칙

을 적용합니다. 결과 기능적 종속성을 F에 추가합니다. + 각 쌍 의 기

능적 종속성 f1 및 f2 에 대해 F + if f1 및 f2가 전이성을 사용하여 결합될

수 있는 경우입니다.

그런 다음 F +가 더 이상 변경되지 않을 때까지 결과 기능적 종속성을 F + 에 추

가합니다.

§ 2n x 2n = 22n 가능한 기능 종속성 여기서 n은 R의 속성 수 § 참고: 나중에 이 작업에 대한 대체 절차를 볼 것입니다.

속성 세트의 폐쇄 Closure of Attribute Sets

§ 속성 집합 a가 주어지면, under F(a+로 표시됨) 의 폐쇄를 a under F에 의해 기능적으로 결정되는 속성 집합으로 정의합니다.

§ a+를 계산하는 알고리즘 , F에서 a의 폐쇄

```
결과 := ㄱ;
while (결과로 변경) do for each b ® g
        in F do begin if b Í result then
    result :=
        result È g end
```


속성 세트 클로저의 예 Example of Attribute Set Closure

§ R = (A, B, C, G, H, I) § F = {A ® B
A ® C
CG®에이치
CG® 나
B ® H} §

(AG)+

1. 결과 = AG
2. 결과 = ABCG (A ® C 및 A ® B) 3. 결과 = ABCGH(CG ® H 및 CG Í AGBC) 4. 결과 = ABCGHI(CG ® I 및 CG Í AGBCH)

§ AG는 후보 키입니까?

1. AG는 슈퍼 키입니까?
   1. AG ® R 입니까? == (AG)+ Ê R
2. AG의 하위 집합이 수퍼키입니까?
   1. A ® R 입니까? == (A)+ Ê R 2 입니다 . G ® R 입니까? == Is (G)+ Ê R 3. 일반적으로: 크기 n-1의 각 하위 집합을 확인합니다.

속성 클로저의 사용 Using Attribute Closure

속성 클로저 알고리즘은 여러 가지 용도로 사용됩니다. § 수퍼키 테스트:

- a가 슈퍼키인지 테스트하기 위해 a+를 계산하고 a+가 R의 모든 속성을 포함하는지 확인합니다.

§ 기능 종속성 테스트

- 기능 종속성 a ® b가 유지되는지(즉, F+에 있는지) 확인하려면 b Í a+인지 확인하십시오.
- 즉, 속성 클로저를 사용하여 a+를 계산 한 다음, b를 포함합니다.
- 간단하고 저렴한 테스트이며 매우 유용합니다.

§ F의 컴퓨팅 폐쇄에 대한 대체 방법을 제공합니다.

- 각 g Í R에 대해 클로저 g+를 찾고 각 S Í g+ 에 대해 기능 종속성 g ® S를 출력합니다.

정식 표지 Canonical Cover

§ 관계 스키마에 기능적 종속성 F 집합이 있다고 가정합니다. 사용자가 관계에 대한 업데이트를 수행할 때마다 데이터베이스 시스템은 업데이트가 기능적 종속성을 위반하지 않도록 해야 합니다. 즉, F의 모든 기능 종속성이 새 데이터베이스 상태에서 충족됩니다.

§ 업데이트가 세트 F의 기능 종속성을 위반하는 경우 시스템은 업데이트를 롤백해야 합니다.

§ 테스트를 통해 위반 사항을 확인하는 데 소요되는 노력을 줄일 수 있습니다. 주어진 세트와 동일한 클로저를 갖는 단순화된 기능 종속성 세트. § 이 단순화된 집합을 정규 표지 라고 합니다.

§ 정식 표지를 정의하려면 먼저 외부 속성을 정의해야 합니다 . To define canonical cover we must first define e

- F + 를 변경하지 않고 제거할 수 있는 경우 F의 기능적 종속성 속성은 관련이 없습니다 .

관련 없는 속성 Extraneous Attributes

§ 기능 종속성의 왼쪽에서 특성을 제거하면 더 강력한 제약 조건이 될 수 있습니다.

- 예를 들어 AB ® C가 있고 B를 제거하면 더 강력한 결과 A ® C를 얻습니다. A ® C는 논리적으로 AB ® C를 암시하기 때문에 더 강력할 수 있지만 AB ® C는 그 자체로 논리적으로 그렇지 않습니다. A ® C를 의미

§ 그러나 우리의 기능 종속성 세트 F가 어떻게 되는지에 따라 AB ® C에서 B를 안전하게 제거할 수 있습니다.

- 예를 들어, • F = {AB ® C, A ® D, D ® C}

라고 가정합니다. • 그러면 F가 논리적으로 A ®

C를 암시하므로 B는 AB 에서 관련이 없음을 나타낼 수 있습니다.
® 씨.

