

제 7 장



릴레이션 정규화

- 7.1 정규화 개요
- 7.2 함수적 종속성
- 7.3 릴레이션의 분해(decomposition)
- 7.4 제1정규형, 제2정규형, 제3정규형, BCNF
- 7.5 역정규화
 - 연습문제

7장. 릴레이션 정규화

□ 릴레이션 정규화

- ✓ 부주의한 데이터베이스 설계는 제어할 수 없는 데이터 중복을 야기하여 여러 가지 갱신 이상(update anomaly)을 유발함
- ✓ 어떻게 좋은 데이터베이스 설계를 할 것인가? 데이터베이스에 어떤 릴레이션들을 생성할 것인가? 각 릴레이션에 어떤 애트리뷰트들을 둘 것인가?
- ✓ 정규화(normalization)는 주어진 릴레이션 스키마를 함수적 종속성과 기본 키를 기반으로 분석하여, 원래의 릴레이션을 분해함으로써 중복과 세 가지 갱신 이상을 최소화함

7.1 정규화 개요

□ 좋은 관계 데이터베이스 스키마를 설계하는 목적

- ✓ 정보의 중복과 갱신 이상이 생기지 않도록 하고, 정보의 손실을 막으며, 실세계를 훌륭하게 나타내고, 애트리뷰트들 간의 관계가 잘 표현되는 것을 보장하며, 어떤 무결성 제약조건의 시행을 간단하게 하며, 아울러 효율성 측면도 고려하는 것
- ✓ 먼저 갱신 이상이 발생하지 않도록 노력하고, 그 다음에 효율성을 고려함

7.1 정규화 개요(계속)

□ 갱신 이상(update anomaly)

✓ 수정 이상(modification anomaly)

- 반복된 데이터 중에 일부만 수정하면 데이터의 불일치가 발생

✓ 삽입 이상(insertion anomaly)

- 불필요한 정보를 함께 저장하지 않고는 어떤 정보를 저장하는 것이 불가능

✓ 삭제 이상(deletion anomaly)

- 유용한 정보를 함께 삭제하지 않고는 어떤 정보를 삭제하는 것이 불가능

7.1 정규화 개요(계속)

예 : 나쁜 설계 1

그림 7.1과 같은 구조와 내용을 갖는 사원 릴레이션으로부터 설계를 시작한다고 가정해 보자. 그림 7.1의 사원 릴레이션은 회사의 사원에 관한 정보를 저장하는 릴레이션이다. 이 회사에서는 각 사원이 두 개까지의 부서에 속할 수 있다.

사원	사원이름	사원번호	주소	전화번호	부서번호1	부서이름1	부서번호2	부서이름2
	김창섭	2106	우이동	726-5869	1	영업	2	기획
	박영권	3426	사당동	842-4538	3	개발	^	^
	이수민	3011	역삼동	579-4685	2	기획	3	개발

[그림 7.1] 사원 릴레이션

7.1 정규화 개요(계속)

예 : 나쁜 설계 2

그림 7.1 릴레이션 대신에 그림 7.2 릴레이션처럼 설계하면 각 사원마다 부서 수를 제한할 필요가 없다. 그러나 이 설계는 또 다른 단점을 갖고 있다. 이 릴레이션의 단점은 아래와 같다.

사원	사원이름	사원번호	주소	전화번호	부서번호1	부서이름1
	김창섭	2106	우이동	726-5869	1	영업
	김창섭	2106	우이동	726-5869	2	기획
	박영권	3426	사당동	842-4538	3	개발
	이수민	3011	역삼동	579-4685	2	기획
	이수민	3011	역삼동	579-4685	3	개발

[그림 7.2] 사원 릴레이션

7.1 정규화 개요(계속)

□ 정보의 중복

각 사원이 속한 부서 수만큼 동일한 사원의 튜플들이 존재하므로 사원이름, 사원번호, 주소, 전화번호 등이 중복되어 저장 공간이 낭비됨

□ 수정 이상 (modification anomaly)

만일 어떤 부서의 이름이 바뀔 때 이 부서에 근무하는 일부 사원 튜플에서만 부서이름을 변경하면 데이터베이스가 불일치 상태에 빠짐

7.1 정규화 개요(계속)

□ 삽입 이상 (insertion anomaly)

만일 어떤 부서를 신설했는데 아직 사원을 한 명도 배정하지 않았다면 이 부서에 관한 정보를 입력할 수 없음

□ 삭제 이상 (deletion anomaly)

만일 어떤 부서에 속한 사원이 단 한 명이 있는데, 이 사원에 관한 튜플을 삭제하면 이 사원이 속한 부서에 관한 정보도 릴레이션에서 삭제됨

수정 이상, 삽입 이상, 삭제 이상을 총칭하여 갱신 이상 (update anomaly)이라 함

7.1 정규화 개요(계속)

□ 릴레이션 분해

- ✓ 하나의 릴레이션을 두 개 이상의 릴레이션으로 나누는 것
- ✓ 릴레이션의 분해는 필요한 경우에는 분해된 릴레이션들로부터 원래의 릴레이션을 다시 구할 수 있음을 보장해야 한다는 원칙을 기반
- ✓ 분해를 잘못하면 두 릴레이션으로부터 얻을 수 있는 정보가 원래의 릴레이션이 나타내던 정보보다 적을 수도 있고 많을 수도 있음
- ✓ 릴레이션의 분해는 릴레이션에 존재하는 함수적 종속성에 관한 지식을 기반으로 함

7.1 정규화 개요(계속)

예: 릴레이션 분해

그림 7.2의 사원 릴레이션을 그림 7.3의 사원1 릴레이션과 부서 릴레이션으로 분해한다.

사원1	사원이름	사원번호	주소	전화번호	부서번호
	김창섭	2106	우이동	726-5869	1
	김창섭	2106	우이동	726-5869	2
	박영권	3426	사당동	842-4538	3
	이수민	3011	역삼동	579-4685	2
	이수민	3011	역삼동	579-4685	3

부서	부서번호	부서이름
	1	영업
	2	기획
	3	개발

[그림 7.3] 그림 6.2의 사원 릴레이션을 사원1 릴레이션과 부서 릴레이션으로 분해

7.1 정규화 개요(계속)

□ 그림 7.2에서 예를 들었던 갱신 이상 문제는 아래와 같이 해결됨

✓ 부서이름의 수정

- 어떤 부서에 근무하는 사원이 여러 명 있더라도 사원1 릴레이션에는 부서 이름이 포함되어 있지 않으므로 수정 이상이 나타나지 않음

✓ 새로운 부서를 삽입

- 만일 어떤 신설 부서에 사원이 한 명도 배정되지 않았더라도, 부서 릴레이션의 기본 키가 부서번호이므로 이 부서에 관한 정보를 부서 릴레이션에 삽입할 수 있음

✓ 마지막 사원 투플을 삭제

- 만일 어느 부서에 속한 유일한 사원에 관한 투플을 삭제하더라도 이 부서에 관한 정보는 부서 릴레이션에 남아 있음

7.1 정규화 개요(계속)

□ 정규형(normal form)의 종류

- ✓ 제1정규형(first normal form), 제2정규형(second normal form), 제3정규형(third normal form), BCNF(Boyce–Codd normal form), 제4정규형(fourth normal form), 제5정규형(fifth normal form)
- ✓ 일반적으로 산업계의 데이터베이스 응용에서 데이터베이스를 설계할 때 BCNF까지만 고려함

7.1 정규화 개요(계속)

❑ 관계 데이터베이스 설계의 비공식적인 지침

- ✓ 지침 1: 이해하기 쉽고 명확한 스키마를 만들라
 - 여러 엔티티 타입이나 관계 타입에 속한 애트리뷰트들을 하나의 릴레이션에 포함시키지 않음

