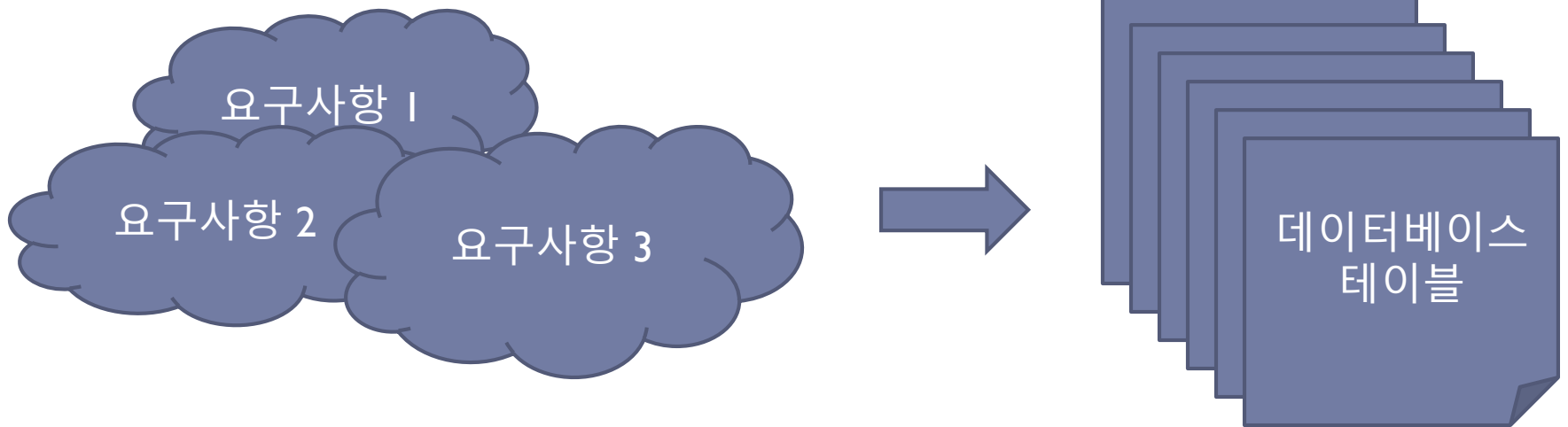


제 6 장 데이터베이스 설계

- 개념적 설계
- 개체관계 다이어그램
- 논리적 설계

데이터베이스 설계(Database Design)

- ▶ 사용자의 요구사항(requirements)으로부터 현실세계를 반영한 데이터베이스 구조 도출해내는 과정
 - ▶ 어떠한 필드로 구성된 테이블을 어떠한 물리적 형태의 데이터베이스로 구성할 것인가를 결정



데이터베이스 설계(Database Design)

▶ 요구사항

대학은 학생과, 교수의 인적 구성요소를 갖는다. 학생 정보로는 학번, 이름, 주소가 있으며, 교수는 사번, 이름, 임용연도가 있다. 대학은 다수의 학과가 있으며, 각 학과는 학과번호, 학과명을 갖고 학과사무실이 있다. 각 학생과 교수는 각각 하나의 학과에 소속된다.

또한 대학에는 교과목 리스트 있으며, 각 교과목에는 교과목번호, 교과목명, 학점수 정보가 있다. 대학은 매학기 교과목 일부를 강좌로 개설하며 하나의 교과목은 여러 개의 분반으로 개설될 수 있다. 또한 개설된 분반에는 강의실, 정원과 담당교수가 배정된다. 학생들은 매학기 강좌들을 수강하며 학점이 부여된다.

▶ 관계형 데이터 모델링 결과

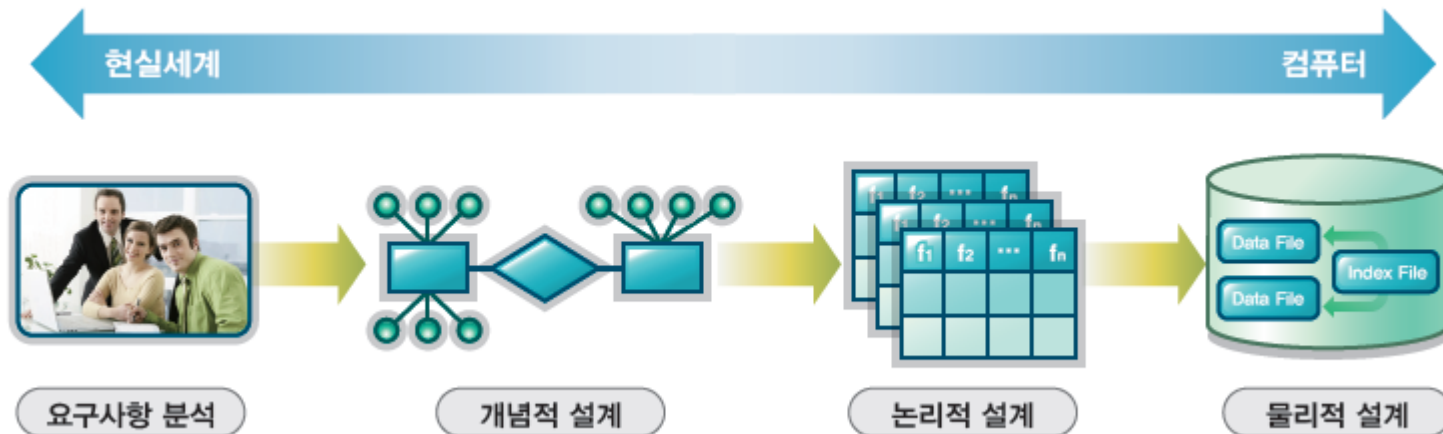
```
student (stu_id, name, address, dept_id)
professor (prof_id, name, dept_id, year_emp)
department (dept_id, dept_name, office)
curriculum (curriculum_id, title, credit)
class (class_id, curriculum_id, year, semester, division, prof_id, classroom, enroll)
takes (stu_id, class_id, grade)
```

▶ DBMS에 구현

```
create table student ( stu_id varchar2(20), ....)
create table professor ( prof_id varchar2(20), ....)
```

데이터베이스 설계 단계

- ▶ 요구사항 분석
 - ▶ DB 사용환경 분석 후 대상 및 제한 조건 도출.
- ▶ 개념적 설계(conceptual schema)
 - ▶ 분석 결과를 추상화된 표현 방식으로 기술 -> 개념적 스키마 생성
- ▶ 논리적 설계(logical schema)
 - ▶ 논리적 데이터베이스 구조에 맞는 스키마 생성 -> 논리적 스키마 생성
- ▶ 물리적 설계
 - ▶ 실제 컴퓨터에 저장되는 방식 설계



개념적 설계

- ▶ 사용자의 요구사항 분석 후, 컴퓨터에 표현방식 보다는 추상적인 형태로 설계
 - ▶ 사용자기 이해하기 쉬운 형태로 표현
- ▶ 개념적 모델을 이용하여 개념적 스키마 생성
- ▶ 대표적 개념적 모델
 - ▶ 개체관계 모델(Entity Relationship Model)
- ▶ 개념적 스키마
 - ▶ 데이터베이스에 대한 추상적인 설계도
 - ▶ 개체 관계 다이어그램(Entity Relationship Diagram: ERD)
 - ▶ 개체관계 모델의 표현 수단
 - ▶ ER 스키마(ER schema)라고도 함

논리적 설계

- ▶ 논리적 설계
 - ▶ 논리적 모델을 이용하여 논리적 스키마 생성
 - ▶ 즉, ERD를 이용하여 데이터베이스 스키마를 설계
- ▶ 논리적 모델 : 관계형 데이터 모델
 - ▶ 테이블 : 여러 데이터 도메인의 순서쌍의 집합
 - ▶ 데이터베이스 : 테이블의 집합
- ▶ 논리적 스키마
 - ▶ 테이블 구조도
 - ▶ 개념적 설계 단계에서 생성된 ERD를 바탕으로 생성되는 테이블

물리적 설계

- ▶ 특정 DBMS가 제공하는 물리적 구조에 따라 테이블 저장 구조 설계
- ▶ 테이블 저장 구조란?
 - ▶ 필드의 데이터 타입
 - ▶ 인덱스 지정 여부
 - ▶ 물리적 스키마
- ▶ 단순한 물리적 설계과정
 - ▶ 논리적 설계 단계에서 생성된 테이블 구조도에 따라 SQL create table 구문으로 각각의 테이블 생성
- ▶ 추가 고려 사항
 - ▶ 인덱스, 뷰 등 설정

데이터베이스 설계 단계 요약



설계 과정의 고려사항

- ▶ 충실성 (faithfulness)
 - ▶ 필요로 하는 모든 데이터를 표현한다.
- ▶ 단순성 (simplicity)
 - ▶ 단순하고 이해하기 쉬운 구조로 표현한다.
- ▶ 중복의 최소화 (redundancy minimization)
 - ▶ 데이터의 중복을 최소화 한다.
 - ▶ 저장공간의 효율적 사용, 데이터 일관성 유지
- ▶ 제약조건의 표현
 - ▶ 데이터가 갖추어야 할 조건을 표현한다.

개체관계 모델: 개념적 설계

- ▶ 개체(Entity)
 - ▶ 현실 세계에서 물리적/추상적으로 존재하는 실체
 - ▶ 사람, 자동차, 집, 수업, 성적, 과목 등
- ▶ 개체집합(Entity Set)
 - ▶ 동일한 특성을 갖는 개체들의 모임
 - ▶ {'김광식', '김정현' ...} : 학생(student) 개체집합
 - ▶ {'컴퓨터공학과', '산업공학과', ...} : 학과(department) 개체집합
- ▶ 속성(Attribute)
 - ▶ 개체의 특성
 - ▶ 관계형 데이터 모델의 필드와 같은 개념
 - ▶ 동일한 특성을 갖는 개체 => 속성이 동일한 개체

개체 집합의 예

student (stu_id, resident_id, name, year, address)	department (dept_id, dept_name, office)
(1292001, 900424-1825409, 김광식, 3, 서울)	(920, 컴퓨터공학과, 201호)
(1292002, 900305-1730021, 김정현, 3, 서울)	(923, 산업공학과, 207호)
(1292003, 891021-2308302, 김현정, 4, 대전)	(925, 전자공학과, 308호)
...	...

student 개체집합

department 개체집합

필드와 속성의 차이점

- ▶ 필드 – 관계형 데이터 모델
 - ▶ 관계형 데이터모델에서 테이블의 컬럼
 - ▶ 원자 값만 허용됨
- ▶ 속성 – 개체관계 모델
 - ▶ 개체관계 모델에서 개체의 특성
 - ▶ 다중값 속성, 복합 속성 가능
 - ▶ 다중값 속성
 - ▶ 하나의 속성에 여러 개의 값이 들어감
 - ▶ 예) family_member 속성에는 가족이름 여러 명이 포함
 - ▶ 복합 속성
 - ▶ 하나의 속성이 여러 개의 속성으로 구성됨
 - ▶ 예) address 속성은 세부적으로 (district, city, street)로 구성됨

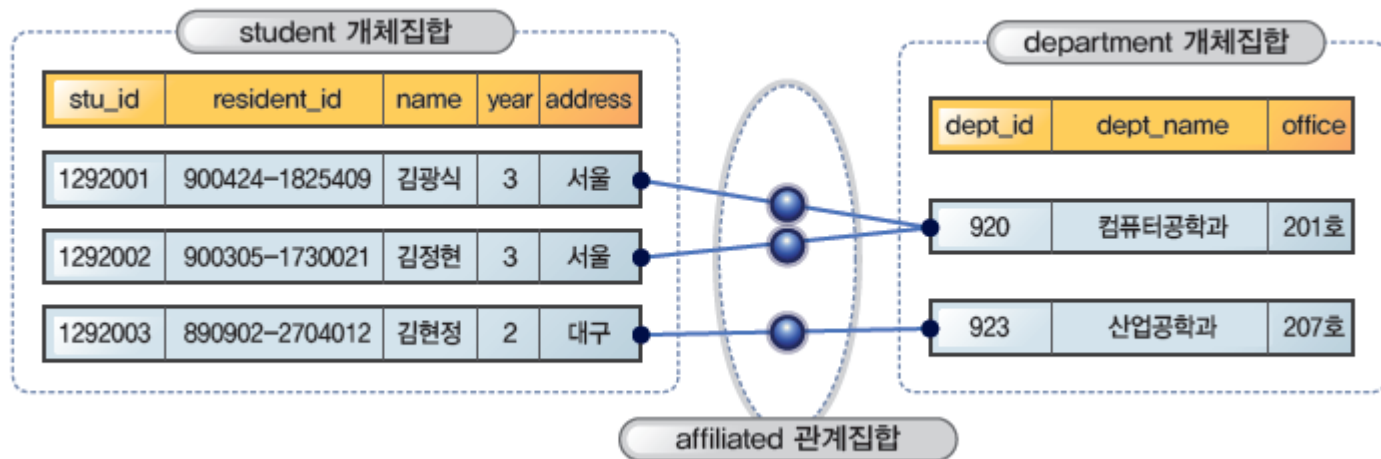
관계와 관계집합

▶ 관계

- ▶ 개체간의 대응성을 표현
- ▶ 개체간의 관계를 통해 의미를 규정할 수 있음
 - ▶ 예) 개체 '김광식'은 개체 '컴퓨터공학과'에 소속(affiliated)된다.
=> 순서쌍 ('김광식', '컴퓨터공학과')으로 표현

