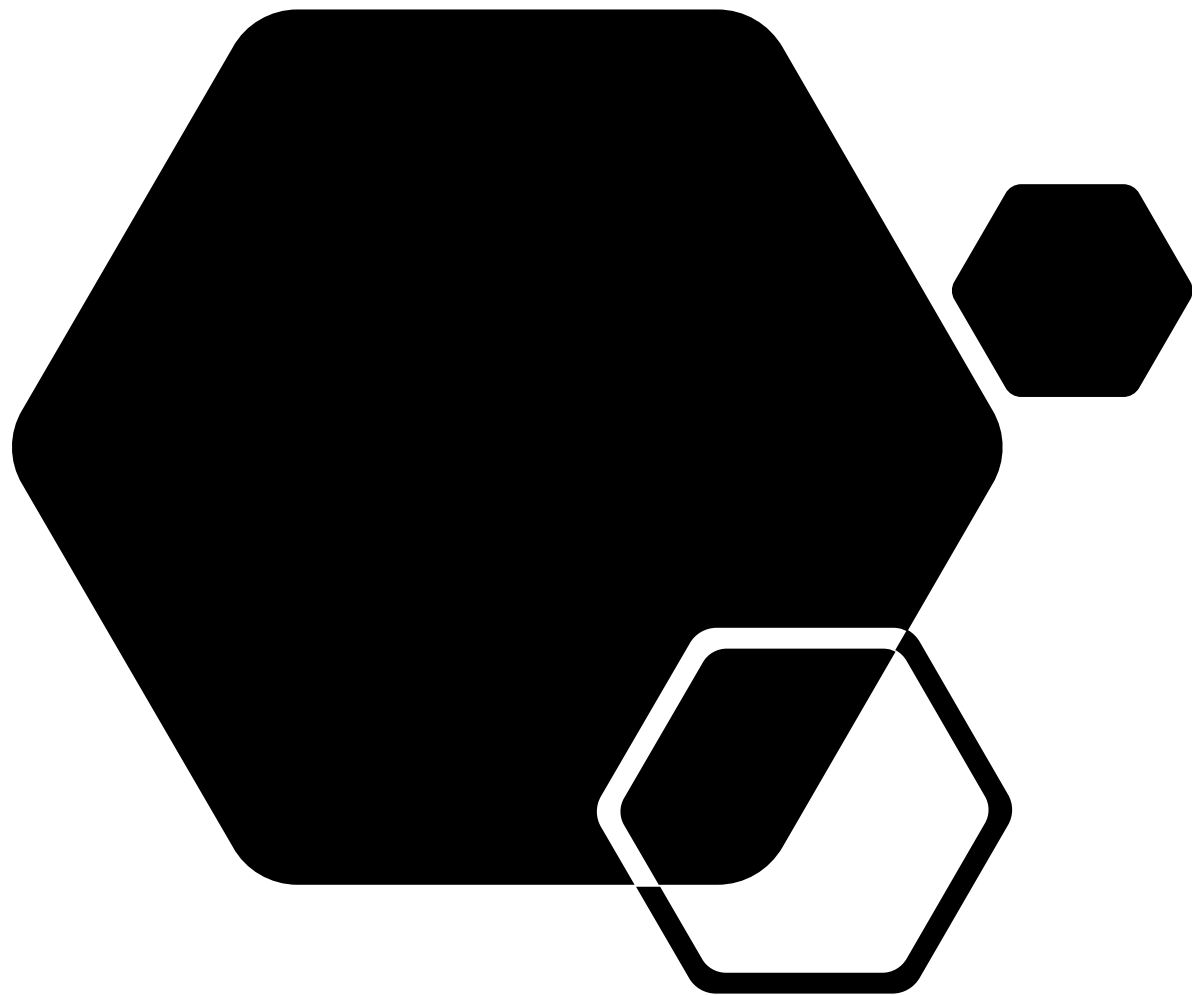


# RDBMS Design 실습



- 데이터 모델의 정의

- 데이터 모델(data model) : 데이터의 관계, 접근과 그 흐름에 필요한 처리 과정에 관한 추상화된 모형

- 개념적 데이터 모델, 논리적 데이터 모델, 물리적 데이터 모델

- 데이터 모델링(data modeling) : 주어진 개념으로부터 논리적인 데이터 모델을 구성하는 작업

- 개념적 데이터 모델






- 개념적 데이터 모델은 현실 세계의 요소를 인간이 이해할 수 있는 정보 구조로 표현.
- 개체와 개체 사이의 관계를 이용하여 현실 세계를 표현
- 개념적 데이터 모델로는 E-R 모델