외부 속성(계속) Extraneous Attributes (Cont.)

§ 기능 종속성의 오른쪽에서 속성 제거
더 약한 제약 조건 이 될 수 있습니다 .

- 예를 들어 AB ® CD가 있고 C를 제거하면 더 약한 결과 AB ® D를 얻을 수 있습니다. AB ® D만 사용하면 더 이상 AB ® C를 추론할 수 없기 때문에 더 약할 수 있습니다 . § 그러나 무엇에 따라 우리의 기능적 종속성 세트 F가 발생하면 AB ® CD에

서 C를 안전하게 제거할 수 있습니다.

- 예를 들어, F = {AB ® CD, A ® C}라고 가

정합니다. • 그러면 AB ® CD를 AB

® D로 교체한 후에도
여전히 AB ® C 및 AB ® CD를 추론할 수 있습니다.

관련 없는 속성 Extraneous Attributes

§ F의 기능적 종속성의 속성은 F를 변경하지 않고 제거할 수 있는 경우 관련이 없습니다
+ § 기능적 종속성과 기

능적 종속성의 집합 F를 고려하십시오.
® F의 b.

• 왼쪽에서 제거: 특성 A는 if에서 관련이 없습니다 .

§ A à 및

§ F는 논리적으로 (F – {a ® b}) È {(a – A) ® b}를 의미합니다. • 오른쪽에

서 제거: 속성 A는 § A Î b 및 § 기능적 종속성 집합 인 경우 b에서 관련이 없습니다 .

(F – {a ® b}) È {a ®(b – A)}는 논리적으로 F를 의미합니다 .

§ 참고: "강한" 기능적 종속성은 항상 더 약한 종속성을 의미하기 때문에 위의 각 사례에서 반대 방향의 의미
는 사소합니다 . • 왼쪽에서 제거: F – {a ® b}) È {(a – A) ® b}는 논리적으로 F를 의미합니다. • 오
른쪽에서 제거: F는 논리적으로 (F – {a ® b}) È { a ®(b – A)} .

속성이 관련 없는지 테스트 Testing if an Attribute is Extraneous

§ R을 관계 스키마라고 하고 F를 기능적
R을 유지하는 종속성. 기능 종속성 a ® b의 속성을 고려하십시오. § 속성 A Î b가 b에
서 관련이 없는지 테스트

하기 위해

• 세트를 고려하십시오.

F' = (F – {a ® b}) È {a ®(b – A)}, • a+가 F' 아래

에 A를 포함하는지 확인 ; 그렇다면 A는 b에서 관련이 없습니다.

§ 속성 A Î a가

• g = a – {A} 라고 합니다 . F 에서 g ® b를 추론할 수 있는지 확인합니다. §
F 의 종속성을 사용하여 g+를 계산 합니다 . 만약 g+가 b
§ 의 모든 속성을 포함한다면 , A는

불필요한 속성의 예 Examples of Extraneous Attributes

§ F = {AB ® CD, A ® E, E ® C }라고 하자. § C
가 AB ® CD와 관련이 없는지 확인하기 위해:

- F' = {AB ® D, A ® E, E ® C} 에서 AB의 속성 클로저를 계산합니다. • 클로저는 CD를 포함하는 ABCDE입니다. • 이는 C가 관련이 없음을 의미합니다 . •
  AB+ = {A,B,C, D,E} • F' = {AB ® D, A ® E,
  E ® C}

정식 표지 Canonical Cover

F에 대한 정식 커버는 다음 과 같은 종속성 Fc 의 집합입니다.

§ F는 논리적으로 Fc 의 모든 종속성을 의미 하고, § Fc는 논리적으로 F의 모든 종속성을 의미하며, § Fc 의 기능적 종속성은 외부 속성을 포함 하지 않으며 , § Fc 의 기능적 종속성의 각 왼쪽은 고유합니다. 즉, Fc 에는 두 개의 종속성이 없습니다.

- a1 ® b1 및 a2 ® b2는 다음과 같습니다.
- a1 = a2

정식 표지

§ F에 대한 표준 커버를 계산하려면:

Fc = F

반복

합집합 규칙을 사용하여 다음 형식의 F에 있는 모든 종속 항목을 바꿉니다.

a1 ® b1 및 a1 ® b2 와 a1 ® b1 b2

a 또는 b에서 관련 없는 속성이 있는 Fc 의 기능적 종속성 a ® b 찾기

/* 참고: F 가 아닌 Fc를 사용하여 외부 속성 테스트 수행 */

관련 없는 속성이 발견되면 a ® b에서 삭제합니다.

