

CHAPTER

08

정규화

1. 정규화와 이상 현상
2. 함수 종속성
3. 기본 정규형

학습목표

- 정규화의 필요성과 이상 현상을 이해한다.
- 함수 종속성의 개념과 함수 종속 다이어그램을 알아 본다.
- 기본 정규형을 이해하고 정규화를 적용해 본다.

1. 정규화와 이상 현상

- 데이터베이스 설계(database design)

- 대량의 데이터를 데이터베이스 안에 어떻게 조직하여 구성하느냐는 중요한 문제
- 현실 세계를 정확하고 자연스럽게 반영할 수 있도록 데이터베이스의 논리적 구조를 결정
- 전체 데이터베이스에 대한 통합된 하나의 스키마(schema)를 정의

- 이상 현상

- 잘못된 스키마 정의는 원하지 않는 여러 이상 현상을 발생시킴
 - SELECT문으로 검색할 경우는 아무런 이상 현상이 발생하지 않음
 - INSERT, UPDATE, DELETE문으로 데이터를 변경할 때는 3가지 이상 현상이 발생

- 정규화

- 잘못된 스키마 정의를 바로 잡는 데이터베이스 설계 방법

1.2 이상 현상

1) 삽입 이상(insertion anomaly)

- 새 영화번호 'm006'이 상영관 '4관'에서 상영된다는 사실을 '예약' 릴레이션에 추가가정
- 불필요한 데이터를 함께 입력하지 않고서는 원하는 데이터만 입력이 불가능한 상황

2) 갱신 이상(update anomaly)

- 영화번호 'm002'인 영화의 상영관을 '3관'에서 '4관'으로 변경 가정
- 중복된 속성 값 중 일부가 수정되지 않을 경우 데이터 불일치가 발생할 수 있는 상황

3) 삭제 이상(deletion anomaly)

- 고객번호 'y001'인 고객이 영화번호 'm004'인 영화의 예약을 취소하는 경우 가정
- 삭제할 때 원하지 않는 유용한 데이터까지 함께 삭제되어 데이터 손실이 발생할 수 있는 상황

예약

고객번호	영화번호	티켓수	상영관
c001	m002	3	3관
c004	m005	2	2관
c003	m002	5	3관
c002	m001	1	1관
y001	m004	2	5관
c004	m003	1	2관
y001	m002	4	3관
c003	m001	1	1관
c004	m002	2	3관

1.3 정규화의 개념

- 이상현상 발생 원인

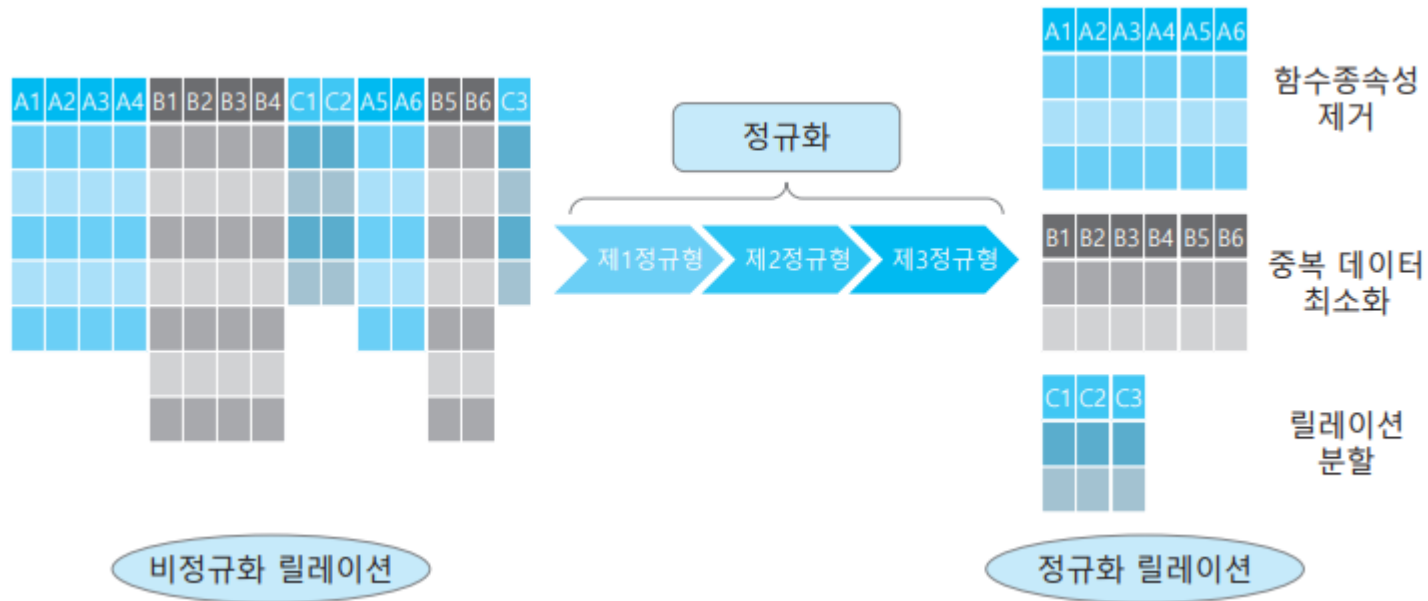
- 속성 사이의 많은 연관 관계를 무리하게 하나의 릴레이션으로 표현할 때 발생
 - 이를 방지하려면 속성 사이의 연관 관계 즉, 종속성(dependency)을 분석하여 하나의 릴레이션에는 하나의 종속성만 표현되도록 릴레이션을 분해하면 됨
- 기본적으로 연관성이 높은 속성들을 하나의 릴레이션으로 구성하는 것이 바람직