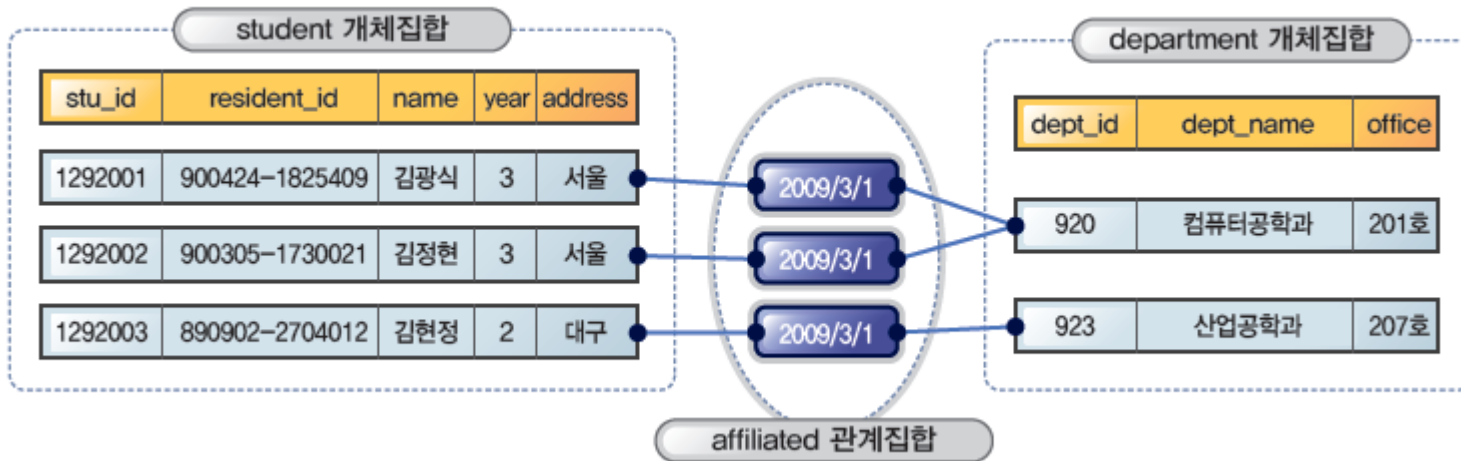
▶ 관계집합

- ▶ 동일한 유형의 관계들의 집합



관계 집합의 속성

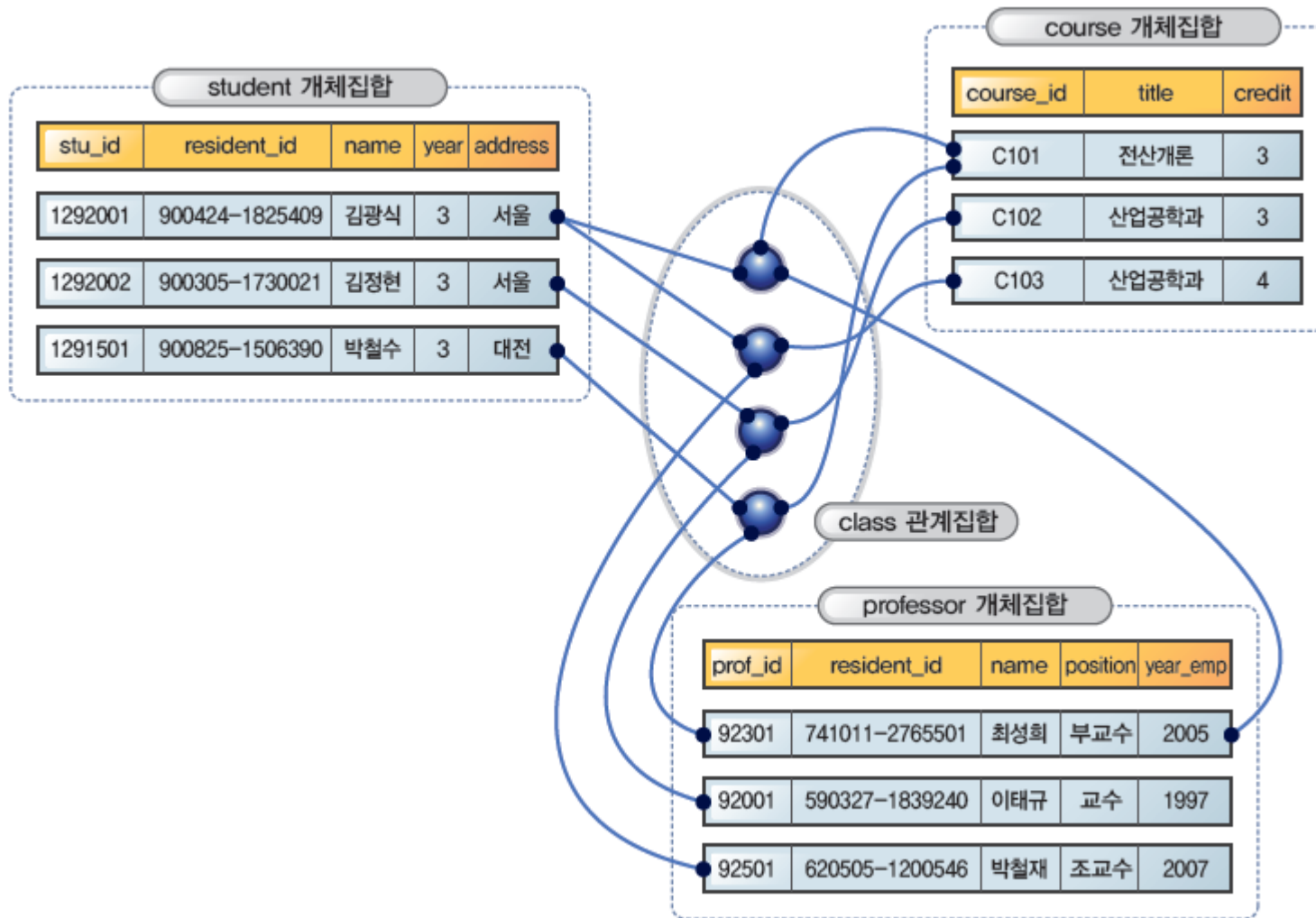
- ▶ 관계집합에도 속성의 정의가 가능
 - ▶ 개체들간의 관계의 특성을 표현
- ▶ 예 : 학생이 학과에 소속된 날짜



관계 집합의 차수(degree)

- ▶ 관계집합에 참여하는 개체집합의 개수
- ▶ 이진관계(binary relationship)
 - ▶ 두 개체집합 사이에 정의된 관계집합
 - ▶ 학생 – 학과의 소속관계
 - ▶ 학생 '김광식'은 학과 '컴퓨터공학과' 소속
- ▶ 삼진관계(ternary relationship)
 - ▶ 세 개체집합 사이에 정의된 관계집합
 - ▶ 학생-과목-교수의 강의/수강 관계
 - ▶ 학생 '김광식'은 교수 '최성희'가 강의하는 과목 '전산개론' 을 수강

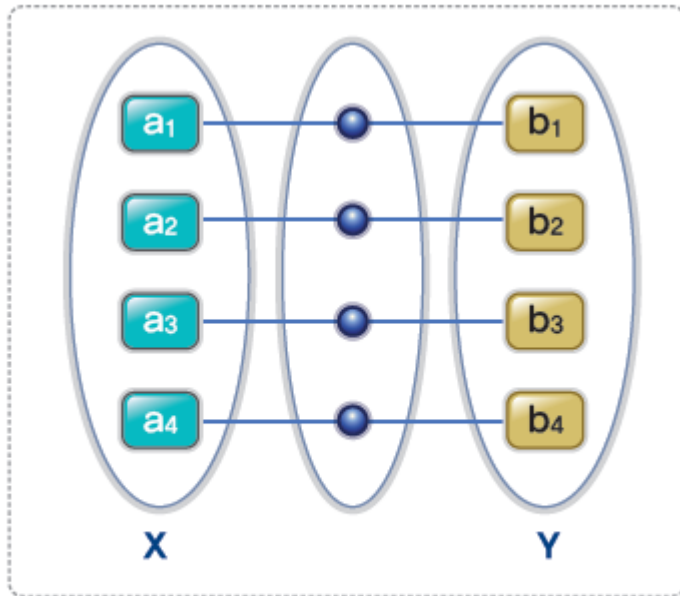
학생-과목-교수 삼진관계의 예



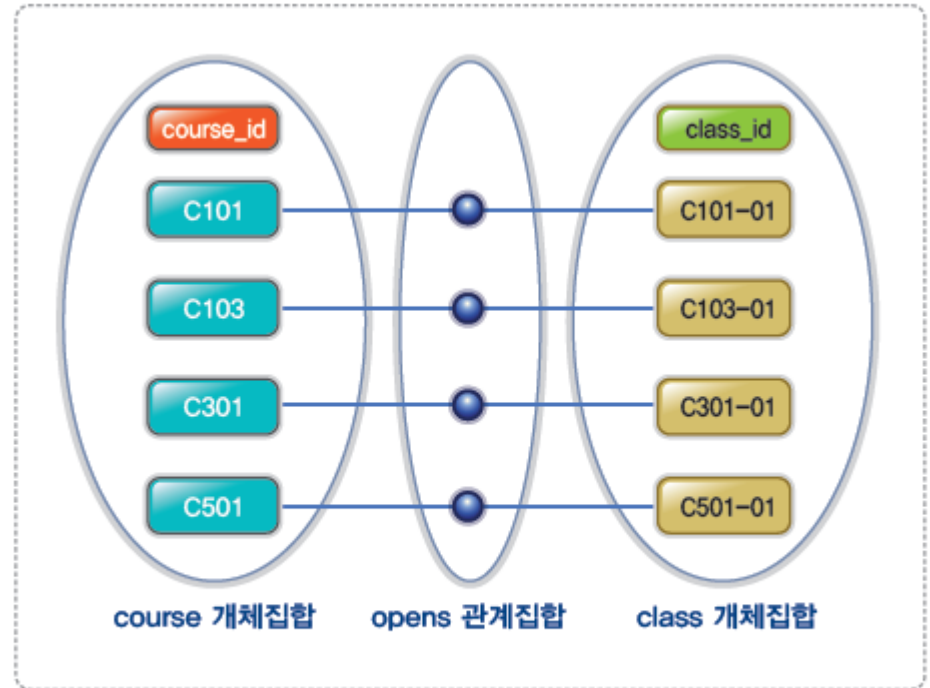
관계의 대응수(mapping cardinality)

- ▶ 관계집합에서 각 개체들이 참여할 수 있는 대응의 개수
- ▶ 학생 한 명은 하나의 학과에만 소속될 수 있다.
 - ▶ student 개체집합의 각 개체는 최대한 하나의 department 개체와 관계를 맺을 수 있음
 - ▶ ('김광식', '컴퓨터공학과'), ('김광식', '전자공학과')는 공존할 수 없다.
- ▶ 대응수의 종류
 - ▶ 일대일(one-to-one)
 - ▶ 일대다(one-to-many)
 - ▶ 다대일(many-to-one)
 - ▶ 다대다(many-to-many)

일대일(one-to-one)