학생_학과

학생번호	학과이름	학과전화번호	과목번호	성적
------	------	--------	------	----

- ✓ 지침 2: 널값을 피하라
- ✓ 지침 3: 가짜 투플이 생기지 않도록 하라
- ✓ 지침 4: 스키마를 정제하라

7.2 함수적 종속성

□ 함수적 종속성의 개요

- ✓ 정규화 이론의 핵심
- ✓ 릴레이션의 애트리뷰트들의 의미로부터 결정됨
- ✓ 릴레이션 스키마에 대한 주장이지, 릴레이션의 특정 인스턴스에 대한 주장이 아님
- ✓ 릴레이션의 가능한 모든 인스턴스들이 만족해야 함
- ✓ 실세계에 대한 지식과 응용의 의미를 기반으로 어떤 함수적 종속성들이 존재하는가를 파악해야 함
- ✓ 함수적 종속성은 제2정규형부터 BCNF까지 적용됨

7.2 함수적 종속성(계속)

□ 결정자(determinant)

- ✓ 어떤 애트리뷰트의 값은 다른 애트리뷰트의 값을 고유하게 결정할 수 있음
- ✓ 그림 7.4의 사원 릴레이션에서 사원번호는 사원이름을 고유하게 결정함
- ✓ 주소는 사원이름을 고유하게 결정하지 못함
- ✓ 결정자는 주어진 릴레이션에서 다른 애트리뷰트(또는 애트리뷰트들의 집합)를 고유하게 결정하는 하나 이상의 애트리뷰트를 의미
- ✓ 결정자를 아래와 같이 표기하고, 이를 “A가 B를 결정한다”(또는 “A는 B의 결정자이다”)라고 말함

$A \rightarrow B$

7.2 함수적 종속성(계속)

사원	<u>사원번호</u>	사원이름	주소	전화번호	직책	<u>부서번호</u>	부서이름
	4257	정미림	홍제동	731-3497	팀장	1	홍보
	1324	이범수	양재동	653-7412	프로그래머	2	개발
	1324	이범수	양재동	653-7412	웹 디자이너	1	홍보
	3609	안명석	양재동	425-8520	팀장	3	홍보

[그림 7.4] 사원 릴레이션

사원번호 → 사원이름

사원번호 → 주소

사원번호 → 전화번호

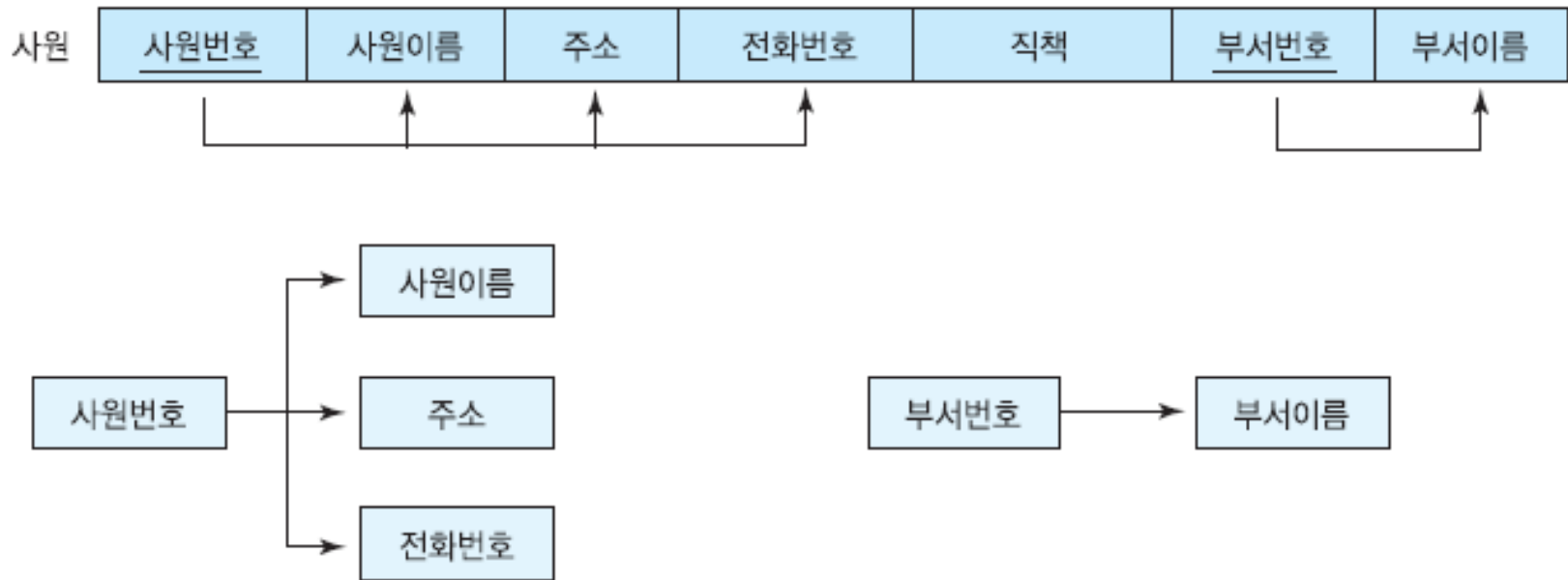
부서번호 → 부서이름

7.2 함수적 종속성(계속)

□ 함수적 종속성

- ✓ 만일 애트리뷰트 A가 애트리뷰트 B의 결정자이면 B가 A에 함수적으로 종속한다고 말함
- ✓ 다른 말로 표현하면, 주어진 릴레이션 R에서 애트리뷰트 B가 애트리뷰트 A에 함수적으로 종속하는 필요 충분 조건은 각 A 값에 대해 반드시 한 개의 B 값이 대응된다는 것
- ✓ 예: 사원번호가 사원이름, 주소, 전화번호의 결정자이므로 사원이름, 주소, 전화번호는 사원번호에 함수적으로 종속
- ✓ 예: 직책은 (사원번호, 부서번호)에 함수적으로 종속하지, 사원번호에 함수적으로 종속하지는 않음

7.2 함수적 종속성(계속)



[그림 7.5] 사원 릴레이션의 함수적 종속성의 두 가지 다이어그램

7.2 함수적 종속성(계속)

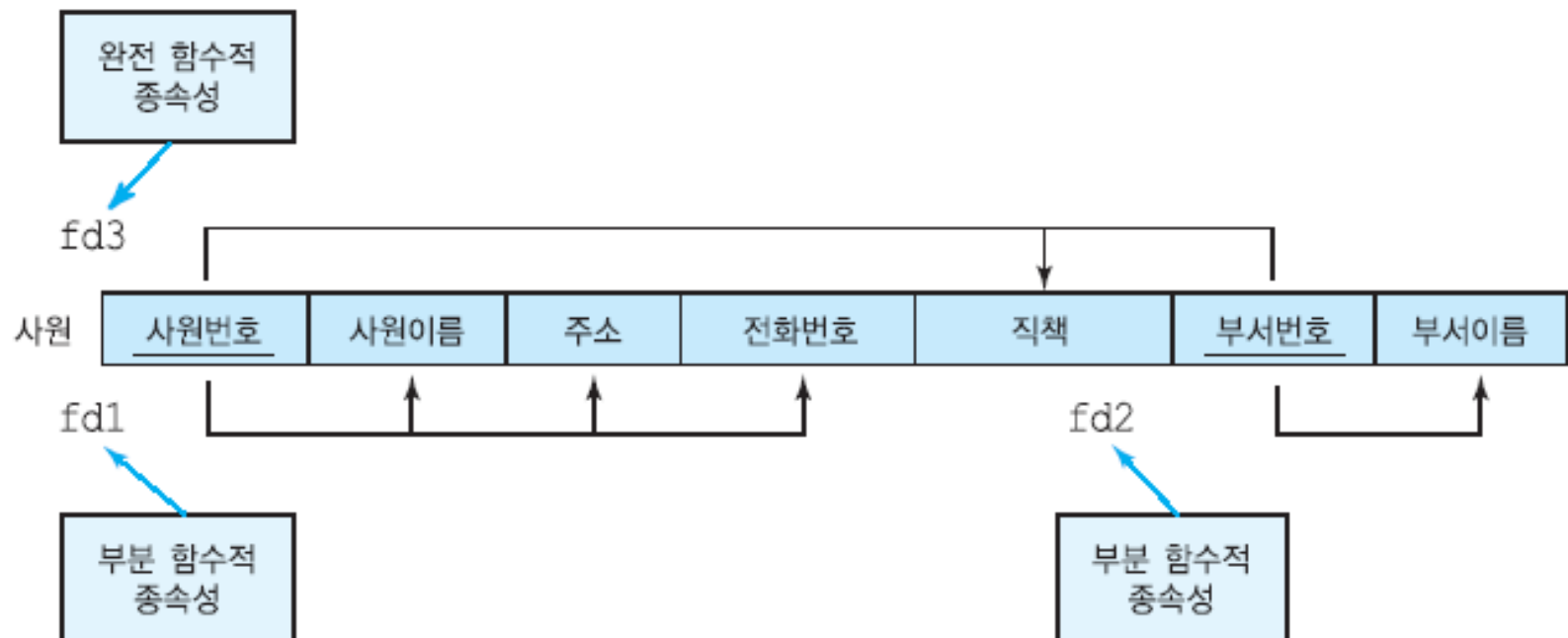
□ 완전 함수적 종속성(FFD: Full Functional Dependency)