- 정규화(normalization)

- 데이터 이상 방지를 위해 중복을 감소시키는 구조로 단계적 규칙에 따라 릴레이션을 분해하는 과정
- 잘못된 설계를 바로 잡는 과정

정규화의 개념

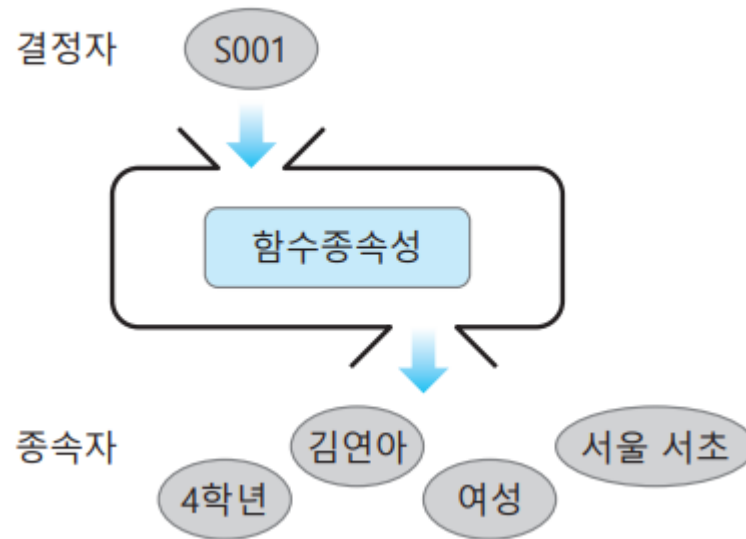
- 정규화 과정에서 이상 문제를 해결하려면 연관성이 높은 속성들로만 릴레이션을 구성해야 함
 - 릴레이션 속성 사이의 연관성을 평가하기 위한 척도가 필요 -> 함수 종속성



2. 함수 종속성

2.1 함수 종속성 정의

- 함수 종속성(FD: Functional Dependency)
 - 같은 릴레이션 안의 속성 간에 특정 속성 값이 함수적으로 다른 속성 값을 결정하는 종속 관계



함수 종속성

- 함수 종속성(FD: Functional Dependency)
 - 같은 릴레이션 안에 포함된 속성 사이의 연관성을 분석할 수 있는 척도
 - '속성1 → 속성2'로 표기
 - ✓ 어떤 속성2의 값이 다른 속성1의 값에 의해 결정되는 함수 종속 관계
 - ✓ 속성1은 속성2를 결정하는 결정자(determinant), 속성2는 속성1에 종속된 종속자(dependent)

학생_1

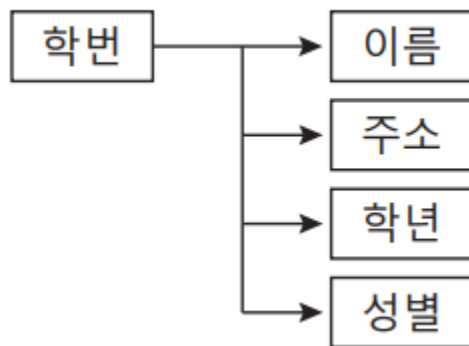
학번	이름	주소	학년	성별
s001	김연아	서울 서초	4	여
s002	홍길동	미정	1	남
s003	이승엽	미정	3	남
s004	이영애	경기 분당	2	여
s005	송윤아	경기 분당	4	여
s006	홍길동	서울 종로	2	남
s007	이은진	경기 과천	1	여

학번 → 이름
학번 → 주소
학번 → 학년
학번 → 성별

학번 → (이름, 주소, 학년, 성별)