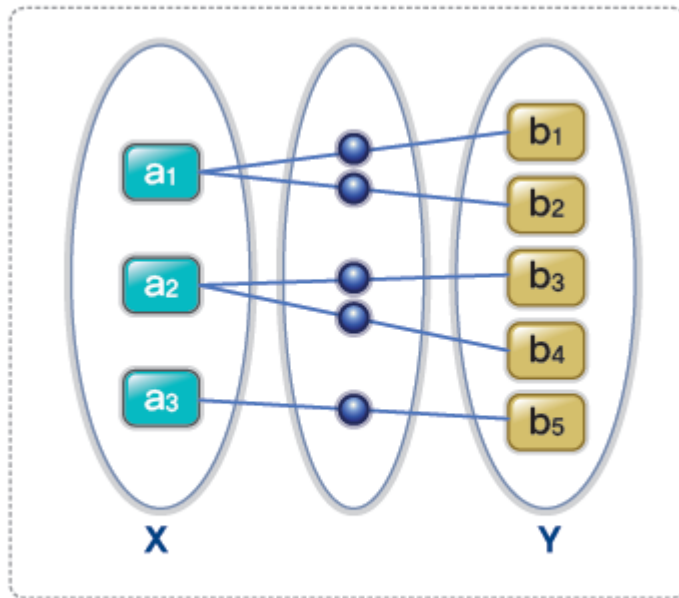


(a) X와 Y 간의 일대일 관계

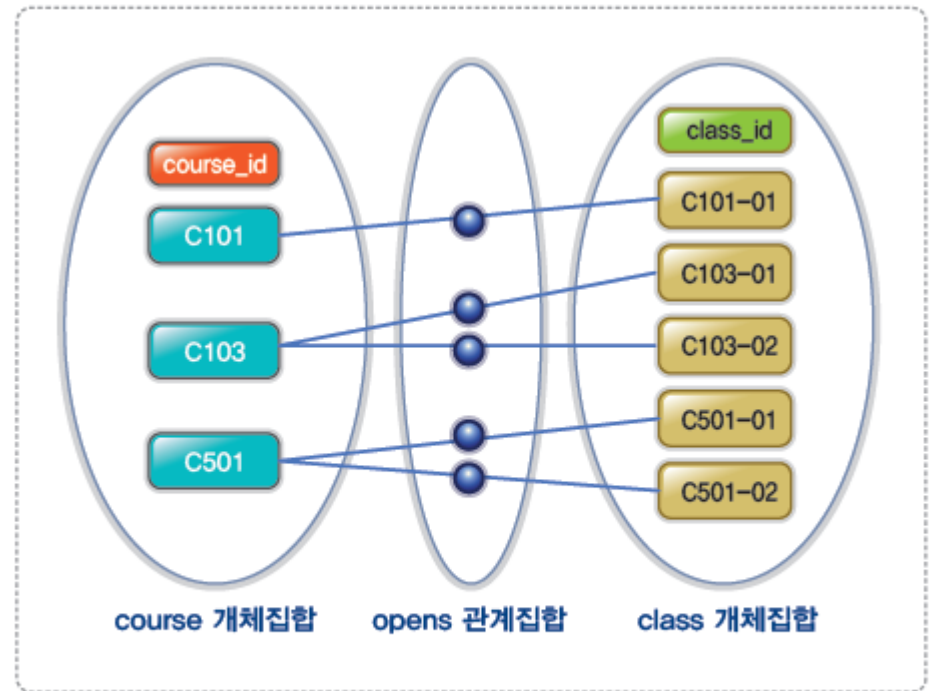


(b) course와 class 간의 일대일 관계

일대다(one-to-many), 다대일(many-to-one)

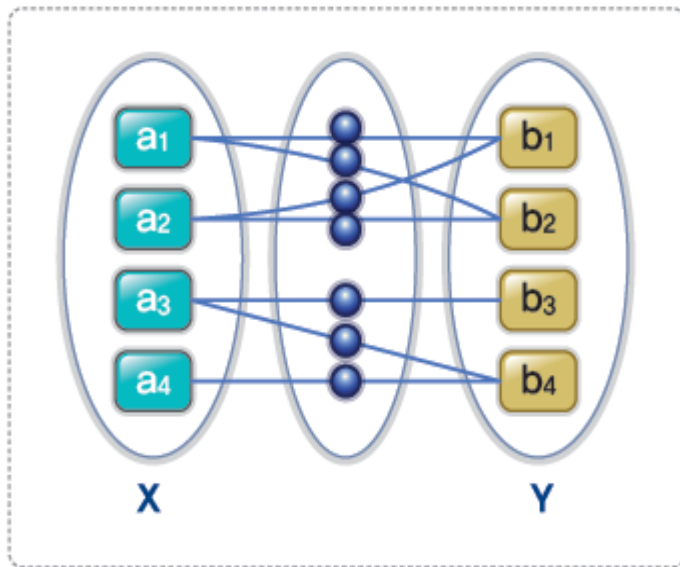


(a) X와 Y 간의 일 대 다 관계

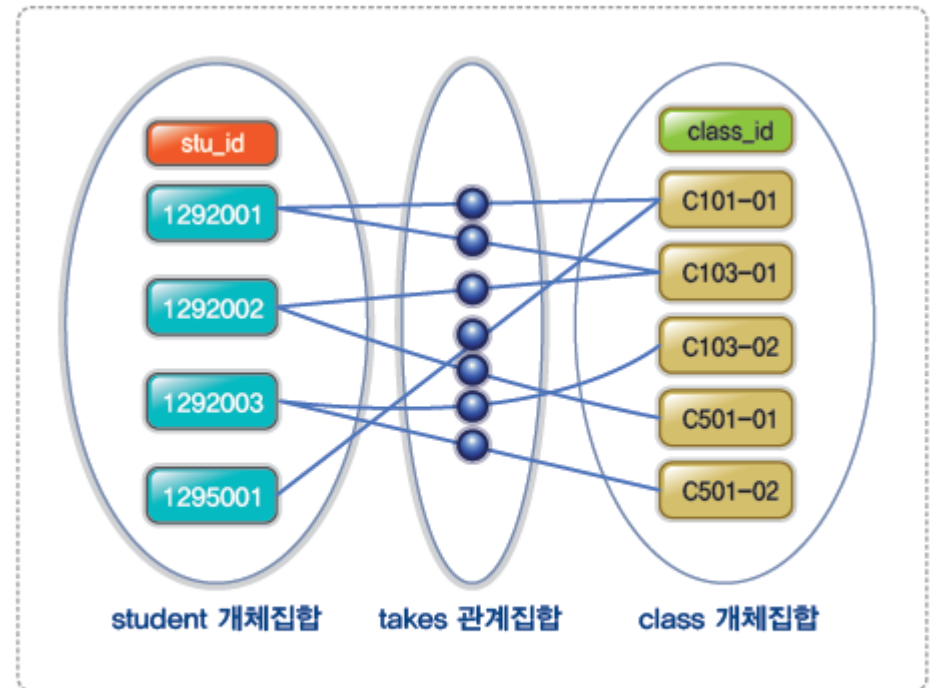


(b) course와 class 간의 일 대 다 관계

다대다(many-to-many)



(a) X와 Y 간의 다 대 다 관계

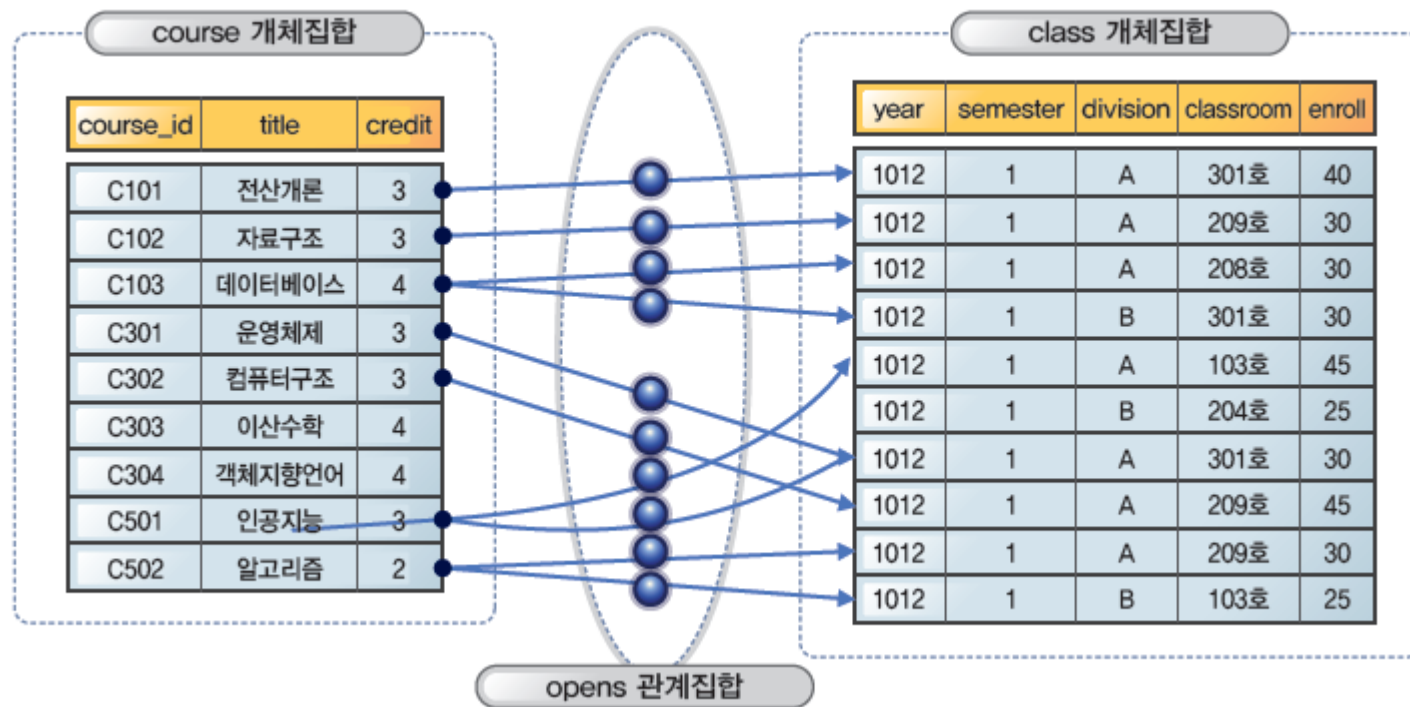


(b) student와 class 간의 다 대 다 관계

약성 개체집합 vs. 강성 개체집합

- ▶ 강성 개체집합(strong entity set)
 - ▶ 기본키 형성에 필요한 속성을 모두 갖는 개체집합
- ▶ 약성 개체집합(weak entity set)
 - ▶ 기본키 형성에 필요한 속성을 모두는 갖지 못한 개체집합
- ▶ 예)
 - ▶ course 개체집합
 - ▶ 속성: {course_id, title, credit}
 - ▶ 기본키는 course_id -> 강성 개체집합
 - ▶ class 개체집합
 - ▶ 속성: {year, semester, division, classroom, enroll}
 - ▶ {year, semester, division}은 동일 교과목(course) 내에서만 유일함, 따라서 약성 개체집합임
 - ▶ 약성 개체집합의 존재가 강성 개체집합의 존재에 의해 결정

약성 개체집합 vs. 강성 개체집합



약성 개체집합 vs. 강성 개체집합

- ▶ 약성 개체집합은 강성 개체집합에 항상 종속된다.
 - ▶ 약성 개체집합은 독립적으로 존재할 수 없으며 강성 개체집합이 존재해야 존재할 수 있다.
- ▶ 약성 개체집합과 강성 개체집합의 예
 - ▶ 직원 개체집합 vs. 부양가족 개체집합
 - ▶ 건물 개체집합 vs. 내부사무실 개체집합
 - ▶ 교과목(course) 개체집합 vs. 강좌(class) 개체집합

약성 개체집합 vs. 강성 개체집합

- ▶ 약성 개체집합과 강성 개체집합의 대응수 : 다대일
- ▶ 전체 참여(total participation)
 - ▶ 약성 개체집합의 모든 개체가 다대일 관계에 참여
- ▶ 부분 참여(partial participation)
 - ▶ 약성 개체집합의 일부 개체만 다대일 관계에 참여

부분 키(partial key), 구별자(discriminator)

- ▶ 약성 개체집합에서 강성 개체집합의 특정 개체 내에서만 유일한 값을 갖는 속성 집합
- ▶ course(교과목) 개체집합의 속성
 - ▶ course_id, title, credit
- ▶ class(강좌) 개체집합의 속성
 - ▶ year, semester, division, classroom, enroll
- ▶ 약성 개체집합인 class 개체집합의 부분키
 - ▶ 특정 교과목(course)에 대해서만 유일한 값을 갖는 강좌(class)의 속성
 - ▶ year, semester, division