~까지 (Fc는 변경되지 않음)

§ 참고: 유니온 규칙은 일부 관련 없는 특성 이후에 적용될 수 있습니다.
삭제되었으므로 다시 적용해야 합니다.

예제 1: Canonical Cover 계산

§ R = (A, B, C)

F = {A ® BC
B ® C
A ® B
AB ® C}

§ A ® BC와 A ® B를 A ® BC로 결합

- 집합은 이제 {A ® BC, B ® C, AB ® C}입니다.

§ A는 AB ® C에서 관련이 없습니다.

- AB ® C에서 A를 삭제한 결과가 다른 종속성에 의해 암시되는지 확인합니다.

§ 예: 사실 B ® C는 이미 존재합니다!

- 이제 집합은 {A ® BC, B ® C}입니다.

§ C는 A ® BC {A->B, B->C}에서 관련이 없습니다. • A ® C가 A ®

B 및 기타 종속 항목에 의해 논리적으로 암시되는지 확인합니다.

§ 예: A ® B 및 B ® C에서 전이성을 사용합니다.

- 더 복잡한 경우 A의 속성 클로저를 사용할 수 있습니다.

§ 정식 표지는 A ® B입니다.

B ® C

예제 2: Canonical Cover 계산

§ R = {A, B, C, D} §

F = {A ®BC, B ® C, A ® B, AB ® C, AC ® D } • A ®BC 및 A ® B는 A ®BC로 결합됩니다.

• F = {A ®BC, B ® C, AB ® C, AC ® D } • A ®BC의 경우 C는 외부 속성입니다.

• F = {A ®B, B ® C, AB ® C, AC ® D } • AB ® C의 경우 C는 외부 속성이므로 AB ® C는 삭제됩니다. • F = {A ®B, B ® C, 교류 ® D }

• AC ® D의 경우 C는 외부 속성입니다.

§ 이는 F = {A ®B, B ® C, AC ® D } 에서 A+ = {ABCD}이기 때문입니다.

• Fc = {A ®B, B ® C, A ® D } • Fc = {A ®BD, B ® C}.

종속성 보존

§ Fi 를 Ri 의 속성만 포함하는 종속성 F + 의 집합이라고 합니다 . • 분해는 종속성 보존입니다. if (F1 È F2 È ⋯ È Fn)+ = F + § 위의 정의를 사용하여 종속성 보존을 위한 테스트는

기하급수적인 시간.

§ 분해가 종속성 보존이 아닌 경우 기능적 종속성 위반에 대한 업데이트를 확인하는 데 비용이 많이 드는 컴퓨팅 조인이 필요할 수 있습니다.

종속성 보존(계속) Dependency Preservation (Cont.)

§ F를 스키마 R에 대한 종속성 집합이라고 하고 R1, R2 , .., Rn을 R 의 분해라고 합니다 . § Ri 에 대한 F 의 제한 은 리 .

§ 제한의 모든 기능 종속성은 하나의 관계 스키마 속성만 포함하므로 하나의 관계만 확인하여 이러한 종속성을 만족하는지 테스트할 수 있습니다.

§ 제한의 정의는 F +의 모든 종속성을 사용한다는 점에 유의하십시오.
F에 있는 사람들.

§ 제한 집합 F1, F2 , .. , Fn은 효율적으로 확인할 수 있는 기능 종속성 집합입니다.

종속성 보존을 위한 테스트 Testing for Dependency Preservation

§ R을 R1, R2, ⋯, Rn 으로 분해할 때 종속성 a ® b가 유지되는지 확인하기 위해 다음 테스트를 적용합니다(F 에 대해 속성 클로저 수행).

• 결과 = 분해에서
각 Ri 에
대한 반복 t = (결과 Ç Ri)+ Ç Ri 결과 = 결과 È t

때까지 (결과는 변경되지 않음) • 결과가 b의
모든 속성을 포함하는 경우 기능적 종속성 a
® b는 보존됩니다.

§ 분해가 종속성 보존인지 확인하기 위해 F의 모든 종속성에 테스트를 적용합니다.

§ 이 절차는 지수 시간 대신 다항식 시간이 걸립니다.
F+ 및 (F1 IS F2 IS ⋯ IS Fn)+를 계산하는 데 필요합니다.

예kample

§ R = (A, B, C)
F = {A ® B, B ® C}
키 = {A}

§ R은 BCNF에 없음

§ 분해 R1 = (A, B), R2 = (B, C) • BCNF의 R1 및
R2 • 무손실 조인 분해

• 종속성 보존 • F = {A ® B, B ®
C, A ® C}

§ 이전 알고리즘을 사용하여 A ® C가 보존되는지 확인

다음을 이용한 분해 알고리즘ition Using
기능 종속성l Dependencies