2.2 함수 종속 다이어그램

- 함수 종속 다이어그램(FD diagram)
 - 하나의 릴레이션을 구성하는 속성들 간의 복잡한 함수 종속 관계를 이해하기 쉽도록 표현한 그림
 - 릴레이션 속성은 사각형으로, 속성 간의 함수 종속성은 화살표로 표기
- ‘학생_1’ 릴레이션의 모든 함수 종속성을 함수 종속 다이어그램으로 표현



완전 함수 종속과 부분 함수 종속성

- 완전 함수 종속(full functional dependency)

- 특정 속성이 결정자인 둘 이상의 전체 속성 조합에는 함수 종속이면서 결정자의 어떤 일부 속성에도 함수 종속이 아닐 때
 - 결정자인 기본키에 속한 모든 속성 값을 통해서만 기본키가 아닌 일반 속성을 결정할 수 있음
 - 보통 함수 종속은 완전 함수 종속을 의미
 - 결정자가 단일 속성이면 당연히 완전 함수 종속임

- 부분 함수 종속(partial functional dependency)

- 특정 속성이 결정자인 둘 이상의 전체 속성 조합에도 함수 종속이면서 결정자의 일부 속성에도 함수 종속일 때
 - 결정자인 기본키에 속한 일부 속성 값을 통해서도 기본키가 아닌 일반 속성을 결정할 수 있음

[참고] 릴레이션의 구성 요소

- 릴레이션
 - 릴레이션 스키마와 릴레이션 인스턴스 2가지 요소로 구성

- 릴레이션의 구성

학생

릴레이션

학번

이름

학년

성별

s001

김연아

4

여

s002

홍길동

1

남

s003

이승엽

3

남

릴레이션 스키마 (relation schema)

릴레이션 인스턴스 (relation instance)

No	구분	설명
1	릴레이션 스키마 (Relation Schema)	<ul style="list-style-type: none">릴레이션의 구조와 형식을 정의하는 일종의 틀 또는 청사진입니다.스키마는 릴레이션의 열(속성)에 대한 정보를 포함하며, 각 열의 이름과 데이터 유형 (예: 문자열, 숫자, 날짜)을 명시합니다.스키마는 데이터베이스 디자이너에 의해 정의되며, 릴레이션의 구조를 변경하려면 스키마를 수정해야 합니다.
2	릴레이션 인스턴스 (Relation Instance)	<ul style="list-style-type: none">실제 데이터로 채워진 릴레이션의 인스턴스 또는 구체적인 데이터 집합을 나타냅니다.각 행(튜플)은 릴레이션 스키마에 정의된 열에 대한 값으로 채워집니다.릴레이션 인스턴스는 데이터베이스에서 실제로 저장되는 정보를 나타내며, 사용자가 조회, 삽입, 갱신 및 삭제 작업을 수행할 때 조작되는 데이터입니다.

함수 종속 다이어그램의 예

- 함수 종속 다이어그램의 예(수강_1)

수강_1

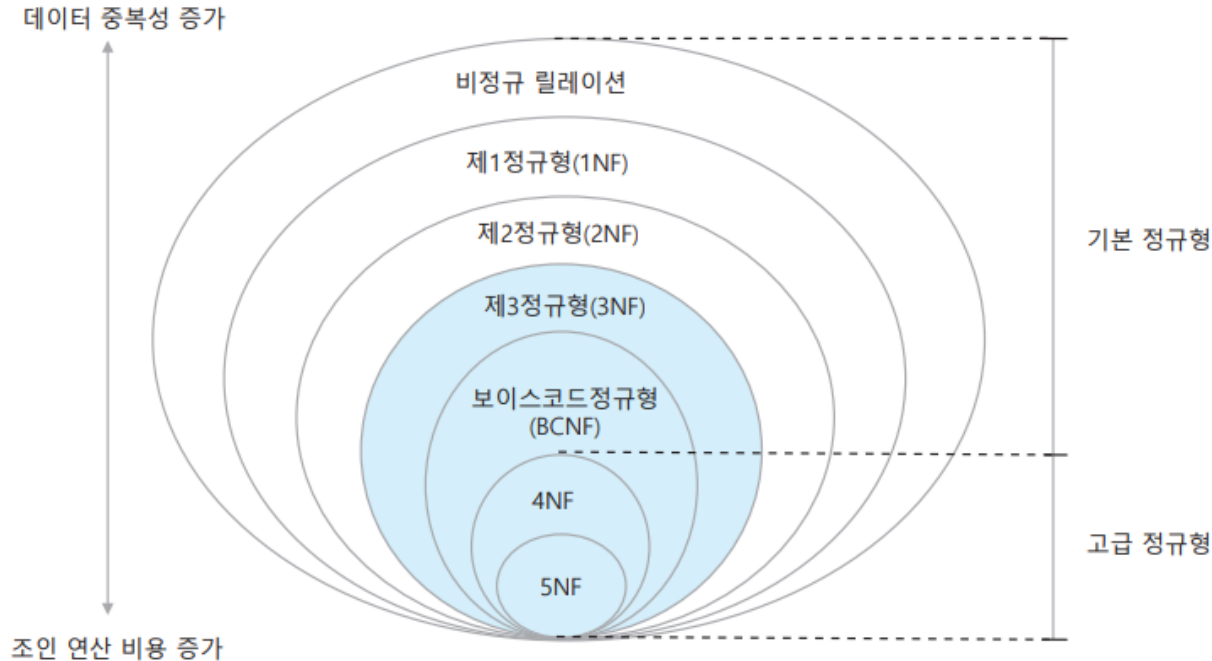
학번	과목번호	학점	성별	강의교수
s001	c002	A	여	강창욱
s004	c005	C	여	이기찬
s003	c002	B	남	강창욱
s002	c001	F	남	홍리라
s001	c004	B	여	김우영
s004	c003	A	여	박미라
s001	c005	C	여	이기찬
s003	c001	B	남	홍리라
s004	c002	A	여	강창욱