- 논리적 데이터 모델

- 개념적 데이터 모델을 컴퓨터가 이해할 수 있도록 변환한 데이터 모델로 일반적으로 데이터 모델이라고 하면 논리적 데이터 모델을 의미
- 특정 DBMS는 특정 논리적 데이터 모델 하나만 선정하여 사용
- 논리적 데이터 모델에는 관계 모델, 계층 모델, 네트워크 모델.

- 개체 관계 모델(E-R model; Entity-Relationship model)

- 개체-관계 모델은 개념적 데이터 모델의 대표적 모델이며 개체 타입(Entity Type)과 이들 간의관계 타입(Relationship Type)을 이용해 현실 세계를 개념적으로 표현.
- 데이터를 개체(Entity), 관계(Relationship), 속성(Attribute)으로 묘사하며, 개체 - 관계 모델을 이용하여 현실 세계를 개념적으로 모델링한 결과물을 그림으로 표현한 것을 개체-관계 다이어그램(**E-R diagram**)이라 한다.
- E-R 다이어그램은 다음과 같은 기호를 이용하여 모델을 표현

기호	기호 이름	의미
	사각형	개체(Entity) 타입
	마름모	관계(Relationship) 타입
	타원	속성(Attribute)
	밑줄 타원	기본키 속성
	선, 링크	개체타입과 속성연결

## ■ 개체(entity)

- 현실 세계에서 조직을 운영하는 데 꼭 필요한 사람이나 사물과 같이 구별되는 모든 것을 개체로 표현
- 저장할 가치가 있는 중요 데이터를 가지고 있는 사람이나 사물, 개념, 사건 등이 개체가 될 수 있다.
- 개체는 다른 개체와 구별되는 이름을 가지고 있고, 각 개체만의 고유한 특성이나 상태, 즉 속성을 하나 이상 가지고 있다.
  - 예) 서점에 필요한 개체로는 고객, 책 등을 들 수 있으며, 학교에 필요한 개체로는 과목, 학생, 교수 등을 들 수 있다. 개체는 파일 구조의 레코드(record)와 대응
- E-R 다이어그램에서 사각형으로 표현하고 사각형 안에 이름을 표기하여 개체를 표현.

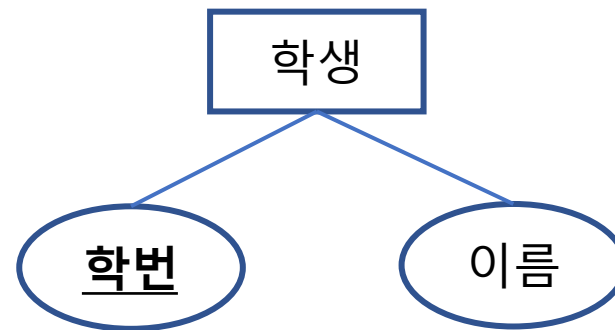
학생

- 속성(attribute)
  - 개체나 관계가 가지고 있는 고유의 특성이 속성
  - 의미있는 데이터의 가장 작은 논리적 단위로 이는 파일 구조의 필드(field)와 대응되는 용어
  - 속성은 E-R 다이어그램에서 타원으로 표현하고 타원 안에 속성 이름을 표기하여 표현

학번

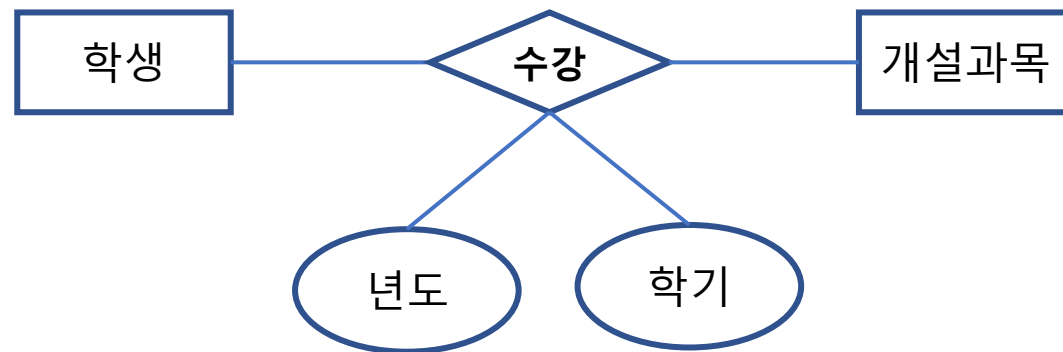
## ■ 키속성(key attribute)