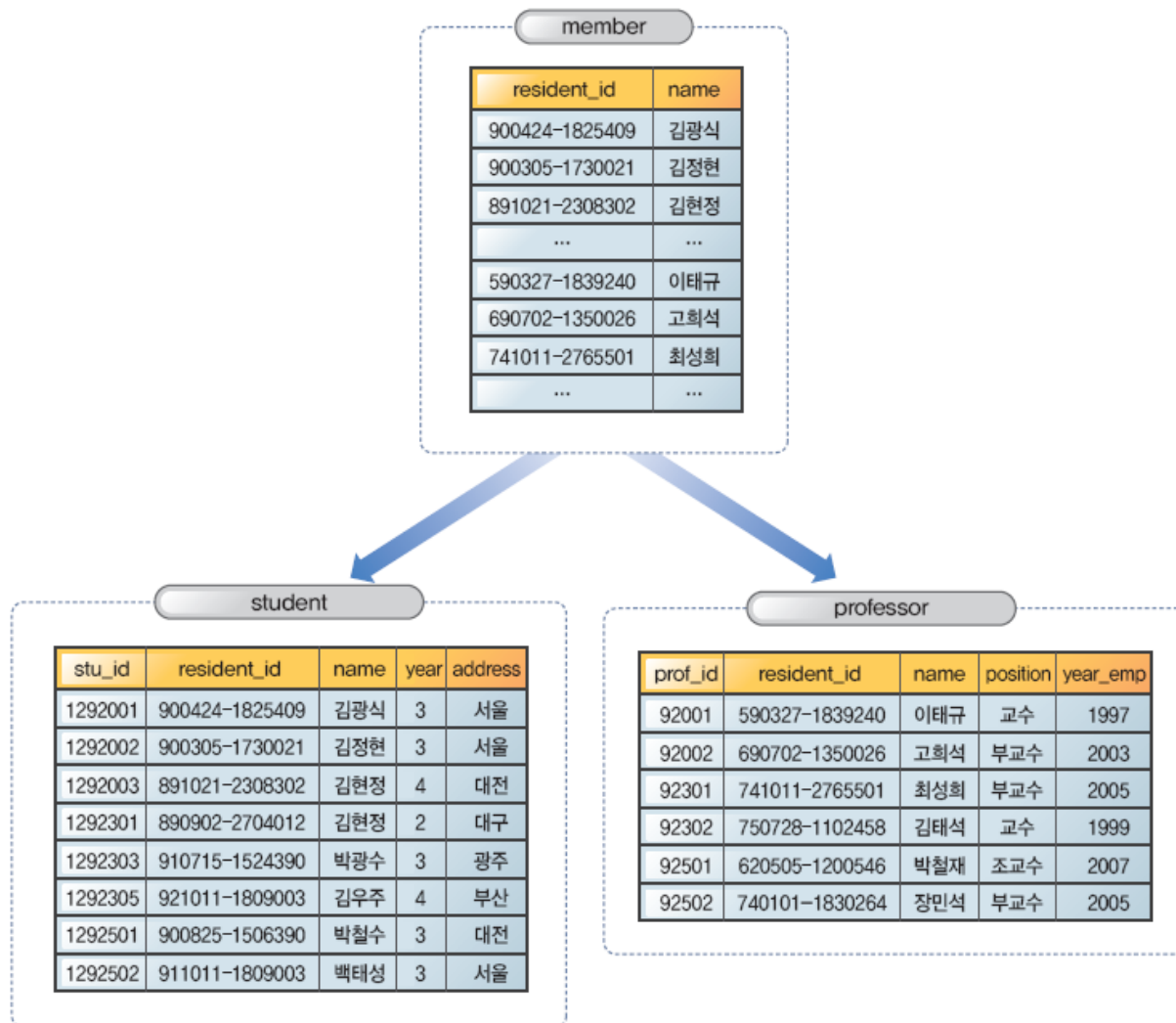
약성 개체집합의 기본키

- ▶ 약성 개체집합 자체만으로 기본키를 가질 수 없음
- ▶ 약성 개체집합의 기본키
 - ▶ 약성 개체집합의 부분키 + 강성 개체집합의 기본키
- ▶ 강좌(class) 개체집합의 기본키
 - ▶ **course_id**, year, semester, division

일반화 관계 vs. 세분화 관계









- ▶ 현실세계에 존재하는 개체들은 위상적으로 계층관계를 이루는 경우가 많음
 - ▶ 일반적 개념의 개체를 보다 구체화된 개념의 개체들로 분류 또는 분할해서 보여 줄 수 있음
 - ▶ 상위 개체집합 (high-level entity set) -> 하위 개체집합(low-level entity set)
- ▶ 일반화 관계(generalization)
 - ▶ 여러 개체집합의 공통적인 특징을 모아 상위 개체집합 생성
- ▶ 세분화 관계(specialization)
 - ▶ 하나의 개체집합을 여러 개의 하위 개체집합으로 분류






일반화 관계 vs. 세분화 관계



개체관계 다이어그램(ERD)

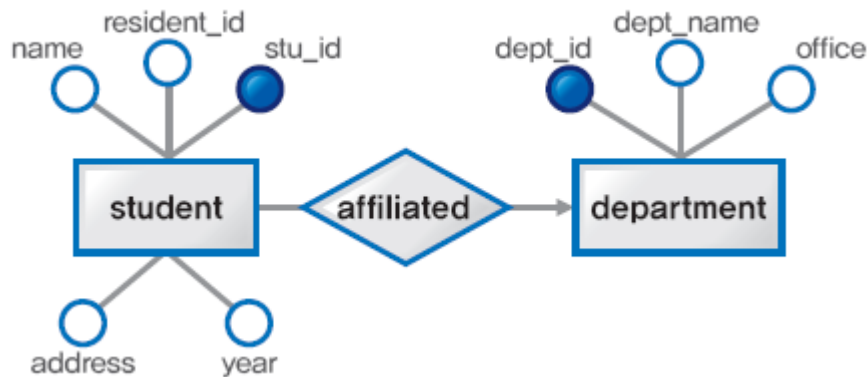
구성요소

기호	의미
	개체 집합
	약성 개체 집합
	관계 집합
	약성 관계 집합
	일반 속성
	기본키 속성
	부분키 속성
	다중 값 속성

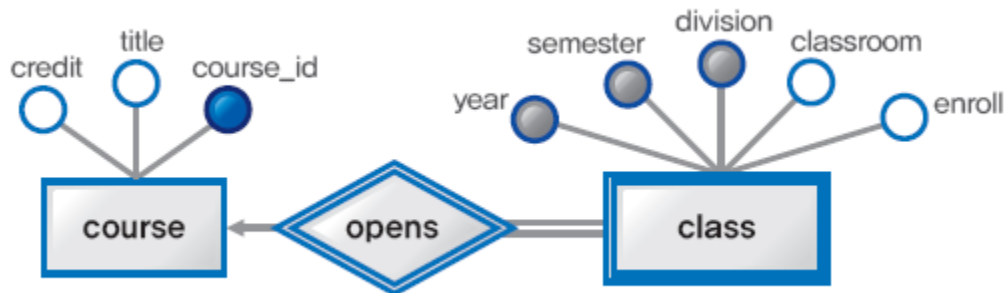
기호	의미
	일 대 일 관계
	일 대 다 관계
	다 대 다 관계
	일반화/세분화관계
	전체 참여 일반화/세분화 관계

ERD의 예

▶ student와 department의 ERD

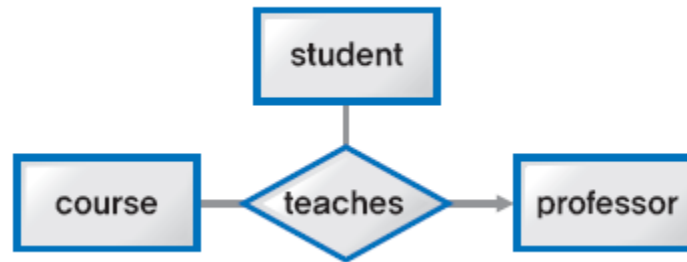


▶ course와 class의 ERD (약성 개체집합의 표현)

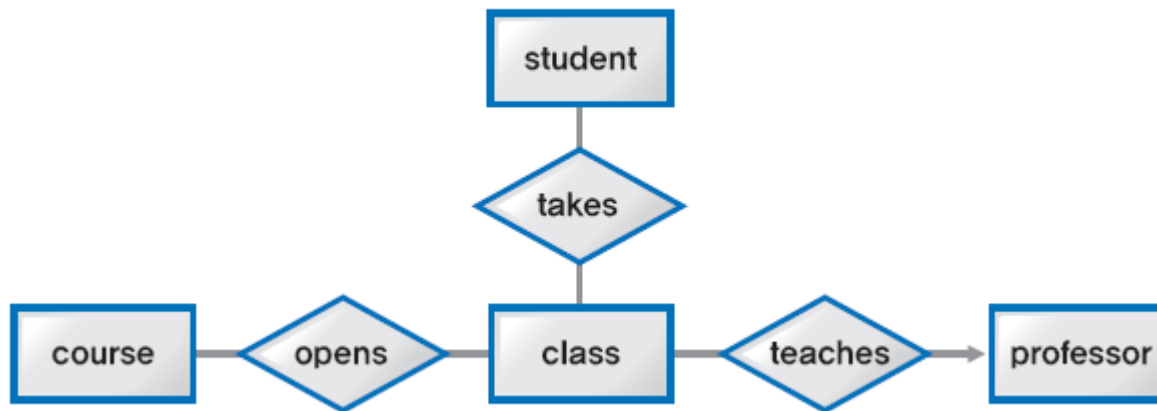


삼진관계 ERD

- ▶ student, course, professor 개체집합의 개체관계



- ▶ 이진관계로 표현된 ERD



자기연관관계(self-relationship) ERD

- ▶ 직원(employee) 개체집합
 - ▶ 각 직원은 자신을 관리하는 상관이 존재
 - ▶ 상관도 직원 개체집합에 속함
 - ▶ 각 상관은 여러 명의 직원을 관리



- ▶ 역할(role)
 - ▶ 각 개체의 역할에 따라 대응수가 다름

자기연관관계(self-relationship) ERD

- ▶ 교과목(course) 개체집합
 - ▶ 한 교과목은 여러 개의 선수(precede) 교과목을 가짐
 - ▶ 한 교과목은 여러 개의 후수(antecede) 교과목을 가짐
 - ▶ 다대다의 자기연관관계



개체 관계 스키마의 설계

▶ 학사 데이터베이스 구축을 위한 요구사항

대학의 구성원은 학생과 교수로 각 학생에게는 고유의 학번이 부여되며 이외에 주민등록번호, 이름, 주소, 학년의 정보를 갖는다. 교수에게도 고유의 교수번호가 부여되며 주민등록번호, 이름, 직급, 임용연도 등의 정보가 있다. 학생과 교수는 하나의 학과에 소속되며 학과는 학과번호, 학과명, 사무실 등이 있다. 각 교과목은 교과목번호, 교과목명, 학점수를 가지며 한 학기에 하나 이상의 강좌가 개설될 수 있다. 개설된 강좌는 강의실과 한 명의 교수가 배정된다. 한 강좌에 수강 정원이 초과하는 경우에는 여러 개의 분반을 개설할 수 있다. 학생은 한 학기에 하나 이상의 개설된 강좌를 수강할 수 있고 성적이 부여된다.