{학번, 과목번호} → (학점, 성별, 강의교수)
학번 → 성별
과목번호 → 강의교수



3. 기본 정규형

3.1 정규형의 종류



● 정규형(NF: Normal Form)

- 정규화 과정에서 릴레이션이 만족해야 하는 특정한 함수 종속성의 충족 조건
 - 특정 정규형이 요구하는 충족 조건을 릴레이션이 만족하면 해당 수준의 정규형을 충족함을 의미함
- 제3정규형 이상의 정규형을 충족하면 충분한 정규화가 이루어졌다고 봄
 - 보통 제3정규형 또는 보이스코드 정규형까지만 정규화를 진행

3.2 제1정규형

- 제1정규형 정의

제1정규형(1NF): 어떤 릴레이션 R에 속한 모든 속성의 도메인이 원자 값(atomic value)만을 갖는다면 제1정규형(First Normal Form)에 속한다.

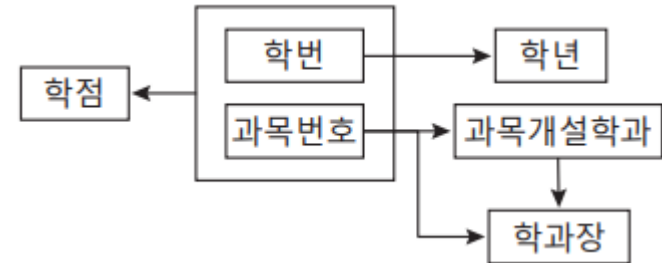
- 정규화 대상인 릴레이션이 관계형 데이터 모델의 기본 원칙을 따르기만 한다면 당연히 제1정규형에 속함

학생_2(학번, 학생이름, 주소, {전화번호}, 성별)	제1정규형 (x)
학생_3(학번, 학생이름, 주소, 집전화번호, 휴대폰번호, 성별)	제1정규형 (0)

제1정규형의 문제점

수강_2

학번	과목번호	학점	학년	과목개설학과	학과장
s001	c001	A	1	컴퓨터	박유찬
s002	c001	B	3	컴퓨터	박유찬
s002	c002	C	3	경영	김철수
s003	c004	A	4	컴퓨터	박유찬
s004	c003	B	2	정보통신	강민애
s004	c003	C	2	정보통신	강민애
s002	c005	F	3	컴퓨터	박유찬



- (1-1) 삽입 이상
 - 과목번호 ‘c006’인 과목의 개설학과가 ‘통계’학과라는 사실만 따로 삽입할 수가 없음
- (1-2) 수정 이상
 - 컴퓨터학과의 학과장이 ‘박유찬’에서 ‘홍길동’으로 변경될 경우, 과목개설학과 ‘컴퓨터’인 모든 투플을 찾아 ‘학과장’ 속성 값을 한꺼번에 ‘홍길동’으로 변경해야 함
- (1-3) 삭제 이상
 - 만약, 학번 ‘s002’ 학생이 과목번호 ‘c002’ 과목 수강을 취소하여 이 투플을 삭제하면 ‘c002’ 과목의 개설학과가 ‘경영’학과이고 학과장이 ‘김철수’라는 원하지 않은 정보까지 데이터베이스에서 함께 삭제됨