- 개체를 구성하고 있는 속성이 실제 값으로 실체화된 개체를 인스턴스(instance)
  - 예) 한 학생의 학생 번호가 201803001'이고, 이름이 '홍길동'이라 할 때, 학생 개체의 <201803001, 홍길동> 이라는 인스턴스로 표현
- 개체의 인스턴스가 여럿 있을 때 각 인스턴스를 유일하게 식별하는 데 사용되는 속성을 키 속성(key attribute)
- 키 속성은 모든 인스턴스가 다른 값을 가져야 한다.
  - 학생 개체에서 '학번'은 모든 학생이 유일한 값을 가지므로 키 속성으로 사용.
- E-R 다이어그램에서 키 속성은 속성 이름에 밑줄



- 관계(relationship)

- 개체와 개체가 맺고 있는 의미 있는 연관성을 관계(relationship)
- 개체 집합들 사이의 대응 관계, 즉 매핑(mapping)을 의미
  - 예) 학생은 특정 학과에 소속되고(소속 관계), 개설 과목을 수강한다(수강 관계)
  - 학생은 개설 과목을 수강한다.
    - <학생> 개체와 <개설과목> 개체 사이에는 <수강> 관계가 성립
- 관계는 E-R 다이어그램에서 마름모로 표현한다.





- 개체와 개체 간의 관계

- 일대일(1:1) 관계

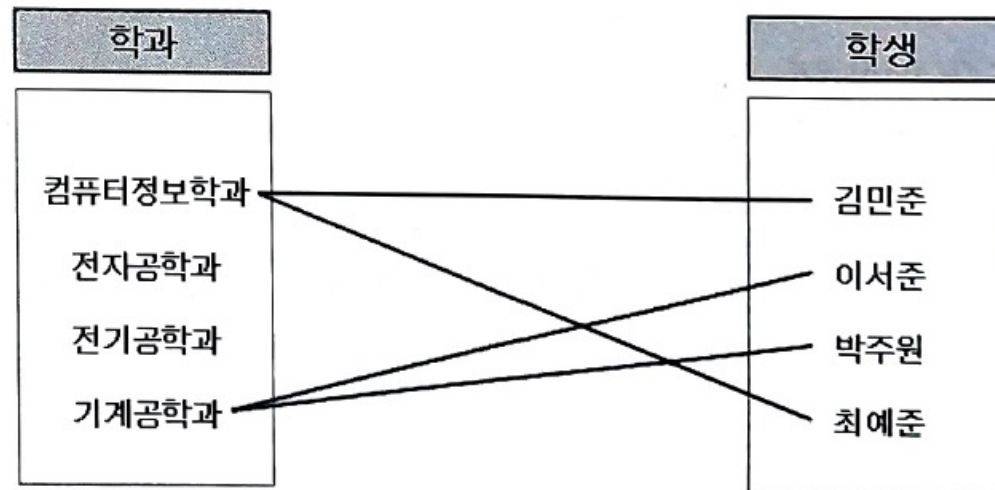
- A 개체의 오직 하나의 인스턴스가 B 개체의 한 인스턴스와 관계를 맺는 경우
      - *일대일 관계는 현실적으로 자주 발생하는 형태는 아님*

- 개체와 개체 간의 관계

- 일대다(1:n) 관계

- A개체의 한 인스턴스가 B개체의 여러 인스턴스와 관계를 맺을 수 있지만, 개체의 한 인스턴스는 A개체의 하나의 인스턴스와 관계를 맺을 수 있는 관계

