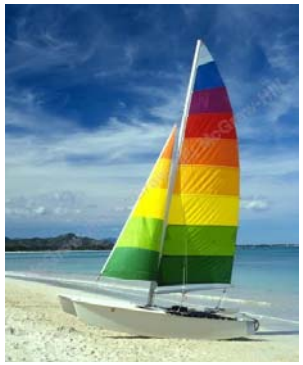

Chapter 8-1:

관계 데이터베이스 설계
(Relational Database Design)

Database System Concepts, 6th Ed.

Chapter 8: Relational Database Design

좋은 릴레이션 설계를 위한 고려 사항

함수 종속 이론

원자형 도메인과 제 1 정규형

함수 종속을 이용한 분해

분해(decomposition)를 위한 알고리즘 (BCNF, 3NF)

다중값 종속을 이용한 분해 (4NF)

다른 정규형들

데이터베이스 설계 절차

스키마 합성하기?

instructor 와 *department* 릴레이션을 합쳐 *inst_dept* 릴레이션을 생성한다고 가정하자

instructor

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

department

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

스키마 합성하기?

instructor 와 *department* 릴레이션을 합쳐 *inst_dept* 릴레이션을 생성:
결과에 정보의 중복 기입 현상이 나타난다.

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

instructor

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

departrment

inst_dept

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

정보의 중복 없이 스킨마 합성하기

두 개의 릴레이션을 합성하여

sec_class(sec_id, building, room_number)

section(course_id, sec_id, semester, year)

다음의 하나의 릴레이션을 생성해 보자

section(course_id, sec_id, semester, year, building, room_number)

이 경우, 정보의 중복이 나타나지 않음.

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

설계 대안: 더 작은 스키마들 ?

만약 우리가 *inst_dept* 스카마에서 시작했다고 가정하면, 어떻게 이 스키마를 *instructor* 와 *department* 스키마로 분해 **(decompose)** 하여야 한다고 알 수 있을까?

다음과 같은 룰이 존재한다.

만약 (*dept_name, building, budget*)의 스카마가 존재하면, 여기에서 *dept_name* 은 후보 키(candidate key) 가 된다.

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

이 때 다음과 같은 함수 종속 (**functional dependency**)이 있다고 표기한다:
dept_name → *building, budget*

inst_dept, 스키마에서 *dept_name* 이 후보키의 조건을 만족시키지 않으므로, building 과 budget 속성값은 중복되어 나타날 수 있다.

이 사실은 *inst_dept* 를 분해하여야 하는 필요성을 나타낸다.

설계 대안: 더 작은 스키마들 ?

모든 분해가 적절한 것은 아니다. *employee(ID, name, street, city, salary)* 를 다음 두 릴레이션으로 분해하는 예를 살펴보자.

employee1 (*ID*, *name*)

employee2 (*name*, *street, city, salary*)

다음 슬라이드에서 정보 손실의 예를 보인다 – 처음 *employee* 릴레이션을 재구성해낼 수 없다 – 즉, 정보 손실이 있는 분해 (**lossy decomposition)**.

A Lossy Decomposition (정보 손실이 있는 분해)

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

employee

employee 1

ID	name
⋮	
57766	Kim
98776	Kim
⋮	

employee 2

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
⋮				

무손실 조인 분해 (Lossless-Join Decomposition)의 예

Lossless join decomposition

Decomposition of $R = (A, B, C)$:

$R_1 = (A, B)$ 과 $R_2 = (B, C)$로 분해

A	B	C
α	1	A
β	2	B

r

A	B
α	1
β	2

$\Pi_{A,B}(r)=r_1$

B	C
1	A
2	B

$\Pi_{B,C}(r)=r_2$

$$\Pi_{A,B}(r) \bowtie \Pi_{B,C}(r) = r$$

A	B	C
α	1	A
β	2	B

목표 — 다음을 위한 이론을 고안한다

특정 릴레이션 R이 좋은 형태인가를 결정

릴레이션 R이 좋은 형태가 아닌 경우, 그것을 아래와 같이 릴레이션의 집합 $\{R_1, R_2, \ldots, R_n\}$으로 분해한다.