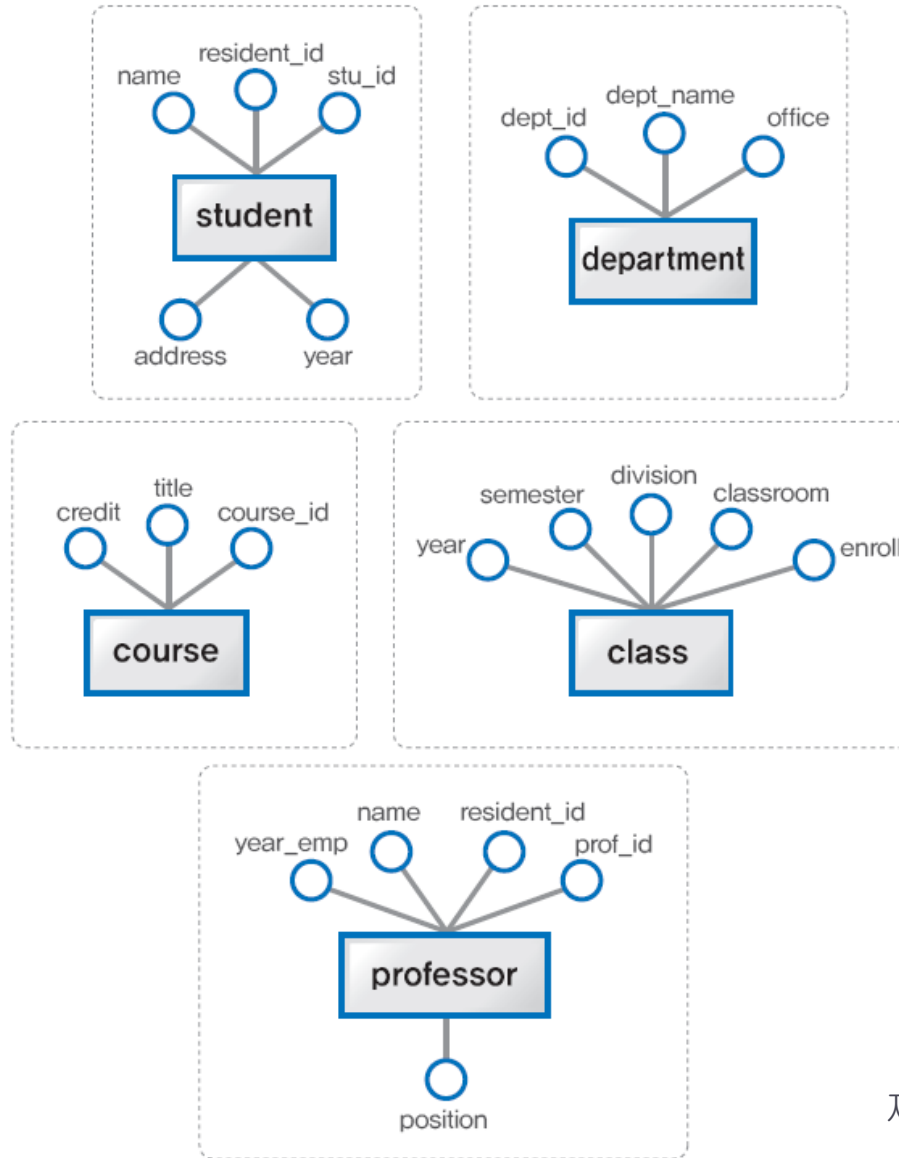
개체 집합의 도출

▶ 개체 집합

- ▶ 요구사항 문서에서 정형화된 중요한 개념

개체 집합	속성	개체 집합	속성
학생(student)	학번(stu_id) 주민등록번호(resident_id) 이름(name) 주소(address) 학년(year)	강좌(class)	연도(year) 학기(semester) 분반(division) 강의실(classroom) 수강인원(enroll)
학과(department)	학과번호(dept_id) 학과명(dept_name) 사무실(office)	교수(professor)	교수번호(prof_id) 주민등록번호(resident_id) 이름(name) 직급(position) 임용년도(year_emp)
교과목(course)	과목번호(course_id) 교과목명(title) 학점수(credit)		

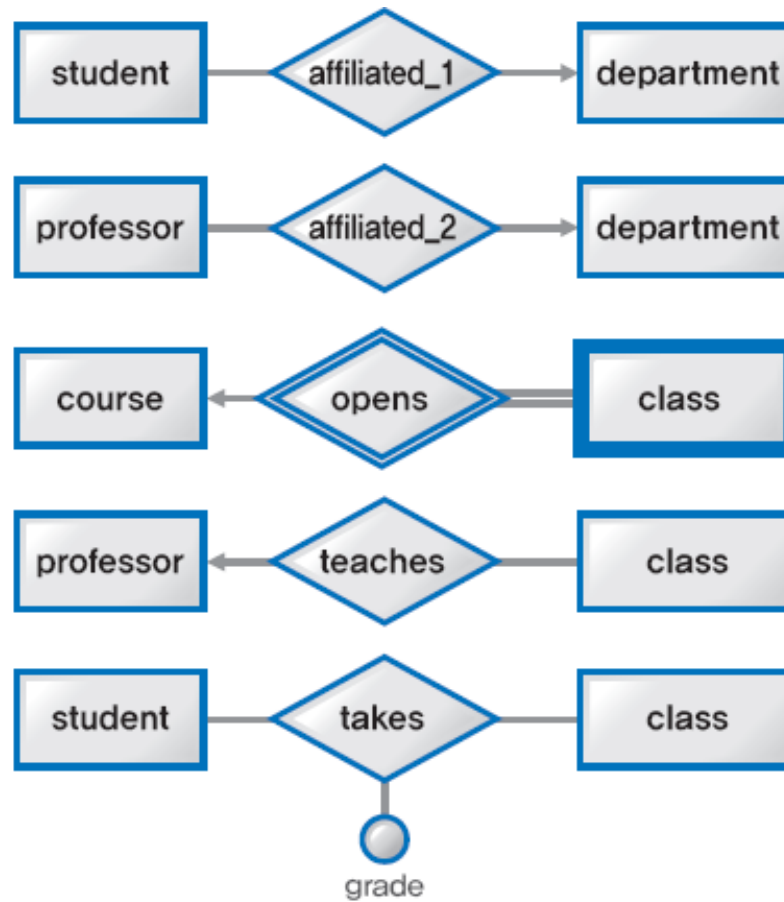
개체 집합 ERD



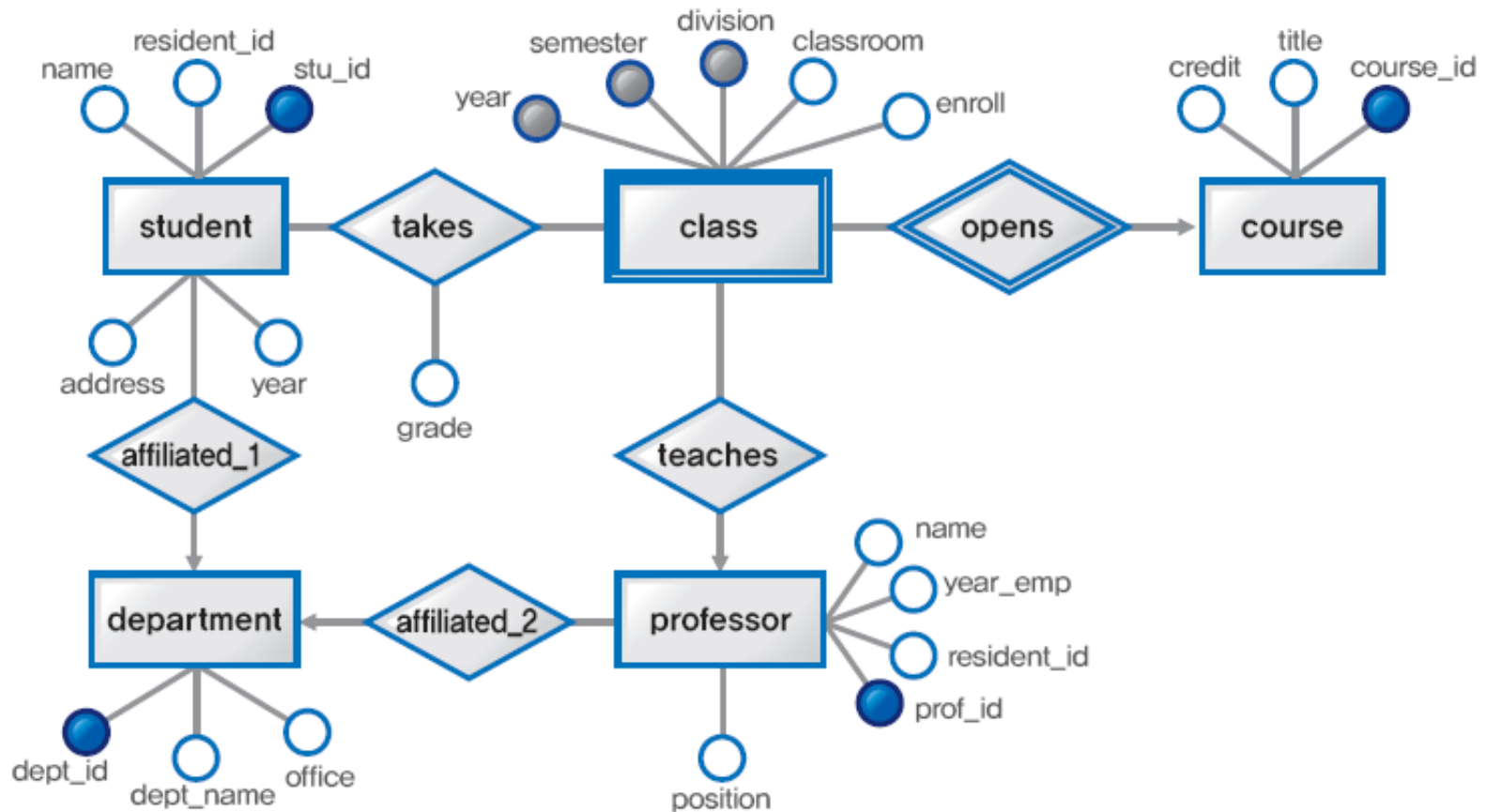
관계 집합 정의

- ▶ 소속(affiliated)
 - ▶ 학생(student)은 학과(department)에 소속(affiliated_1)된다.
 - ▶ 교수(professor)는 학과(department)에 소속(affiliated_2)된다.
- ▶ 개설(opens)
 - ▶ 각 교과목(course)은 학기별로 강좌(class)가 개설(opens)된다.
- ▶ 강의(teaches)
 - ▶ 교수(professor)는 개설된 강좌(class)를 강의(teaches)한다.
- ▶ 수강(takes)
 - ▶ 학생(student)은 강좌(class)를 수강(takes)한다.

관계 집합의 ERD



완성된 ERD (기본키 포함)



설계 순서

▶ 방법 1

- ▶ 개체집합들을 모두 결정하고 그들의 관계를 그 다음으로 결정하는 순으로 설계

▶ 방법 2

- ▶ 요구사항에서 가장 중요하다고 판단되는 개체집합과 관계집합(예를 들어 student와 class 간의 수강 관계)을 우선 결정
- ▶ 점점 확대하여 추가적인 개체집합(course, professor, department 등)과 관계 집합을 이끌어내는 순으로 설계

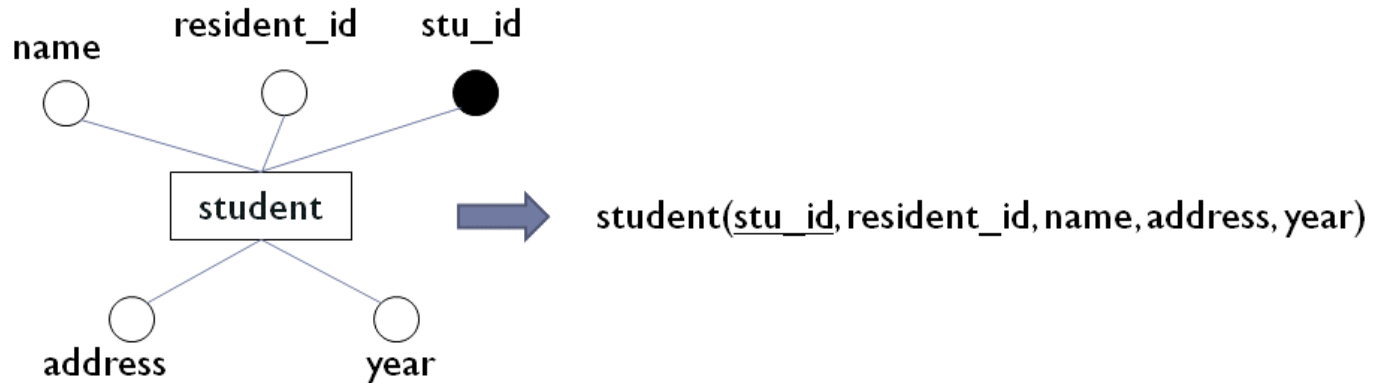
▶ 전적으로 설계자의 판단에 의해 결정

논리적 설계

- ▶ ERD로 부터 테이블 스키마 생성(변환)
- ▶ 논리적 설계 과정
 - ▶ 강성 개체집합을 관계형 테이블로 변환
 - ▶ 약성 개체집합을 관계형 테이블로 변환
 - ▶ 관계집합을 관계형 테이블로 변환
 - ▶ 중복되는 테이블 제거
 - ▶ 가능한 테이블들 결합

강성 개체집합의 변환

- ▶ 하나의 강성 개체집합 => 하나의 테이블
- ▶ 강성 개체집합의 속성 => 테이블 필드
- ▶ 테이블의 기본키는 개체집합의 기본키를 그대로 사용