- ✓ 주어진 릴레이션 R에서 애트리뷰트 B가 애트리뷰트 A에 함수적으로 종속하면서 애트리뷰트 A의 어떠한 진부분 집합에도 함수적으로 종속하지 않으면 애트리뷰트 B가 애트리뷰트 A에 완전하게 함수적으로 종속한다고 말함
- ✓ 여기서 애트리뷰트 A는 복합 애트리뷰트

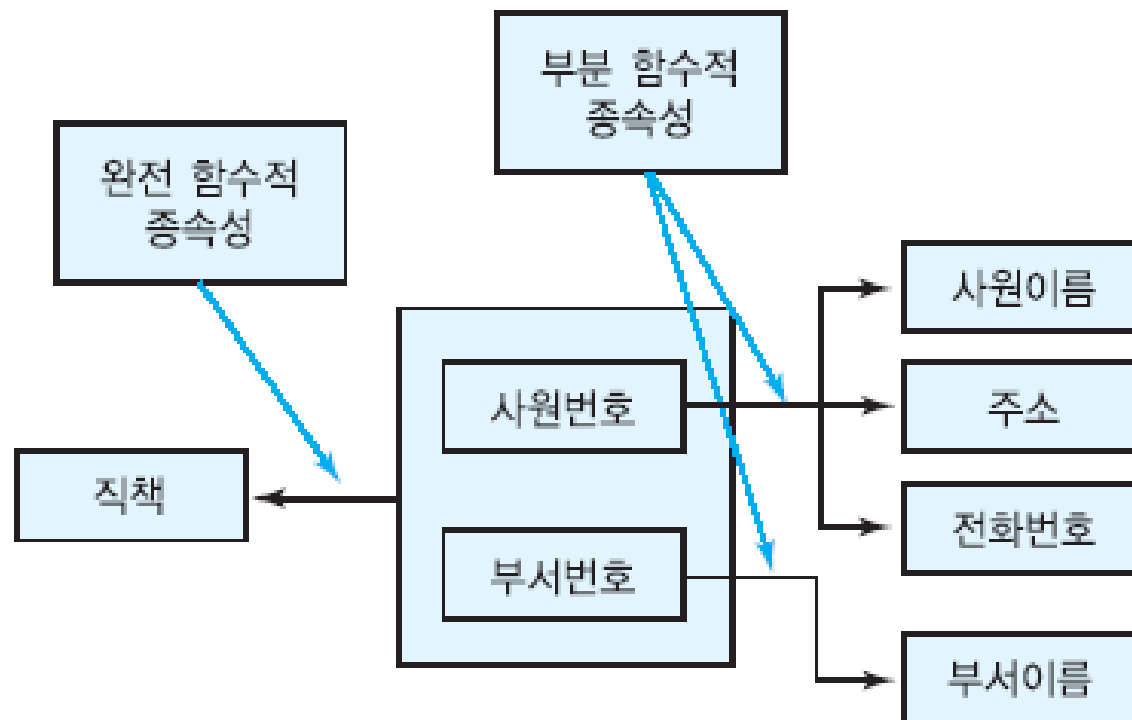
7.2 함수적 종속성(계속)

예 : 완전 함수적 종속성과 부분 함수적 종속성

그림 7.6에서 fd3은 완전 함수적 종속성을 나타내고, fd1과 fd2는 부분 함수적 종속성을 나타낸다.



7.2 함수적 종속성(계속)



[그림 7.6] 완전 함수적 종속성과 부분 함수적 종속성

7.2 함수적 종속성(계속)

□ 이행적 함수적 종속성(transitive FD)

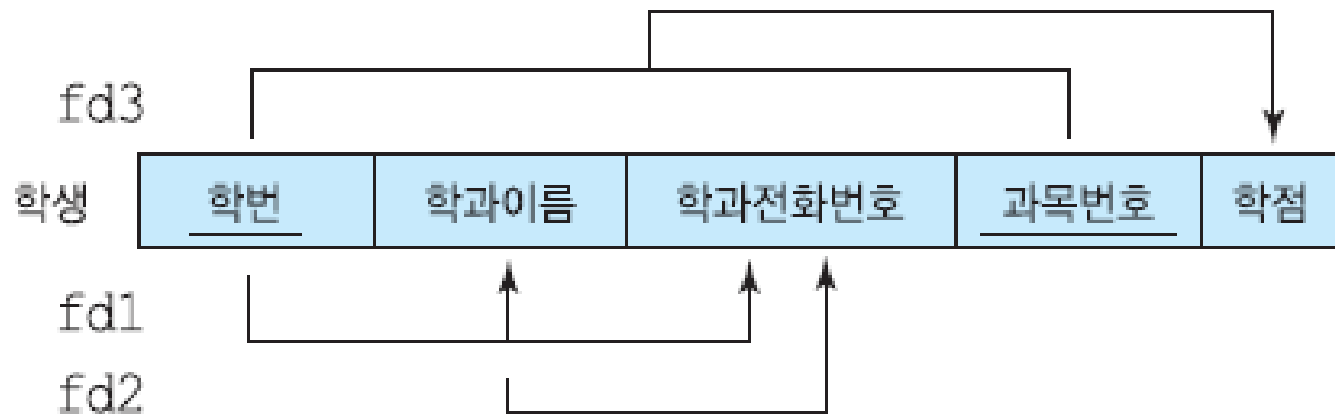
- ✓ 한 릴레이션의 애트리뷰트 A, B, C가 주어졌을 때 애트리뷰트 C가 이행적으로 A에 종속한다($A \rightarrow C$)는 것의 필요 충분 조건은

$$A \rightarrow B \wedge B \rightarrow C$$

가 성립하는 것

- ✓ A가 릴레이션의 기본 키라면 키의 정의에 따라 $A \rightarrow B$ 와 $A \rightarrow C$ 가 성립. 만일 C가 A외에 B에도 함수적으로 종속한다면 C는 A에 직접 함수적으로 종속하면서 B를 거쳐서 A에 이행적으로 종속

7.2 함수적 종속성(계속)



[그림 7.7] 이행적 함수적 종속성

7.3 릴레이션 분해

□ 릴레이션 분해

- ✓ 하나의 릴레이션을 두 개 이상의 릴레이션으로 나누는 것
- ✓ 릴레이션을 분해하면 중복이 감소되고 갱신 이상이 줄어드는 장점이 있는 반면에, 바람직하지 않은 문제들을 포함하여 몇 가지 잠재적인 문제들을 야기할 수 있음
 - 릴레이션이 분해되기 전에는 조인이 필요 없는 질의가 분해 후에는 조인을 필요로 하는 질의로 바뀔 수 있음
 - 분해된 릴레이션들을 사용하여 원래 릴레이션을 재구성하지 못할 수 있음

7.3 릴레이션 분해(계속)

□ 무손실 분해(lossless decomposition)

- ✓ 분해된 두 릴레이션을 조인하면 원래의 릴레이션에 들어 있는 정보를 완전하게 얻을 수 있음
- ✓ 여기서 손실이란 정보의 손실을 뜻함
- ✓ 정보의 손실은 원래의 릴레이션을 분해한 후에 생성된 릴레이션들을 조인한 결과에 들어 있는 정보가 원래의 릴레이션에 들어 있는 정보보다 적거나 많은 것을 모두 포함