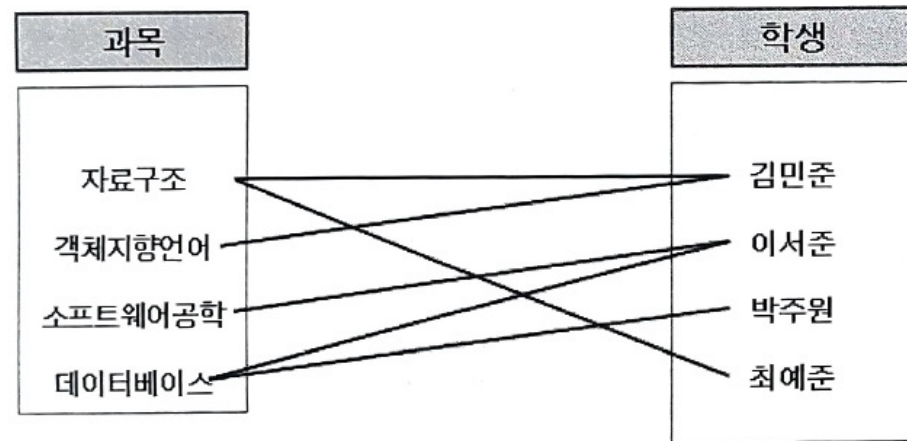
- 예) 한 학생은 오직 한 학과에만 소속될 수 있으며, 한 학과에는 여러 명의 학생이 소속



- 개체와 개체 간의 관계

- 다대다(n:m) 관계

- A개체의 여러 인스턴스가 B개체의 여러 인스턴스와 동시에 관계를 맺는 경우
  - 예) 한 학생은 여러 과목을 수강할 수 있으며, 한 과목은 여러 학생이 수강
  - 다대다 관계는 현실 세계에서 가장 자주 발생하는 형태



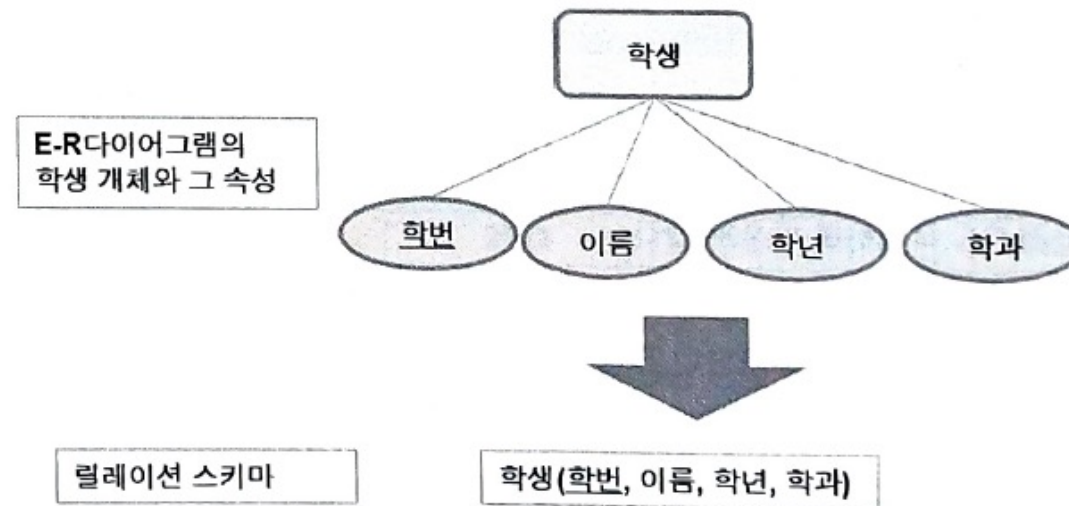
- 관계 데이터 모델

- 논리 데이터 모델이란 개념 데이터 모델을 상세화
  - 논리적인 데이터 집합, 관리 항목, 관계를 정의한 모델
  - 논리 데이터 모델은 전체 데이터 구조에서 가장 핵심을 이루는 모델로서 관계 모델, 계층 모델, 네트워크 모델이 있다.
- 관계 데이터 모델은 E-R 다이어그램으로 표현된 개념 데이터 모델을 이용하여 하나의 개체에 대한 데이터를 2차원 테이블 형태인 릴레이션으로 표현하는 방식
- 관계 데이터 모델을 컴퓨터에 구현한 것이 관계 데이터베이스 시스템