강성 개체집합 변환 결과

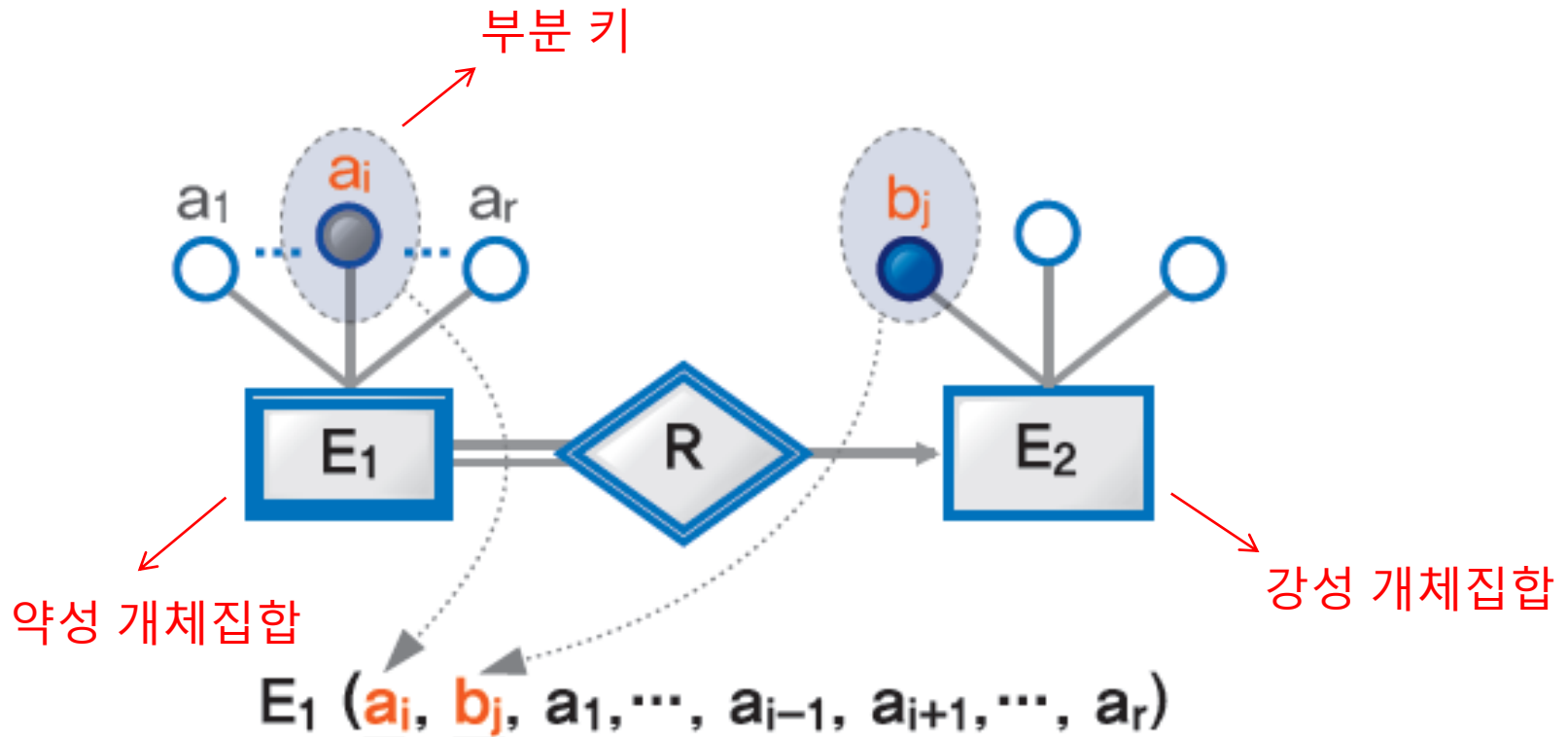
student(stu_id, resident_id, name, address, year)

department (dept_id, dept_name, office)

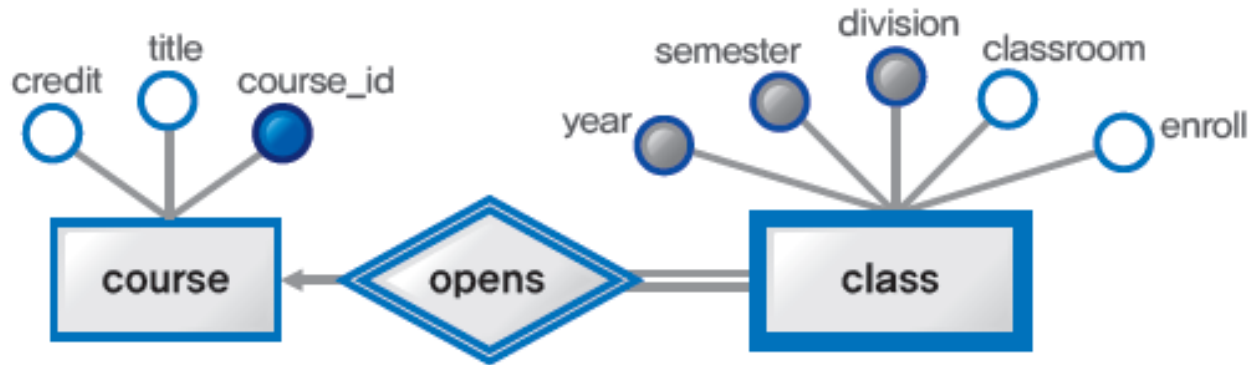
professor (prof_id, resident_id, name, position, year_emp)

course (course_id, title, credit)

약성 개체집합의 변환

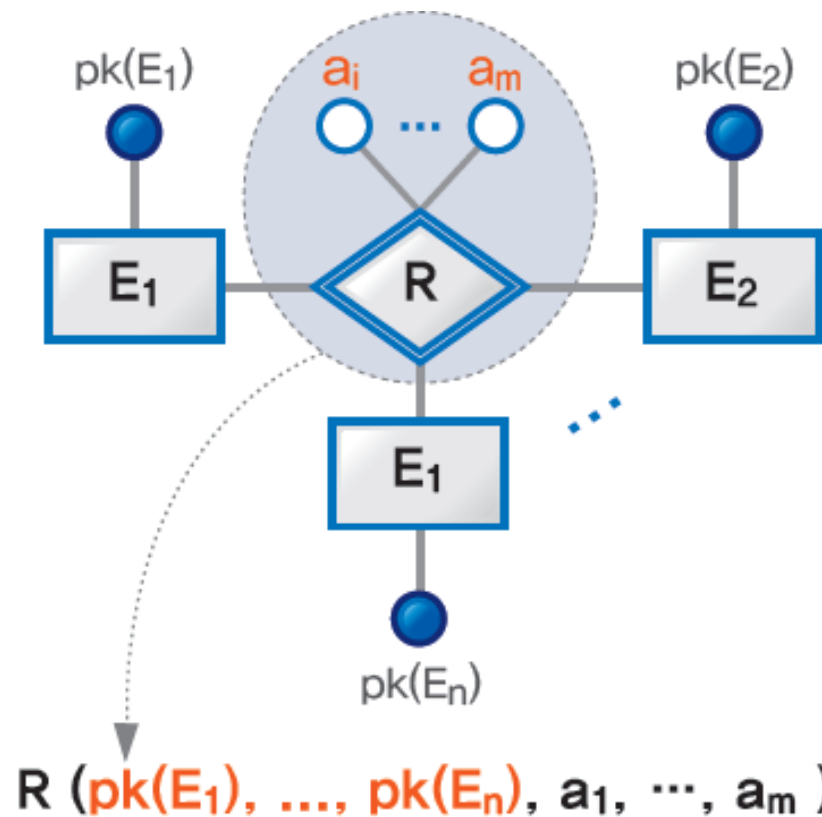


약성 개체집합 변환의 예 : 교과목(class)



class (course_id, year, semester, division, classroom, enroll)

관계 집합의 변환



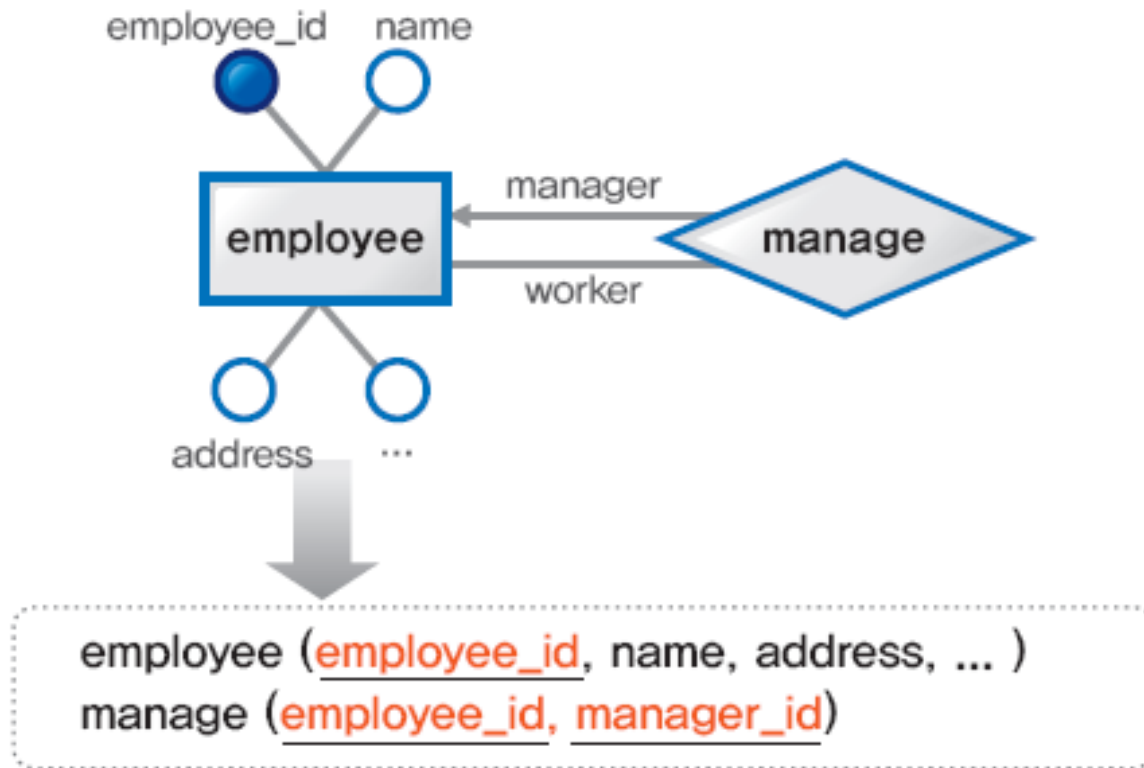
기본키는 두 관계집합의 기본키를 묶어서 설정

관계집합 변환 결과

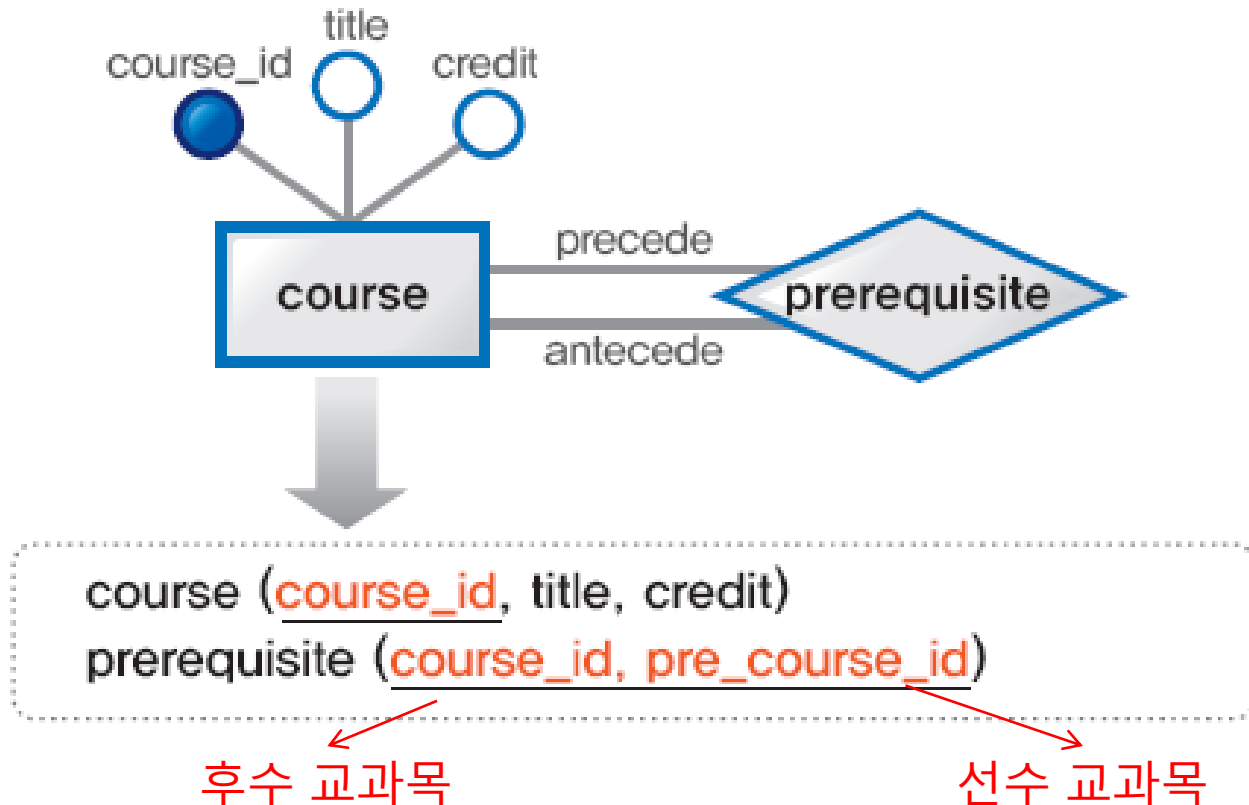
```
affiliated_1 (stu_id, dept_id)  
affiliated_2 (prof_id, dept_id)  
opens (course_id, year, semester, division)  
takes (stu_id, course_id, year, semester, division, grade)  
teaches (course_id, year, semester, division, prof_id)
```