7.3 릴레이션 분해(계속)

학생	<u>학번</u>	이름	이메일	<u>과목번호</u>	학점
	11002	이홍근	sea@hanmail.net	CS310	A0
	11002	이홍근	sea@hanmail.net	CS313	B+
	24036	김순미	smkim@venus.uos.ac.kr	CS345	B0
	24036	김순미	smkim@venus.uos.ac.kr	CS310	A+

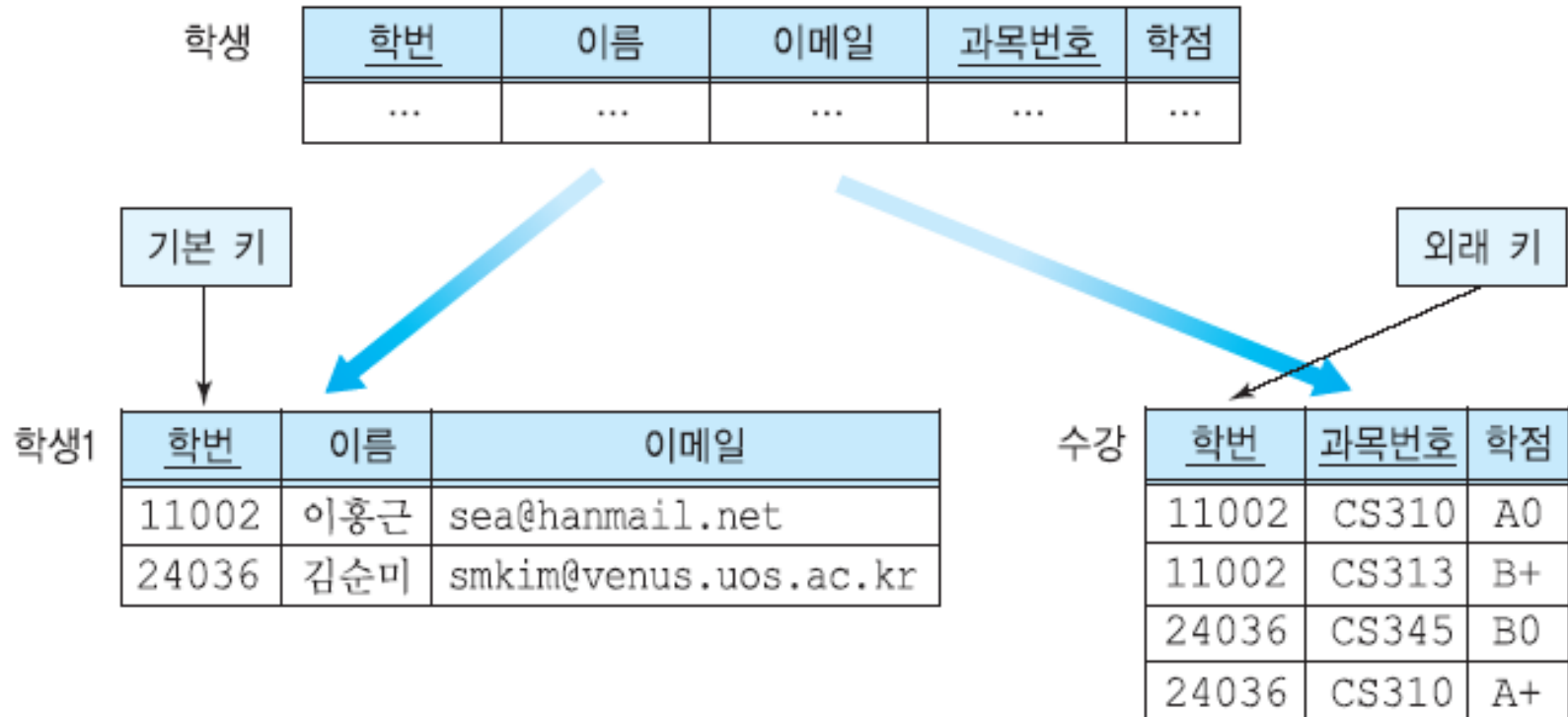
[그림 7.8] 학생 릴레이션

학번 → 이름, 이메일

이메일 → 학번, 이름

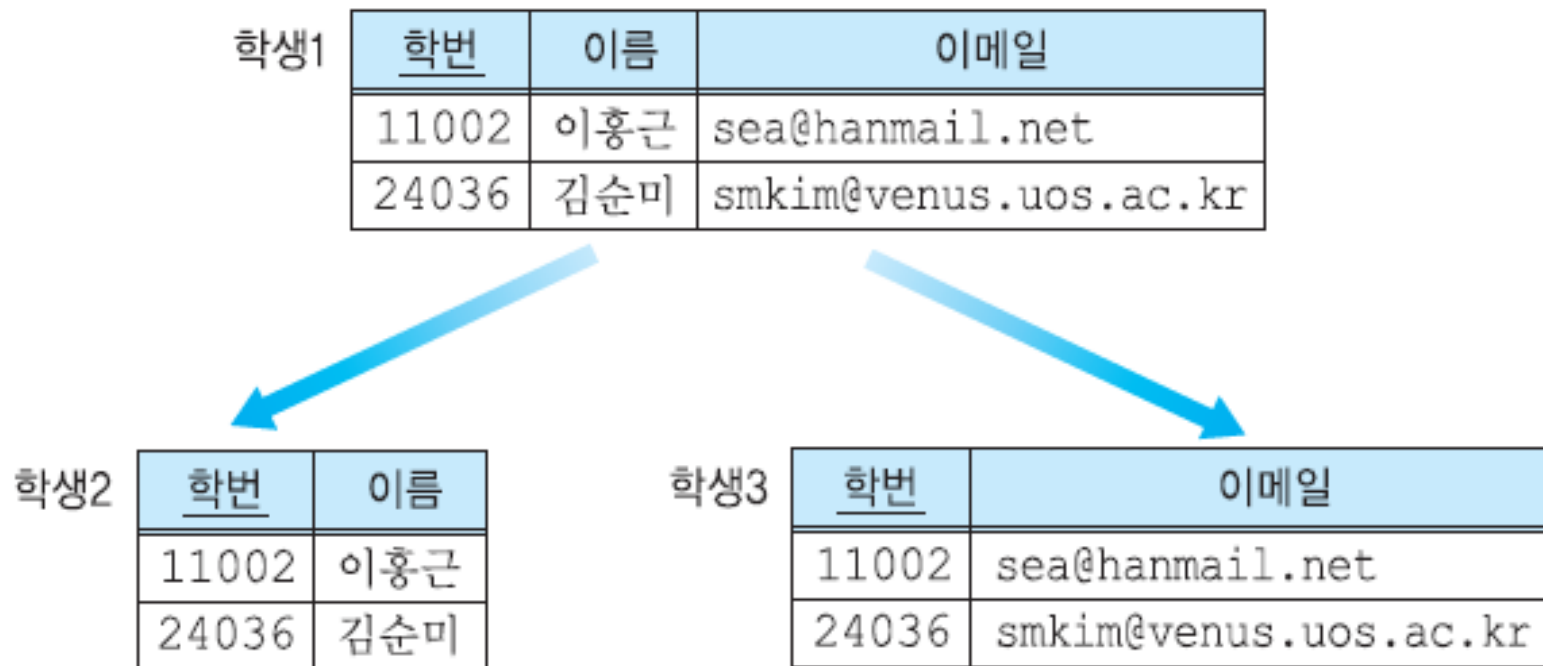
(학번, 과목번호) → 학점

7.3 릴레이션 분해(계속)