- 관계 데이터 모델
  - 논리 데이터 모델이란 개념 데이터 모델을 상세화
    - 논리적인 데이터 집합, 관리 항목, 관계를 정의한 모델
    - 논리 데이터 모델은 전체 데이터 구조에서 가장 핵심을 이루는 모델
      - 관계 모델, 계층 모델, 네트워크 모델
  - 관계 데이터 모델은 E-R 다이어그램으로 표현된 개념 데이터 모델을 이용하여 하나의 개체에 대한 데이터를 2차원 테이블 형태인 릴레이션으로 표현하는 방식
  - 관계 데이터 모델을 컴퓨터에 구현한 것이 관계 데이터베이스 시스템

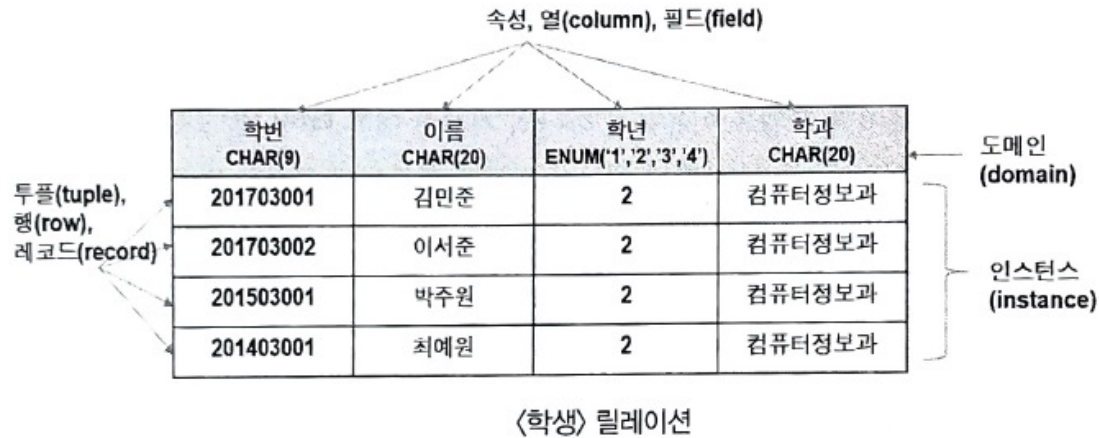
## ■ 릴레이션(Relation)

- E-R 다이어그램의 개체는 관계 데이터 모델에서 하나의 테이블 구조로 표현
- 릴레이션 스키마(relation schema)
  - 릴레이션의 논리적 구조를 나타내며 릴레이션의 이름과 릴레이션이 가지고 있는 모든 속성들로 정의
    - 예) E-R 다이어그램의 학생 개체는 학생 릴레이션으로 학생 개체의 모든 속성은 학생 릴레이션의 속성으로 표현



## ■ 릴레이션(Relation)

### ■ 학생 릴레이션 스키마를 릴레이션으로 표현



#### 릴레이션의 특징.

- 튜플의 유일성 : 한 릴레이션에는 중복된 튜플이 존재하지 않는다.
- 튜플의 무순서 : 한 릴레이션에 포함된 튜플 사이의 순서는 의미 없다.
- 속성의 무순서 : 한 릴레이션을 구성하는 속성 사이의 순서는 의미 없다.
- 속성의 원자값 : 모든 속성값은 원자값만을 가진다.

- 속성(attribute) : 릴레이션(테이블)를 구성하는 항목들로 열(column), 필드(field)와 같은 개념.
- 튜플(tuple) : 릴레이션의 행을 구성하는 속성 값들의 집합으로 행(row), 레코드(record)와 같은 개념.
- 도메인(domain) : 각각의 속성이 가질 수 있는 값을 정의해 놓은 것을 도메인.
- 인스턴스(instance) : 릴레이션 스키마에 따라 테이블에 실제로 저장되어있는 데이터의 집합.

## ■ 키(Key)

- 릴레이션에서 튜플들을 유일하게 구별하는 속성 또는 속성들의 집합
- 기본키(Primary key)
  - 한 릴레이션에서 특정 튜플을 유일하게 구별할 수 있는 속성으로 null값을 가질 수 없다.
  - 당연히 기본키로 정의된 속성에는 동일한 값을 중복하여 저장할 수 없다.
- 외래키(Foreign key)
  - 두개의 릴레이션 <A>와 <B>에서 <A> 릴레이션의 특정 속성이 <B> 릴레이션의 기본키를 참조할 때, 이 <A> 릴레이션의 속성을 외래키
  - 외래키는 참조하는 릴레이션의 기본키와 대응되어 릴레이션 간에 **참조 관계를 표현**하는데 중요한 도구로 사용
  - 외래키로 지정되면 참조 테이블의 기본키에 없는 값은 입력할 수 없다.