자기연관 관계집합의 변환(일대다 대응)

- ▶ 개체집합 관계집합의 변환 규칙 사용
- ▶ 역할의 의미를 반영하여 기본키 명칭 변경 필요



자기연관 관계집합의 변환(다대다 대응)



테이블의 중복과 결합

- ▶ 지금까지 개체집합 5개와 관계집합 5개
 - ▶ 총 10개의 테이블로 변환
 - ▶ 관계집합으로부터 변환된 테이블은 경우에 따라 개체집합으로부터 변환된 테이블과 데이터 중복이 발생
- ▶ 중복이 발생한 테이블은 다른 테이블과 결합되어 하나의 테이블로 표현

테이블의 중복

- ▶ 관계집합에서 변화된 테이블

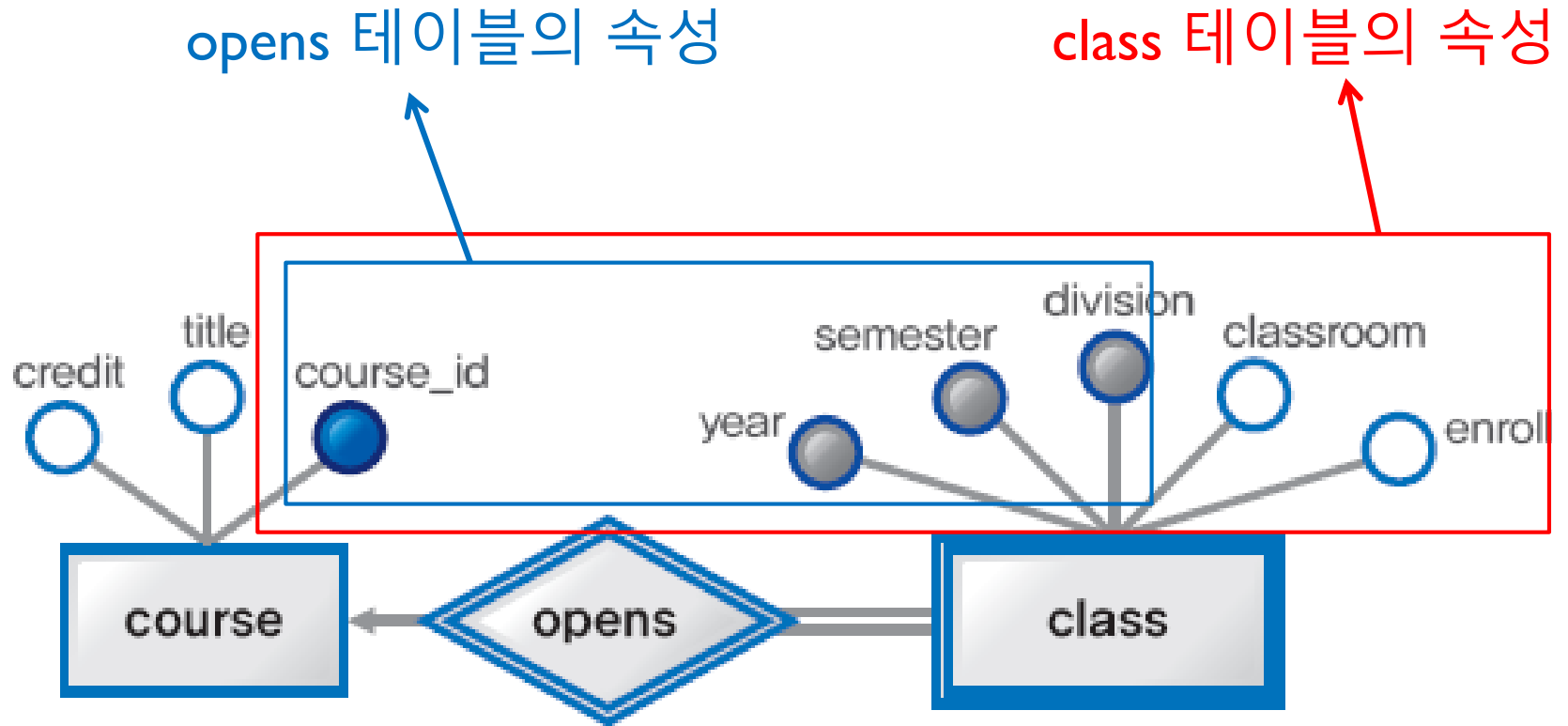
`opens (course_id, year, semester, division)`

- ▶ 개체집합에서 변화된 테이블

`class(course_id, year, semester, division, classroom, enroll)`

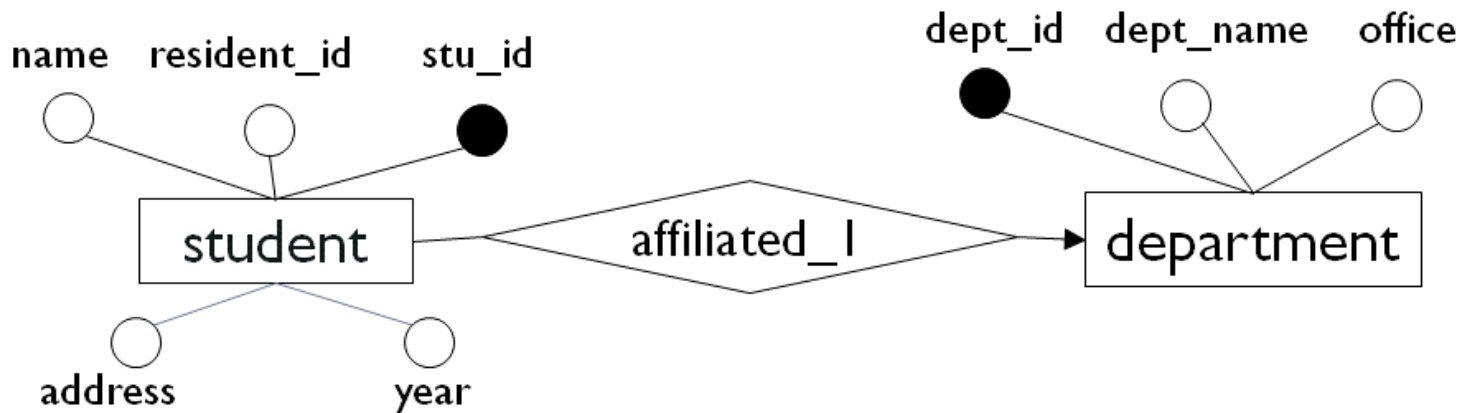
- ▶ 필드들이 중복되므로 관계 테이블 opens 생략 가능

테이블의 중복의 원인



테이블의 결합(다대일 관계)

- ▶ 다대일(일대다) 관계집합의 경우 관계 테이블과 개체 테이블 결합 가능



```
student(stu_id, resident_id, name, address, year)
department (dept_id, dept_name, office)
affiliated_1 (stu_id, dept_id)
```

테이블의 결합(다대일 관계)

- ▶ 관계테이블을 삭제
- ▶ 다대일 대응에서 '일'에 해당하는 개체집합의 기본키를 '다'에 해당하는 개체집합의 테이블에 외래키로 추가
- ▶ 개체집합에 속성이 있을 경우 같이 결합됨

```
student(stu_id, resident_id, name, address, year)  
department (dept_id, dept_name, office)  
affiliated_1 (stu_id, dept_id)
```

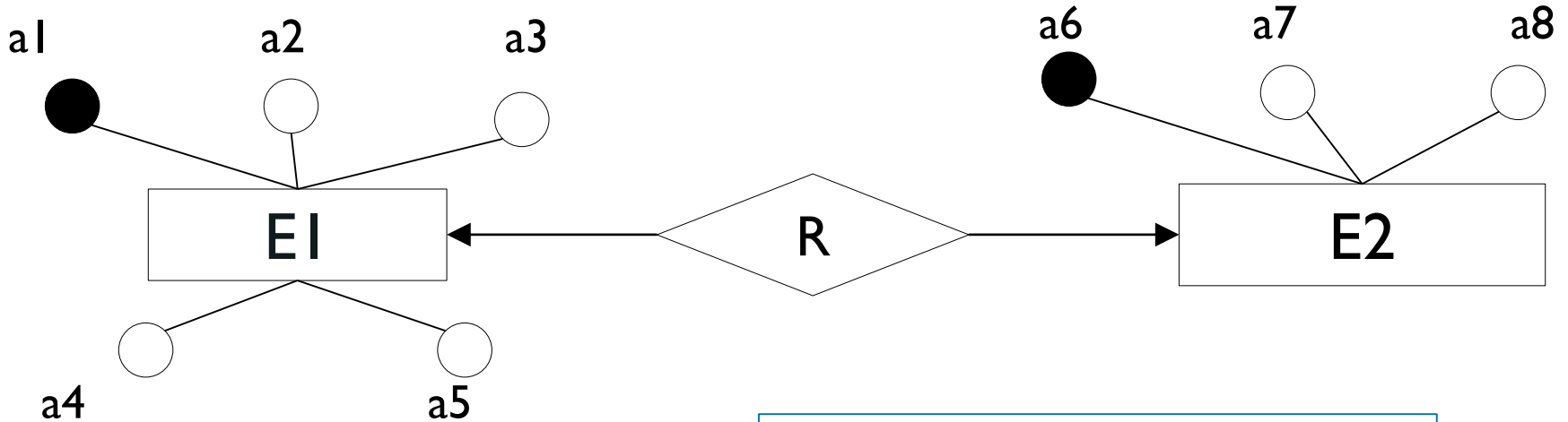


```
student(stu_id, resident_id, name, address, year, dept_id)  
department (dept_id, dept_name, office)
```

다대다 관계 테이블의 결합 방법

- ▶ 결합이 불가능함
- ▶ 결합할 경우 다대다의 표현이 불가능함
- ▶ 예에서 takes 테이블은 결합이 안됨

일대일 관계 테이블의 결합

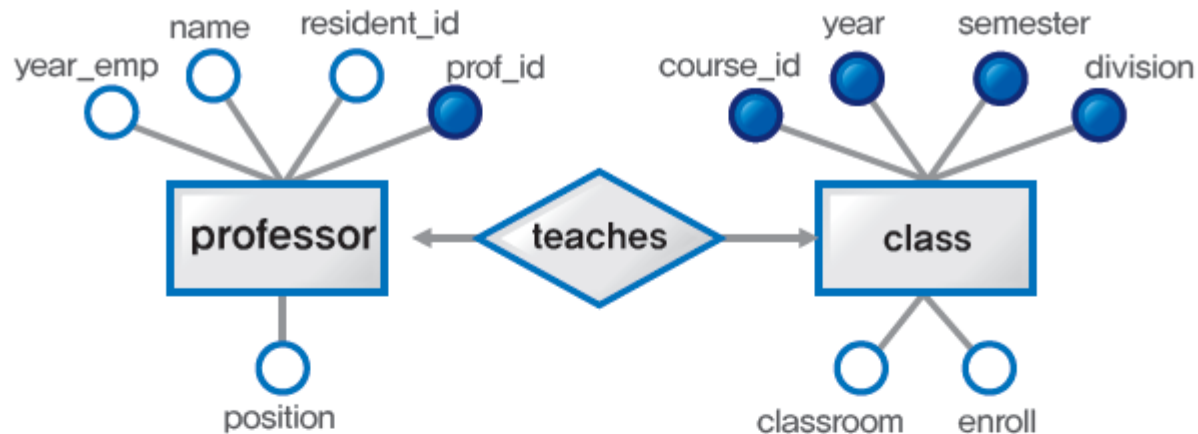


$E1(\underline{a1}, a2, a3, a4, a5)$
 $E2(\underline{a6}, a7, a8)$
 $R(\underline{a1}, \underline{a6})$