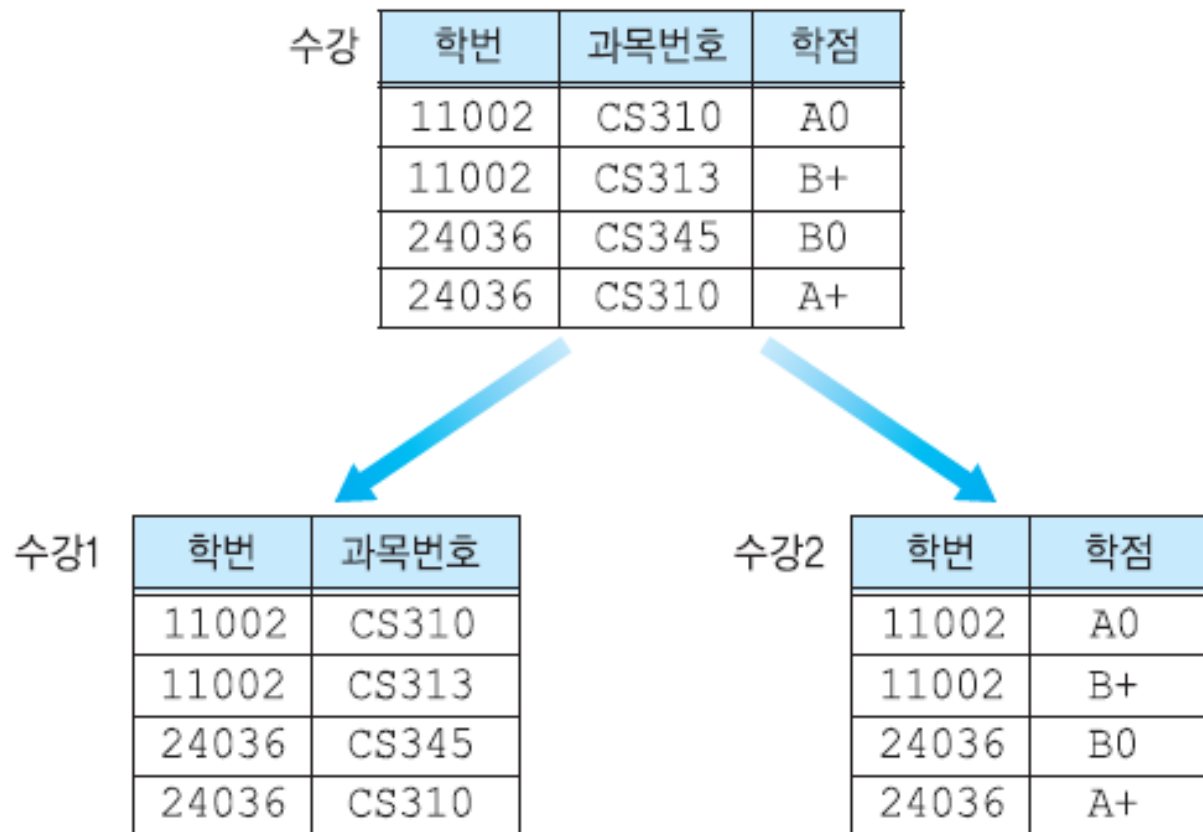
[그림 7.9] 학생 릴레이션을 두 릴레이션으로 분해

7.3 릴레이션 분해(계속)



[그림 7.10] 불필요한 분해

7.3 릴레이션 분해(계속)



[그림 7.11] 나쁜 분해

7.3 릴레이션 분해(계속)

예 : 가짜 튜플

그림 7.11의 수강1 릴레이션과 수강2 릴레이션을 학번, 과목번호, 학점의 세 속성을 사용하여 자연 조인하면 그림 7.12와 같은 결과를 얻는다. 이 릴레이션에서 파란색으로 표시한 튜플들은 그림 7.11의 원래 릴레이션인 수강 릴레이션에 존재하지 않는 튜플들이므로 가짜 튜플에 해당한다.

학번	과목번호	학점
11002	CS310	A0
11002	CS310	B+
11002	CS313	A0
11002	CS313	B+
24036	CS345	B0
24036	CS345	A+
24036	CS310	B0
24036	CS310	A+

[그림 7.12] 가짜 튜플

7.4 제1정규형, 제2정규형, 제3정규형, BCNF

□ 제1정규형

- ✓ 한 릴레이션 R이 제1정규형을 만족할 필요 충분 조건은 릴레이션 R의 모든 애트리뷰트가 원자값만을 갖는다는 것
- ✓ 즉 릴레이션의 모든 애트리뷰트에 **반복 그룹**(repeating group)이 나타나지 않으면 제1정규형을 만족함

학생	학번	이름	과목번호	주소
	11002	이홍근	{CS310, CS313}	우이동
	24036	김순미	{CS310, CS345}	양재동

[그림 7.13] 반복 그룹

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)

❑ 제1정규형을 만족하지 않는 그림 7.13을 제1정규형으로 변환하는 방법

- ✓ 반복 그룹 애트리뷰트에 나타나는 집합에 속한 각 값마다 하나의 튜플로 표현

학생	<u>학번</u>	이름	<u>과목번호</u>	주소
	11002	이홍근	CS310	우이동
	11002	이홍근	CS313	우이동
	24036	김순미	CS345	양재동
	24036	김순미	CS310	양재동

[그림 7.14] 애트리뷰트에 원자값만 있는 릴레이션

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)

❑ 제1정규형을 만족하지 않는 그림 7.13을 제1정규형으로 변환하는 방법(계속)

- ✓ 모든 반복 그룹 애트리뷰트들을 분리해서 새로운 릴레이션에 넣음. 원래 릴레이션의 기본 키를 새로운 릴레이션에 애트리뷰트로 추가함

학생1

<u>학번</u>	이름	주소
11002	이홍근	우이동
24036	김순미	양재동

수강

<u>학번</u>	<u>과목번호</u>
11002	CS310
11002	CS313
24036	CS345
24036	CS310

[그림 7.15] 두 릴레이션으로 분해

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)

□ 제1정규형에 존재하는 갱신 이상

- ✓ 그림 7.16의 학생 릴레이션은 모든 애트리뷰트가 원자값을 가지므로 제1정규형을 만족함
- ✓ 이 릴레이션의 기본 키는 (학번, 과목번호)

학생	학번	학과이름	학과전화번호	과목번호	학점
	11002	컴퓨터과학	210-2261	CS310	A0
	11002	컴퓨터과학	210-2261	CS313	B0
	24036	정보통신	210-2585	IC214	B+

[그림 7.16] 제1정규형을 만족하는 릴레이션

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)

□ 수정 이상

- ✓ 한 학과에 소속한 학생 수만큼 그 학과의 전화번호가 중복되어 저장되므로 여러 학생이 소속된 학과의 전화번호가 변경되었을 때 그 학과에 속한 모든 학생들의 튜플에서 전화번호를 수정하지 않으면 데이터베이스의 일관성이 유지되지 않음

□ 삽입 이상

- ✓ 한 명의 학생이라도 어떤 학과에 소속되지 않으면 이 학과에 관한 튜플을 삽입할 수 없음. 왜냐하면 학번이 기본 키의 구성요소인데 엔티티 무결성 제약조건에 따라 기본 키에 널값을 입력할 수 없기 때문