- 각 릴레이션은 좋은 형태에 있다
- 분해는 무손실 죠인 분해이다

이론은 다음에 근거한다

- 함수 종속(Functional Dependency)
- 다중값 종속(Multi-Valued Dependency)

함수 종속 (Functional Dependencies, FD)

적법한 릴레이션들의 집합에 대한 제약 조건

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

만약 (*dept_name, building, budget*)의 스카마가 존재하면, 여기에서 *dept_name* 은 후보 키(candidate key) 가 된다.

이 때 다음과 같은 함수 종속 (**functional dependency**)이 있다고 표기한다:
dept_name → *building, budget*

어떤 애트리뷰트 집합의 값이 다른 애트리뷰트 집합의 값을 유일하게 결정하도록 요구한다.

함수적 종속은 키의 개념을 일반화한 것이다.

Functional Dependencies (Cont.)

R을 다음과 같은 릴레이션 스키마라 하자.

$$\alpha \subseteq R, \beta \subseteq R$$

R상에 함수적 종속 $\alpha \rightarrow \beta$가 존재하는 필요 충분 조건은, 어떠한 적법한 릴레이션 r(R)에 대해 r의 두 튜플 t_1과 t_2가 애트리뷰트 α가 같고 또한 애트리뷰트 β가 같을 때이다. 즉, 아래의 경우이다.

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

Example 1: 다음과 같은 인스턴스를 갖는 *r*(A,*B*)를 생각해 보자.
이 경우, $A \rightarrow B$ 는 성립하지 않지만, $B \rightarrow A$ 는 성립한다.

1	4
1	5
3	7

Example 2: *dept_name* → *building, budget*

(*dept_name* → *building*, *dept_name* → *building*)

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Functional Dependencies (Cont.)

K가 릴레이션 스키마 R의 수퍼 키인 필요 충분 조건은 $K \rightarrow R$이다.

K가 R의 후보 키인 필요 충분 조건은

- $K \rightarrow R$ 이고
- for no $\alpha \subset K$, $\alpha \rightarrow R$

$K \subseteq R$ 이라고 가정. K의 값이 각각의 가능한 릴레이션 r(R)의 고유한 튜플을 구분하는데 충분하다면 K는 R의 수퍼 키이다. "가능한 r"이란 모델링하고 있는 조직에 존재할 수 있는 릴레이션 r을 의미한다.

Example: {*ID*} and {ID,name} are both superkeys of *instructor.*

수퍼 키 K가 최소의 조건을 만족시키면 후보 키이다.

Example: {*ID*} is a candidate key for *Instructor*

후보 키 중 하나가 **주키**(primary key)로 선택된다.

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Functional Dependencies (Cont.)

함수 종속은 수퍼 키를 이용해서 표현할 수 없는 제약 조건을 표현하도록 한다. 아래의 스키마를 고려해 보자.

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

아래와 같은 함수 종속의 집합을 가진다고 예상하지만,

dept_name→ building

ID → building

다음과 같은 함수 종속을 가진다고는 예상하지 않는다.

dept_name → salary

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

Functional Dependencies (Cont.)

어떤 함수 종속은 모든 릴레이션에 대하여 만족되기 때문에 자명하다 (**trivial**)고 말한다.

예제*:*

- *ID, name* $\rightarrow$ *ID*
- *name* $\rightarrow$ *name*

일반적으로, 만약 $\beta \subseteq \alpha$ 이면 $\alpha \rightarrow \beta$는 자명한 (당연한) 함수종속이다.

함수 종속의 사용 예

함수 종속을 이용하여 다음 작업을 수행한다.

- 주어진 함수적 종속의 집합 하에 릴레이션들이 적법한가의 여부 검사.
  릴레이션 r이 함수적 종속 집합 F하에서 적법하다면, r은 F를 만족한다고 한다 (*r* **satisfies** *F*).
- 적법한 릴레이션상에 제약 조건을 지정. R상의 모든 적법한 릴레이션이 함수적 종속 집합 F를 만족하면, R상에 F가 존재한다고 한다 (*F* **holds on** *R*).

유의: 함수적 종속이 모든 적법한 사례에 존재하지 않는다 하더라도, 릴레이션 스키마의 특정 사례가 함수적 종속을 만족할 수 있다.

예를 들어, *instructor* -schema의 특정 사례가 경우에 따라 *name* → *ID* 를 만족할 수 있다.

Thank You