$E1(\underline{a1}, a2, a3, a4, a5, \textcolor{red}{a6})$
 $E2(\underline{a6}, a7, a8)$

$E1(\underline{a1}, a2, a3, a4, a5)$
 $E2(\underline{a6}, a7, a8, \textcolor{red}{a1})$

$E1(\underline{a1}, a2, a3, a4, a5, \textcolor{red}{a6}, \textcolor{red}{a7}, \textcolor{red}{a8})$



결합 전

a)

```

professor(prof_id, resident_id, name, year_emp, position)
class(course_id, year, semester, division, classroom, enroll)
teaches (prof_id, course_id, year, semester, division)
  
```

결합 1

b)

```

professor(prof_id, resident_id, name, year_emp, position)
class(course_id, year, semester, division, classroom, enroll, prof_id)
  
```

결합 2

c)

```

professor(prof_id, resident_id, name, year_emp, position, course_id, year, semester, division)
class(course_id, year, semester, division, classroom, enroll)
  
```

결합 3

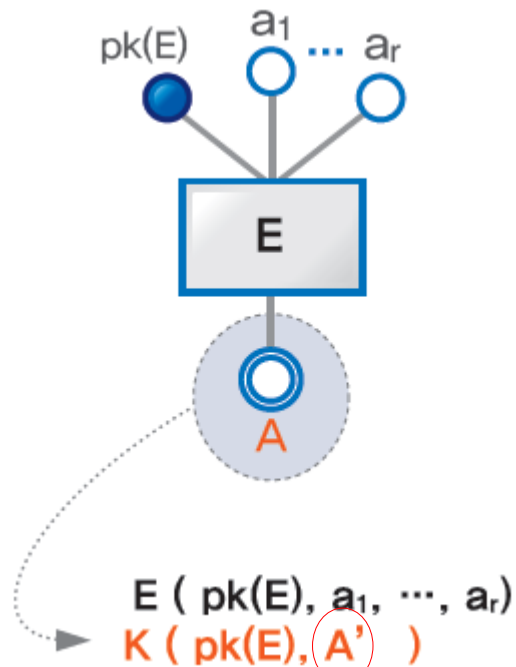
d)

```

professor(prof_id, resident_id, name, year_emp, position, course_id, year, semester,
division, classroom, enroll)
  
```

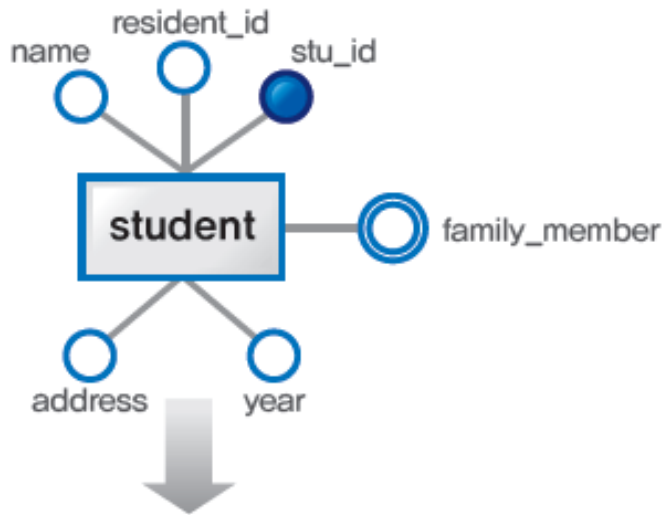
다중 값 속성의 변환

- ▶ 관계형 테이블의 속성은 원자 값만 가능
- ▶ 다중 값 속성에 대응되는 새로운 테이블 생성
- ▶ 기본키는 pk(E)와 A'으로 구성



다중 값 속성 A 에 들어갈
원자 값을 위한 속성

다중 값 속성 변환의 예



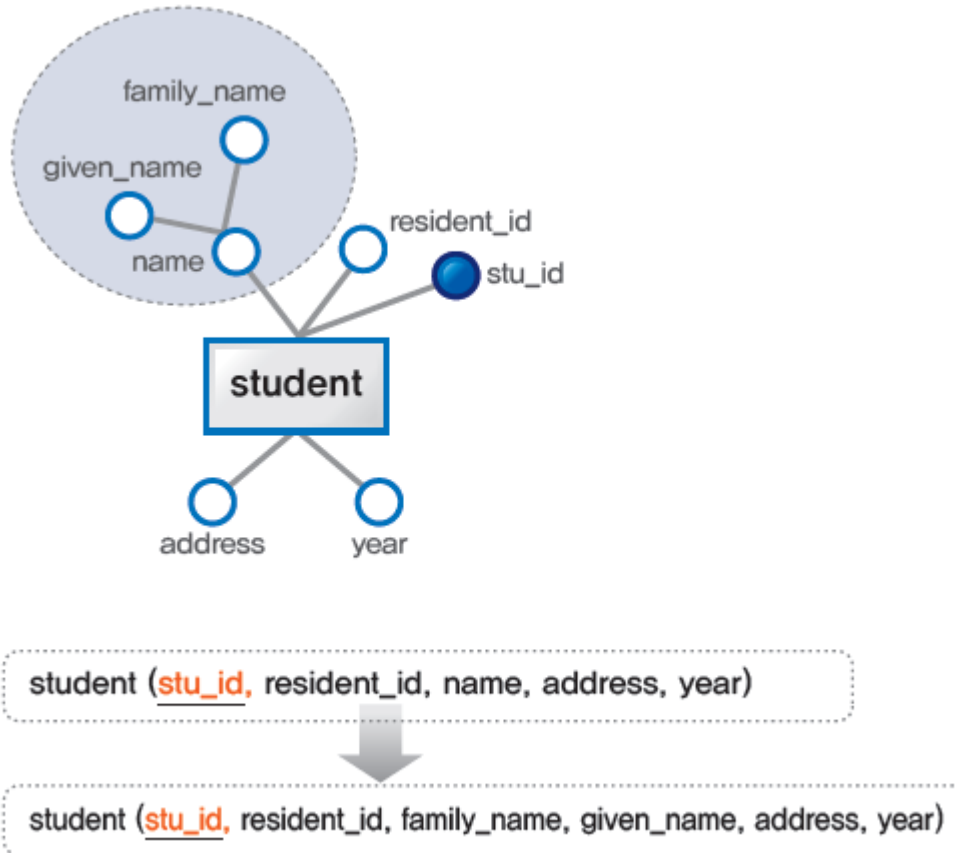
student (stu_id, resident_id, name, address, year)

family_member (stu_id, member_name)

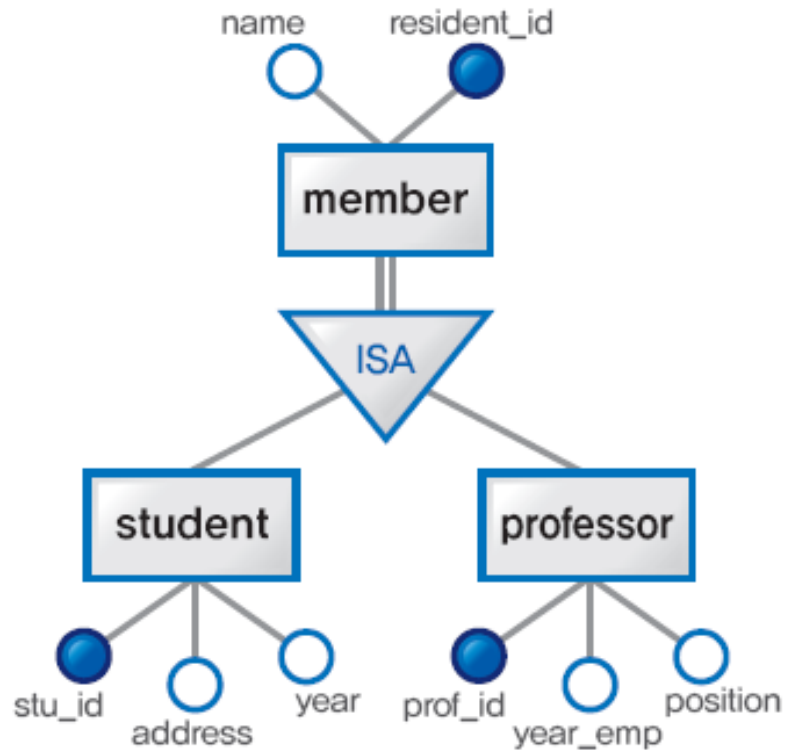
→ 다중 값의 개수 만큼 레코드 저장

복합 속성의 변환

- 복합 속성은 여러 개의 속성으로 분리



일반화 관계의 변환



일반화 관계의 변환

▶ 방법 1: 상위 개체집합의 유지

```
member (resident_id, name)
student (resident_id, stu_id, address, year)
professor (resident_id, prof_id, year_emp, position)
```

- ▶ 상위 개체집합에 존재하는 일부 개체가 하위 개체집합에 속하지 않는 경우(부분 참여)에 유용하게 활용
- ▶ 상위 개체집합 테이블과 하위 개체집합 테이블간의 빈번한 조인(join)이 수반됨
 - ▶ 예) 임용년도가 2010년도인 교수의 이름을 검색

▶ 방법 2: 상위 개체집합 제거

```
student (resident_id, name, stu_id, address, year)
professor (resident_id, name, prof_id, year_emp, position)
```

- ▶ 상위 개체집합에 속하는 모든 개체가 하위 개체집합에 속하는 경우에만 가능
- ▶ 공통 필드에 대한 UNION 연산이 자주 발생 가능

변환과정 요약

강성 개체집합의 변환	<ul style="list-style-type: none"> 개체집합의 속성은 테이블의 필드로 정의 개체집합의 기본키는 테이블의 기본키로 정의
약성 개체집합의 변환	<ul style="list-style-type: none"> 개체집합의 속성은 테이블의 필드로 정의 약성 개체집합의 기본키는 자신의 부분키와 해당 강성 개체집합의 기본키로 구성
관계집합의 변환	<ul style="list-style-type: none"> 관련 개체집합의 기본키들과 자신의 속성을 필드로 하여 테이블을 정의 해당 테이블의 기본키는 관련 개체집합의 기본키들의 집합으로 정의
관계형 테이블의 중복 제거	<ul style="list-style-type: none"> 약성 개체집합 및 일 대 다 대응관계를 가지는 경우, 관계집합에 해당하는 테이블을 삭제 다(many) 측 테이블에 일(one) 측 테이블의 기본키를 가지고 옴
자기연관 관계 집합의 변환	<ul style="list-style-type: none"> 일 대 다 관계인 경우, 관계집합에 해당하는 테이블을 정의할 필요가 없지만, 일(one) 측에 해당하는 역할(role)을 감안하여 새로운 필드를 정의 다 대 다 관계의 경우, 해당 관계집합의 기본키에 해당하는 2개의 필드를 정의하되, 역할(role)에 따라 필드명을 부여
다중값 속성의 변환	<ul style="list-style-type: none"> 복합 속성 자신의 속성과 관련 개체집합의 기본 키로 구성되는 별도의 테이블을 정의 생성된 테이블의 기본키는 해당 개체집합의 기본키로 설정
복합 속성의 변환	<ul style="list-style-type: none"> 복합 속성에 존재하는 세부 속성만으로 필드를 정의
일반화 관계 집합의 변환	<ul style="list-style-type: none"> 상위 개체집합을 유지하는 경우, 하위 개체집합에 해당하는 테이블은 상위 개체집합의 기본키를 가져옴 상위 개체집합을 유지하지 않는 경우, 하위 개체집합 해당하는 테이블은 자신의 속성과 상위 개체집합의 모든 속성을 포함하여 필드를 정의

최종 테이블 스키마