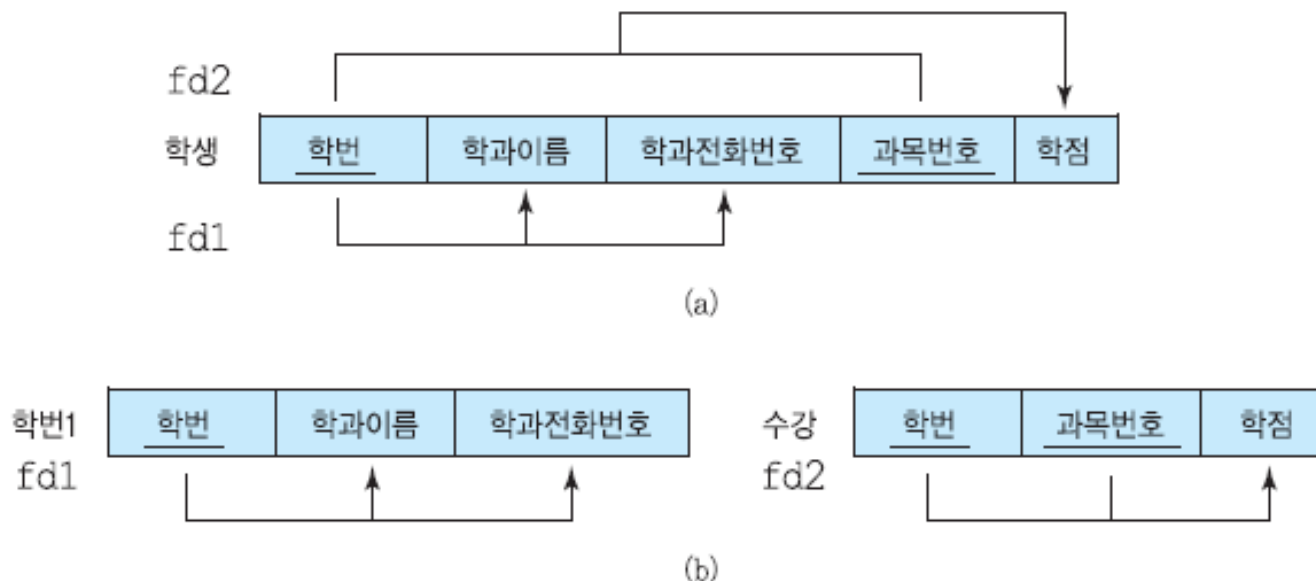
□ 삭제 이상

- ✓ 어떤 학과에 소속된 마지막 학생 튜플을 삭제하면 이 학생이 소속된 학과에 관한 정보도 삭제됨

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)

❑ 갱신 이상이 생기는 이유

- ✓ 기본 키에 대한 부분 함수적 종속성이 학생 릴레이션에 존재함



[그림 7.17] (a) 부분 함수적 종속성이 존재하는 릴레이션(제1정규형)
(b) 부분 함수적 종속성이 존재하지 않도록 분해된 두 릴레이션(제2정규형)

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)

□ 제2정규형

- ✓ 한 릴레이션 R이 제2정규형을 만족할 필요 충분 조건은 릴레이션 R이 제1정규형을 만족하면서, 어떤 후보 키에도 속하지 않는 모든 애트리뷰트들이 R의 기본 키에 완전하게 함수적으로 종속하는 것
- ✓ 기본 키가 두 개 이상의 애트리뷰트로 구성되었을 경우에만 제1정규형이 제2정규형을 만족하는가를 고려할 필요가 있음

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)

□ 제2정규형에 존재하는 갱신 이상

- ✓ 그림 7.18의 학생1 릴레이션의 기본 키는 한 애트리뷰트인 학번이므로 제2정규형을 만족함

학생1	<u>학번</u>	학과이름	학과전화번호
	11002	컴퓨터과학	210-2261
	24036	정보통신	210-2585
	11048	컴퓨터과학	210-2261

[그림 7.18] 제2정규형을 만족하는 릴레이션

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)

□ 수정 이상

- ✓ 여러 학생이 소속된 학과의 전화번호가 변경되었을 때 그 학과에 속한 모든 학생들의 투플에서 전화번호를 수정하지 않으면 데이터베이스의 일관성이 유지되지 않음

□ 삽입 이상

- ✓ 어떤 학과를 신설해서 아직 소속 학생이 없으면 그 학과의 정보를 입력할 수 없다. 왜냐하면 학번이 기본 키인데 엔티티 무결성 제약조건에 따라 기본 키에 널값을 입력할 수 없기 때문

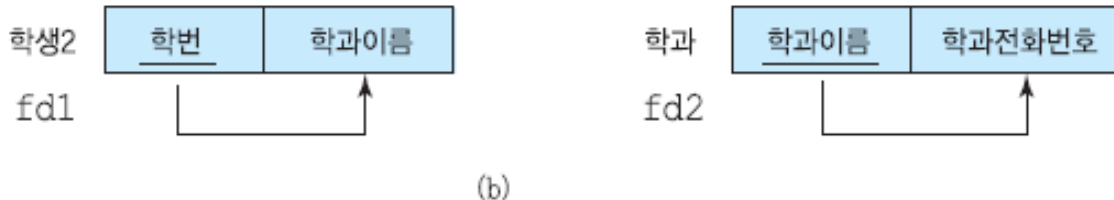
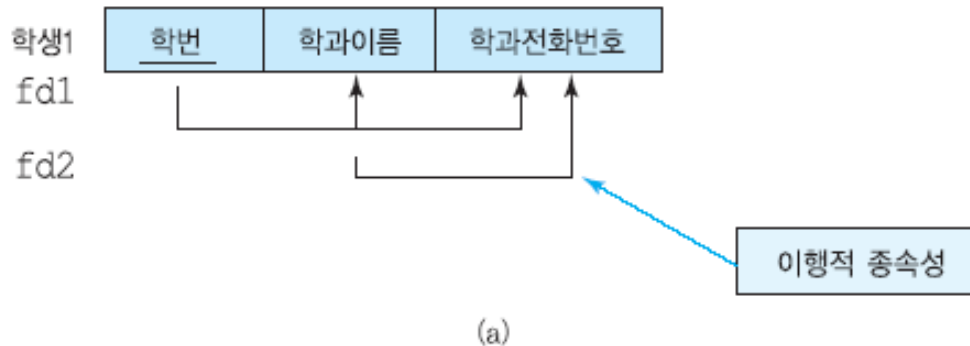
□ 삭제 이상

- ✓ 어떤 학과에서 마지막 학생의 투플이 삭제되면 그 학과의 전화번호도 함께 삭제됨

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)

❑ 갱신 이상이 생기는 이유

- ✓ 학생1 릴레이션에 이행적 종속성이 존재하기 때문



[그림 7.19] (a) 이행적 종속성이 존재하는 릴레이션(제2정규형)

(b) 이행적 종속성이 존재하지 않도록 분해된 두 릴레이션(제3정규형)

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)

□ 제3정규형

- ✓ 한 릴레이션 R이 제3정규형을 만족할 필요 충분 조건은 릴레이션 R이 제2정규형을 만족하면서, 키가 아닌 모든 애트리뷰트가 릴레이션 R의 기본 키에 이행적으로 종속하지 않는 것

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)

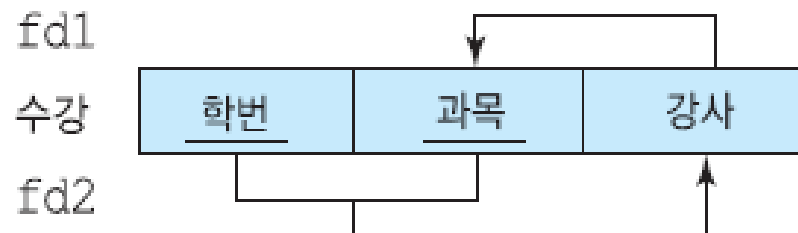
□ 제3정규형에 존재하는 갱신 이상

- ✓ 그림 7.20의 수강 릴레이션에서 각 학생은 여러 과목을 수강할 수 있고, 각 강사는 한 과목만 가르침. 이 릴레이션의 기본 키는 (학번, 과목)
- ✓ 키가 아닌 강사 애트리뷰트가 기본 키에 완전하게 함수적으로 종속하므로 제2정규형을 만족하고, 키가 아닌 강사 애트리뷰트가 기본 키에 직접 종속하므로 제3정규형도 만족함
- ✓ 이 릴레이션에는 아래와 같은 함수적 종속성들이 존재함
(학번, 과목) → 강사
강사 → 과목

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)

수강	<u>학번</u>	<u>과목</u>	강사
	11002	데이터베이스	이영준
	11002	운영 체제	고성현
	24036	자료 구조	엄영지
	24036	데이터베이스	조민형
	11048	데이터베이스	이영준

[그림 7.20] 제3정규형을 만족하는 릴레이션



[그림 7.21] 수강 릴레이션에 존재하는 함수적 종속성

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)

□ 수정 이상

- ✓ 여러 학생이 수강 중인 어떤 과목의 강사가 변경되었을 때 그 과목을 수강하는 모든 학생들의 투플에서 강사를 수정하지 않으면 데이터베이스의 일관성이 유지되지 않음