제1정규형의 문제점 해결

수강_3

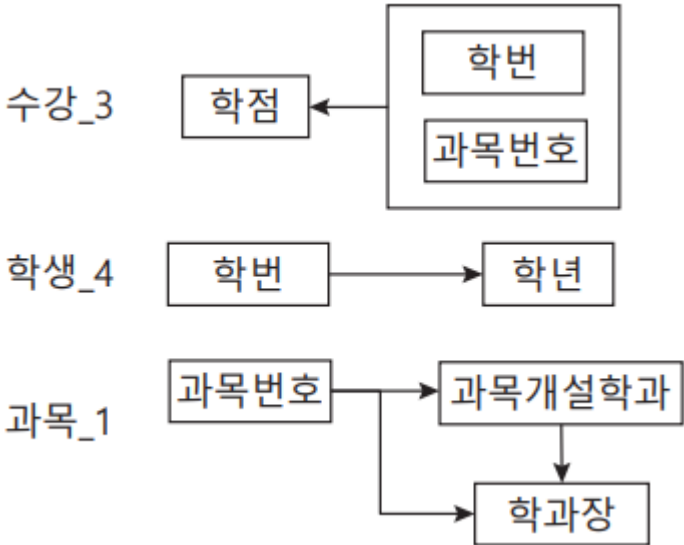
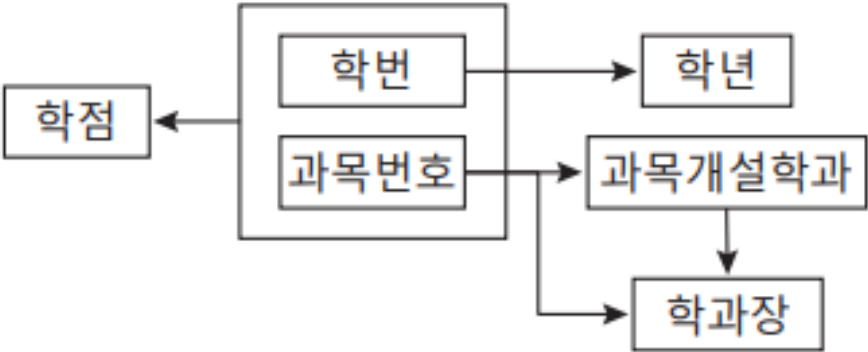
학번	과목번호	학점
s001	c001	A
s002	c001	B
s002	c002	C
s003	c004	A
s004	c003	B
s004	c003	C
s002	c005	F

학생_4

학번	학년
s001	1
s002	3
s003	4
s004	2

과목_1

과목번호	과목개설학과	학과장
c001	컴퓨터	박유찬
c002	경영	김철수
c003	정보통신	강민애
c004	컴퓨터	박유찬
c005	컴퓨터	박유찬



무손실 분해

- 무손실 분해(nonloss decomposition)

- 정규화 과정에서 릴레이션 분해는 어떤 정보도 손실되지 않게 동등한 릴레이션들로 분해해야 함
- 정보 손실 없이 프로젝트(project) 연산으로 무손실 분해된 릴레이션은 자연 조인(natural join) 연산에 의해 다시 분해 이전의 릴레이션으로 복원이 가능
- 복원이 불가능하면 잘못된 분해이며 이는 올바른 정규화가 아님

- 결정자와 결정자에 종속되는 속성들을 함께 떼어내어 새로운 릴레이션을 생성할 때 결정자를 분할 전의 릴레이션에도 공통 속성으로 남겨두어야 함

- 결정자가 이후 외래키로써 분해된 릴레이션 사이의 연관성을 유지할 수 있음

- * 프로젝트 연산

데이터베이스에서 "프로젝트 연산(Project Operation)"은 관계형 데이터베이스 관리 시스템(RDBMS)에서 사용되는 SQL 쿼리나 데이터 처리 작업 중 하나로, 특정 릴레이션(테이블)에서 **원하는 속성(열)만을 선택하여 결과를 생성**하는 작업을 의미합니다.

프로젝트 연산은 데이터의 선택(select)과 추출(extract) 작업을 수행합니다.

```
SELECT student_name, student_id    -- 원하는 속성(열)
FROM Students;
```


3.3 제2정규형

● 제2정규형 정의

제2정규형(2NF): 어떤 릴레이션 R이 제1정규형이고 기본키에 속하지 않는 모든 속성이 기본키에 완전 함수 종속이면 제2정규형(Second Normal Form)에 속한다.