□ 삽입 이상

- ✓ 어떤 과목을 신설하여 아직 수강하는 학생이 없으면 어떤 강사가 그 과목을 가르친다는 정보를 입력할 수 없음
- ✓ 왜냐하면 학번이 기본 키를 구성하는 애트리뷰트인데 엔티티 무결성 제약조건에 따라 기본 키를 구성하는 애트리뷰트에 널값을 입력할 수 없기 때문

□ 삭제 이상

- ✓ 어떤 과목을 이수하는 학생이 한 명밖에 없는데 이 학생의 투플을 삭제하면 그 과목을 가르치는 강사에 관한 정보도 함께 삭제됨

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)

□ 갱신 이상이 생기는 이유

- ✓ 수강 릴레이션에서 키가 아닌 애트리뷰트가 다른 애트리뷰트를 결정하기 때문
- ✓ 이 릴레이션의 후보 키는 (학번, 과목)과 (학번, 강사)

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)

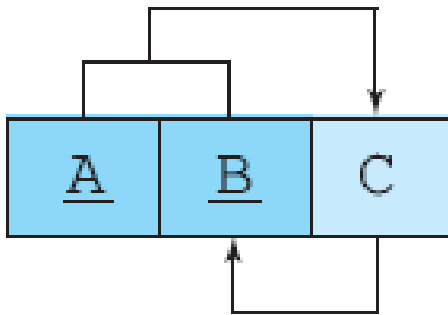
□ BCNF

- ✓ 한 릴레이션 R이 BCNF를 만족할 필요 충분 조건은 릴레이션 R이 제3정규형을 만족하고, 모든 결정자가 후보 키이어야 함
- ✓ 위의 수강 릴레이션에서 강사 애트리뷰트는 후보 키가 아님에도 불구하고 과목 애트리뷰트를 결정하기 때문에 BCNF가 아님
- ✓ 제3정규형을 만족하는 대부분의 릴레이션들은 BCNF도 만족함
- ✓ 하나의 후보 키만을 가진 릴레이션이 제3정규형을 만족하면 동시에 BCNF도 만족함

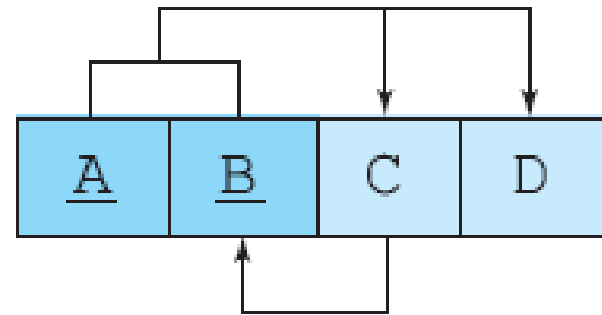
□ BCNF 만드는 방법

- ✓ 제3정규형을 만족하는 릴레이션을 BCNF으로 정규화하려면 키가 아니면서 결정자 역할을 하는 애트리뷰트와 그 결정자에 함수적으로 종속하는 애트리뷰트를 하나의 테이블에 넣음. 이 릴레이션에서 결정자는 기본 키가 됨
- ✓ 그 다음에는 기존 릴레이션에 결정자를 남겨서 기본 키의 구성요소가 되도록 함. 또한 이 결정자는 새로운 릴레이션에 대한 외래키 역할도 함

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)



(a) 애틀리뷰트가 세 개



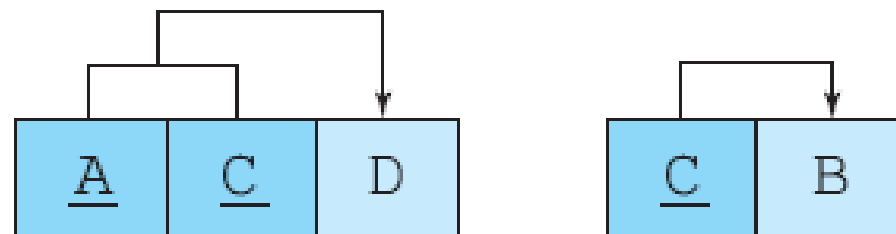
(b) 애틀리뷰트가 네 개

[그림 7.22] 제3정규형을 만족하지만 BCNF는 만족하지 않는 릴레이션

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)



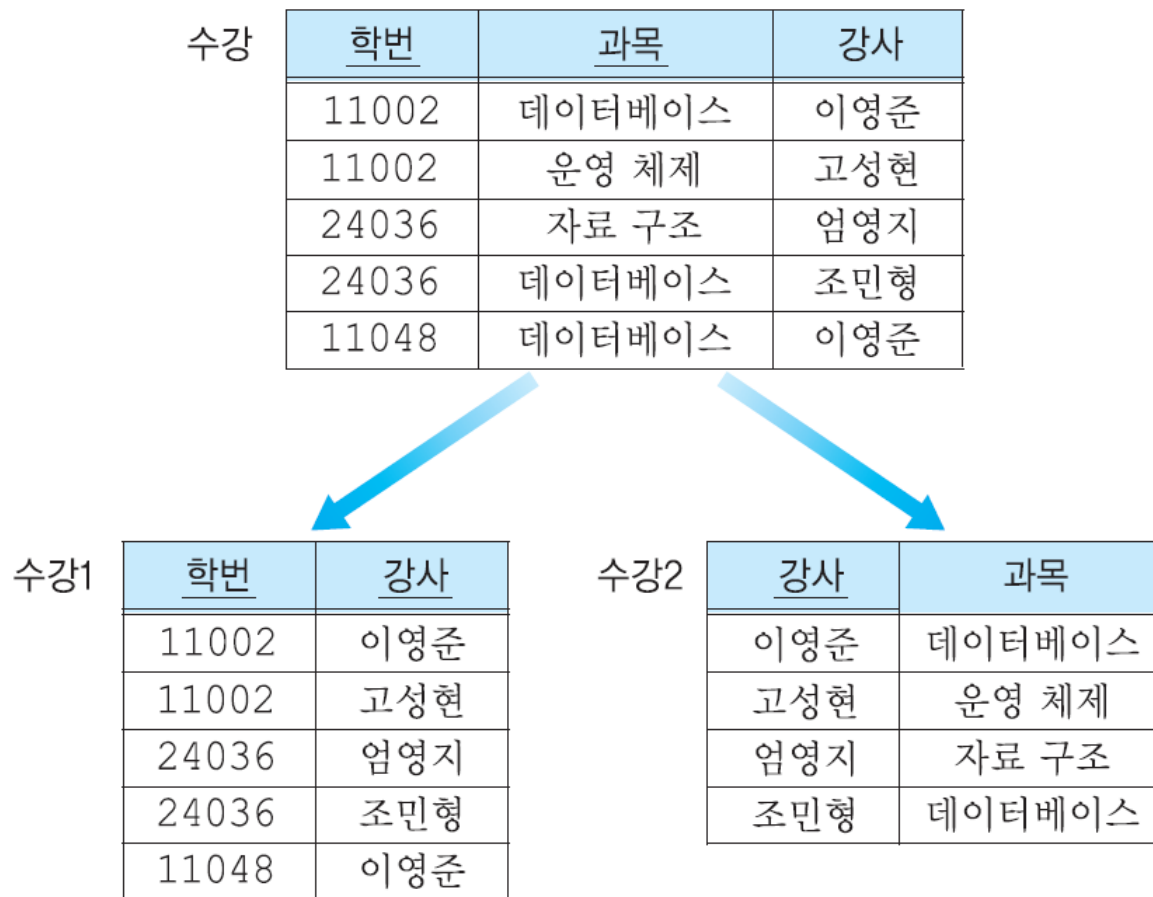
(a) 애트리뷰트가 세 개



(b) 애트리뷰트가 네 개

[그림 7.23] 제3정규형을 BCNF로 분해

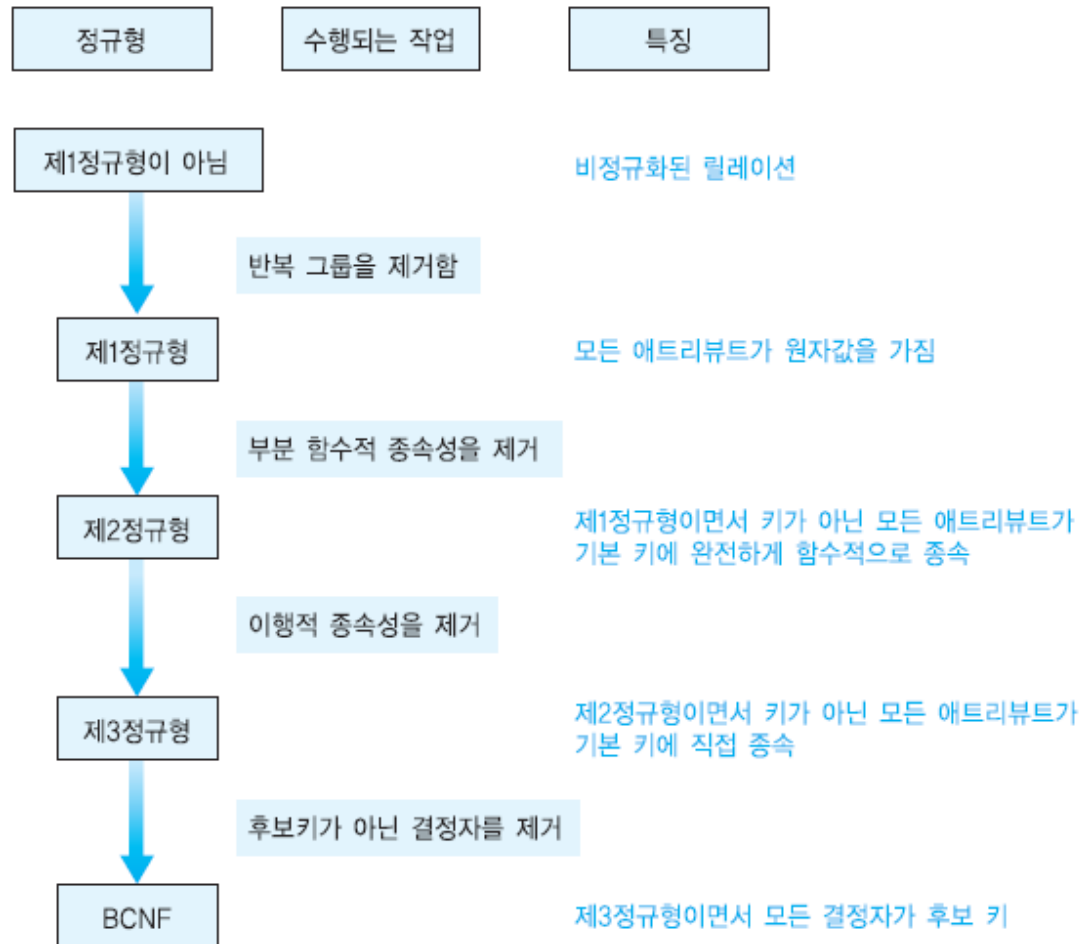
7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)



[그림 7.24] 제3정규형을 BCNF로 정규화

7.4 제1정규형, 제2정규형, 제3정규형, BCNF(계속)

□ 여러 정규형의 요약



[그림 7.25] 각 정규형의 특징과 정규화 과정

7.5 역정규화(denormalization)

□ 정규화 장점

- ✓ 정규화 단계가 진행될수록 중복이 감소하고 갱신 이상도 감소됨
- ✓ 정규화가 진전될수록 무결성 제약조건을 시행하기 위해 필요한 코드의 양도 감소됨

□ 정규화 단점

- ✓ 정규화가 데이터베이스 설계의 중요한 요소이지만 성능상의 관점에서만 보면 높은 정규형을 만족하는 릴레이션 스키마가 최적인 것은 아님
- ✓ 한 정규형에서 다음 정규형으로 진행될 때마다 하나의 릴레이션이 최소한 두 개의 릴레이션으로 분해됨
- ✓ 분해되기 전의 릴레이션을 대상으로 질의를 할 때는 조인이 필요 없지만 분해된 릴레이션을 대상으로 질의를 할 때는 같은 정보를 얻기 위해서 보다 많은 릴레이션들을 접근해야 하므로 조인의 필요성이 증가함

7.5 역정규화(계속)

예 : 조인의 필요성

제2정규형을 만족하는 그림 7.18의 학생1 릴레이션에서 “학번이 11002인 학생이 속한 학과의 이름과 전화번호를 검색하라”는 질의를 아래와 같은 SELECT문으로 표현한다. 한 릴레이션에서 필요한 정보를 모두 찾을 수 있으므로 조인이 필요 없다.

```
SELECT      학과이름, 학과전화번호
FROM        학생1
WHERE       학번 = '11002';
```

그러나 정규화 과정을 거쳐 그림 7.18의 릴레이션이 그림 7.19(b)처럼 두 개의 릴레이션으로 분해되면 동일한 정보를 찾기 위해 아래와 같이 조인을 포함한 SELECT문이 사용된다.

```
SELECT      학과이름, 학과전화번호
FROM        학생2, 학과
WHERE       학번 = '11002'
AND 학생2.학과이름 = 학과.학과이름;
```



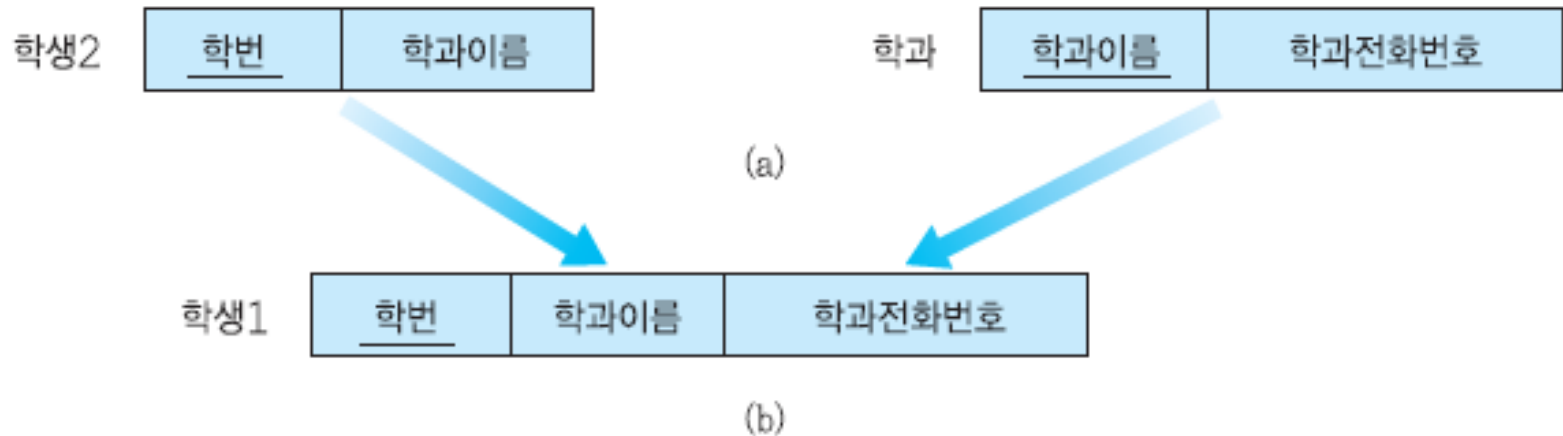
조인조건

7.5 역정규화(계속)

□ 역정규화

- ✓ 때로 데이터베이스 설계자는 응용의 요구 사항에 따라 데이터베이스 설계의 일부분을 역정규화함으로써 데이터 중복 및 갱신 이상을 대가로 치르면서 성능상의 요구를 만족시키기도 함
- ✓ 많은 데이터베이스 응용에서 검색 질의의 비율이 갱신 질의의 비율보다 훨씬 높음. 역정규화는 주어진 응용에서 빈번하게 수행되는 검색 질의들의 수행 속도를 높이기 위해서 이미 분해된 두 개 이상의 릴레이션들을 합쳐서 하나의 릴레이션으로 만드는 작업
- ✓ 즉 역정규화는 보다 낮은 정규형으로 되돌아가는 것

7.5 역정규화(계속)



[그림 7.26] (a) 제3정규형을 만족하도록 분해된 두 릴레이션
(b) 역정규화된 릴레이션