- 제2정규형은 제1정규형을 충족하는 릴레이션의 기본키가 복합 속성일 때, 기본키의 일부 속성이 결정자인지를 검사함
- 만약 기본키가 단일 속성이면, 모든 속성이 완전 함수 종속이므로 검사할 필요가 없이 제2정규형에 속함

학생_3(학번, 학생이름, 주소, 집전화번호, 휴대폰번호, 성별)

제2정규형 (0)

수강_2(학번, 과목번호, 학점, 학년, 과목개설학과, 학과장)

제2정규형 (X)

수강_3(학번, 과목번호, 학점)

제2정규형 (0)

학생_4(학번, 학년)

제2정규형 (0)

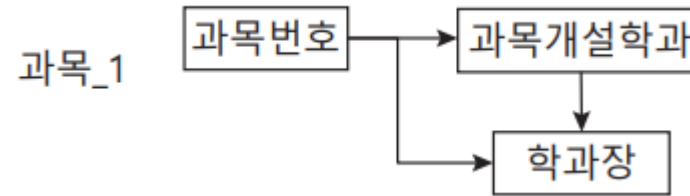
과목_1(과목번호, 과목개설학과, 학과장)

제2정규형 (0)

제2정규형의 문제점

과목_1

과목번호	과목개설학과	학과장
c001	컴퓨터	박유찬
c002	경영	김철수
c003	정보통신	강민애
c004	컴퓨터	박유찬
c005	컴퓨터	박유찬



- (2-1) 삽입 이상
 - 과목개설학과인 ‘통계’학과의 학과장이 ‘홍장미’라는 사실만 따로 삽입할 수가 없음
- (2-2) 수정 이상
 - 컴퓨터학과의 학과장이 ‘박유찬’에서 ‘홍길동’으로 변경될 경우, 여전히 과목개설학과 ‘컴퓨터’인 모든 투플을 찾아 ‘학과장’ 속성 값을 한꺼번에 ‘홍길동’으로 변경해야 함
- (2-3) 삭제 이상
 - 과목번호 ‘c002’의 등록을 취소하여 이 투플을 삭제하면 ‘경영’학과의 학과장이 ‘김철수’라는 원하지 않은 정보까지 함께 삭제됨

제2정규형의 문제점 해결

- 제2정규형을 충족하는 ‘과목_1’ 릴레이션에 삽입(2-1), 수정(2-2), 삭제(2-3) 이상이 발생하는 이유
 - 아직도 둘 이상의 의미적 연관성을 하나의 릴레이션으로 함께 표현했기 때문
- 이행적 함수 종속성(transitive functional dependency)
 - 기본키에 속하지 않은 일반 속성 값이 기본키에 속하지 않은 또 다른 일반 속성 값을 결정함
 - 해결방법: 이행적 종속 관계를 끊어 두 종속 관계를 각기 다른 릴레이션에 표현 즉, 2개의 릴레이션으로 분해함

과목_2

과목번호	과목개설학과
c001	컴퓨터
c002	경영
c003	정보통신
c004	컴퓨터
c005	컴퓨터

과목_3

과목개설학과	학과장
컴퓨터	박유찬
경영	김철수
정보통신	강민애

※ 이행 함수 종속

관계형 데이터베이스에서 정규화와 관련된 개념 중 하나로, 한 속성이 다른 속성을 통해 함수 종속되는 경우를 나타냅니다. 즉, A가 B에 함수 종속이고, B가 C에 함수 종속되면 A가 C에도 함수 종속되는 경우를 의미합니다.

학생 정보 릴레이션:

- 학번 (학생의 고유한 학번)
- 학과 (학생의 학과)
- 학과장 (학과의 학과장)

만약 우리가 학번에만 종속된다고 가정한다면, 이것은 학과와 학과장에 대한 정보 중복을 포함하고 있을 수 있습니다. 즉, 학생의 학과를 알면 학과장을 알 수 있습니다. 이것은 이행 함수 종속입니다.

3.4 제3정규형

● 제3정규형 정의

제3정규형(3NF): 어떤 릴레이션 R이 제2정규형이고 기본키에 속하지 않는 모든 속성이 기본키에 이행적 함수 종속이 아니면, 제3정규형(Third Normal Form)에 속한다.

- 제3정규형은 제2정규형을 충족하는 릴레이션의 기본키가 아닌 일반 속성이 결정자인지를 검사
- 일반 속성이 기본키 속성이 아닌 일반 속성에 종속적일 때 제3정규형에 위배됨

수강_3(학번, 과목번호, 학점)

제3정규형 (0)

학생_4(학번, 학년)

제3정규형 (0)

과목_1(과목번호, 과목개설학과, 학과장)

제3정규형 (X)

과목_2(과목번호, 과목개설학과)

제3정규형 (0)

과목_3(과목개설학과, 학과장)

제3정규형 (0)

제3정규형의 문제점

수강_4

학번	과목번호	학점	강의담당교수
s001	c001	A	p001
s002	c001	B	p001
s002	c002	C	p002
s003	c001	A	p004
s004	c003	B	p003
s005	c003	C	p003
s004	c001	F	p001



- (3-1) 삽입 이상
 - 과목번호 'c004'의 강의담당교수가 'p005'이라는 사실만 따로 삽입할 수는 없음
- (3-2) 수정 이상
 - 강의담당교수 'p001'의 담당 과목번호가 'c005'로 변경될 경우, 'p001'과 관련된 모든 튜플을 찾아 '과목번호' 속성 값을 한꺼번에 'c005'로 변경해야 함
- (3-3) 삭제 이상
 - 학번 's002'인 학생이 교과번호 'c002'의 수강을 취소하여 이 튜플을 삭제하면 'p002'교수가 'c002' 과목을 강의한다는 정보까지 함께 삭제됨

제3정규형의 문제점 해결

- 제3정규형을 충족하는 '수강_4' 릴레이션에 삽입(3-1), 수정(3-2), 삭제(3-3) 이상이 발생
 - 원인: 기본키가 아닌 속성이 기본키 일부 속성의 결정자가 되기 때문
 - 릴레이션의 후보키가 둘 이상이고 기본키가 둘 이상의 속성 조합일 때 발생
- 이러한 문제를 해결하려면 기본키가 아닌 결정자를 분리하여 2개의 릴레이션으로 분해

수강_5

학번	강의담당교수	학점
s001	p001	A
s002	p001	B
s002	p002	C
s003	p004	A
s004	p003	B
s005	p003	C
s004	p001	F

과목_4

강의담당교수	과목번호
p001	c001
p002	c002
p004	c001
p003	c003
p001	c001

부연설명 - 기본 정규형

관계형 데이터베이스에서는 기본 정규형(1NF, 2NF, 3NF)이 데이터의 무결성을 유지하고 중복을 최소화하기 위한 중요한 개념입니다. 아래에 각 기본 정규형에 대한 설명과 변환 방법을 예시와 함께 표로 제시하겠습니다.

정규형	설명	예시	변환 방법
1NF	각 속성 값이 원자적(Atomic)이어야 합니다. 즉, 각 셀에는 하나의 값을 가져야 합니다.	학생 정보 릴레이션: [학번, 과목]	중복 데이터를 분리하고, 원자적인 값을 갖도록 분해. (예: [학번, 수강과목], [학번, 성적])
2NF	1NF를 만족하면서, 기본키에 속하지 않는 모든 속성이 기본키에 완전 함수 종속이다.	주문 정보 릴레이션: [주문번호, 제품, 고객]	부분적 종속을 제거하고 관련된 데이터를 별도의 릴레이션으로 분리. (예: [주문번호, 제품], [고객, 주문번호])
3NF	2NF를 만족하면서, 모든 비주요 속성이 후보 키에 대해 이행 함수 종속을 가지면 안 됩니다.	직원 정보 릴레이션: [사원번호, 부서, 부서위치]	이행 종속을 제거하고 관련된 데이터를 별도의 릴레이션으로 분리. (예: [사원번호, 부서], [부서, 부서위치])

이러한 기본 정규형을 유지하면 데이터 중복을 최소화하고 데이터의 무결성을 유지할 수 있습니다. 정규화 과정은 데이터베이스 설계의 핵심 부분이며, 데이터 모델의 구조를 최적화하고 쿼리 성능을 향상시키는 데 도움이 됩니다.

3.5 보이스코드 정규형

- 보이스코드 정규형(BCNF: Boyce Codd Normal Form)의 정의
 - 복잡한 식별자 관계에 의한 문제를 해결하기 위해 제3정규형을 보완
 - “강한 제3정규형(strong 3NF)”이라고도 함

보이스코드 정규형(BCNF): 릴레이션 R의 모든 결정자(determinant)가 후보키(candidate key)이면 릴레이션 R은 보이스코드 정규형에 속한다.

- 제3정규형이더라도 기본키 속성이 기본키 속성이 아닌 일반 속성에 종속적일 때 보이스코드 정규형에 위배됨
 - 모든 결정자를 후보키로 만듦 즉, 기본키가 아니면서 결정자 역할을 하는 속성과 그 결정자에 함수 종속되는 속성을 하나의 릴레이션으로 분리
 - 이때, 결정자는 원 릴레이션에도 남겨서 외래키 역할을 하도록 함

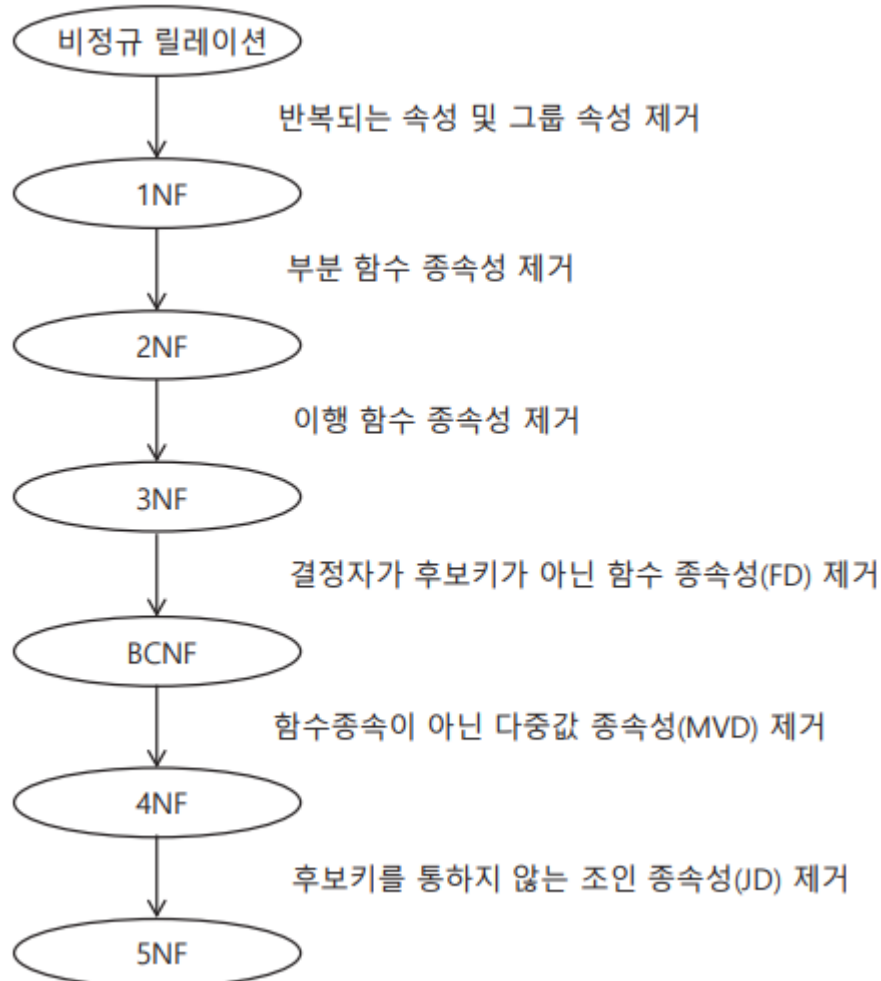
수강_3(학번, 과목번호, 학점)
학생_4(학번, 학년)
과목_1(과목번호, 과목개설학과, 학과장)
과목_2(과목번호, 과목개설학과)
과목_3(과목개설학과, 학과장)
수강_4(학번, 과목번호, 학점, 강의담당교수)
수강_5(학번, 강의담당교수, 학점)
과목_4(강의담당교수, 과목번호)

제3정규형 (0)	BCNF (0)
제3정규형 (0)	BCNF (0)
제3정규형 (X)	BCNF (X)
제3정규형 (0)	BCNF (0)
제3정규형 (0)	BCNF (0)
제3정규형 (0)	BCNF (X)
제3정규형 (0)	BCNF (0)
제3정규형 (0)	BCNF (0)

3.6 정규화의 적용

정규화

- 릴레이션을 정보 표현 측면에서 동등하면서도 중복을 감소시키는 더욱 작은 릴레이션들로 무손실 분해함으로써 이상 현상을 제거하는 데이터베이스 설계의 한 방법



반정규화

- 반정규화(de-normalization)

- 정규화의 반대 개념, ‘역정규화’
 - 정규화와는 반대로 보다 낮은 수준의 정규형으로 릴레이션을 통합
 - 성능 저하가 문제될 경우 분해된 릴레이션을 역으로 통합, 성능을 향상시키는 설계 방법
- 높은 정규형을 만족하는 릴레이션 스키마가 반드시 최적이라고 할 수는 없음
 - 정규화는 데이터의 정합성과 무결성이 강화되는 장점이 있지만 반대로 릴레이션 수가 과다하게 늘어나 SQL 문장이 복잡해지고 조인 연산 등으로 실행 시간이 오래 걸릴 수도 있기 때문
 - 자주 또 다량으로 조회되는 데이터에 대해서는 역으로 데이터 중복을 허용하는 반정규화 수행가능
- 반정규화는 다음 과정을 포함
 - 릴레이션들을 병합
 - 통계·이력 릴레이션을 추가
 - 여러 릴레이션에 같은 속성을 중복하여 추가
 - 총계·평균 같은 파생 속성을 추가
- 반정규화는 데이터베이스 설계의 최종 단계에서 신중하게 고려