학생 릴레이션

학생(학번, 이름, 학년, 학과)

수강 릴레이션

수강(수강번호, 년도, 학기, **학번**)

학생 릴레이션

학번	이름	학년	학과
201703001	김민준	2	컴퓨터정보과
201703002	이서준	2	컴퓨터정보과
201503001	박주원	2	컴퓨터정보과
201403001	최예원	2	컴퓨터정보과

기본키

수강 릴레이션

수강번호	년도	학기	학번
201703001	2018	2	201703001
201703002	2018	2	201703002
201503001	2018	2	201503001
201403001	2018	2	201403001

외래키



- 무결성 제약 조건

- 데이터가 정확하고 유효하게 유지되어 결함이 없는 상태를 무결성이라 한다. 데이터의 무결성을 보장하고 일관된 상태를 유지하기 위한 무결성 제약 조건이 있다

- 개체 무결성 제약 조건

- 릴레이션에서 기본키 속성은 NULL값이나 중복 값을 가질 수 없는 조건

- 예) 개체 무결성 사례 : <학생> 릴레이션에서 '학번' 속성은 NULL값을 가질 수 없으며, 고유한 값을 가져야 한다.

학생 릴레이션

개체 무결성 위반 사례

기본키값의 중복 사용

기본키값으로 NULL사용

학번	이름	학년	학과
201703001	김민준	2	컴퓨터정보과
NULL	이서준	2	컴퓨터정보과
201503001	박주원	2	컴퓨터정보과
201703001	최예월	2	컴퓨터정보과

- 무결성 제약 조건

- 참조 무결성 제약 조건

- 참조할 수 없는 외래키 값을 가질 수 없다.

- 예)<수강> 릴레이션에서 '학번' 속성은 <학생> 릴레이션의 '학번' 속성에 존재하는 값만을 저장할 수 있다.

참조 무결성 위반 사례

학생 릴레이션

기본키

학번	이름	학년	학과
201703001	김민준	2	컴퓨터정보과
201703002	이서준	2	컴퓨터정보과
201503001	박주원	2	컴퓨터정보과
201403001	최예원	2	컴퓨터정보과

수강 릴레이션

수강번호	년도	학기	학번
201703001	2018	2	201703001
201703002	2018	2	201703002
201503001	2018	2	201803007
201403001	2018	2	201403001

학생 릴레이션에  
없는 학번 사용

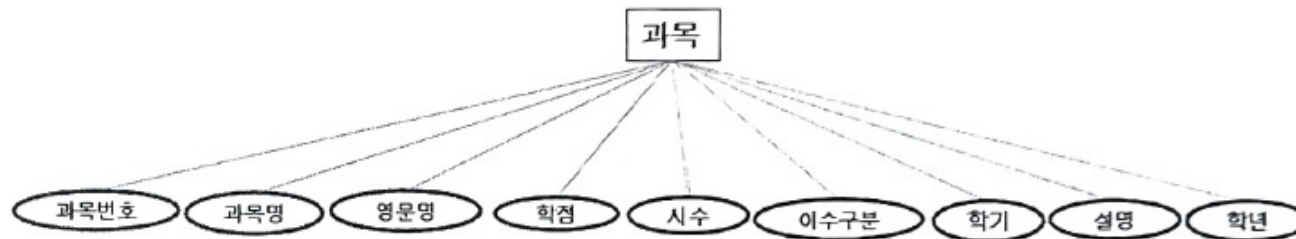
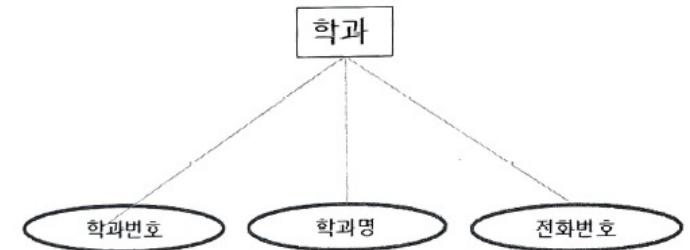
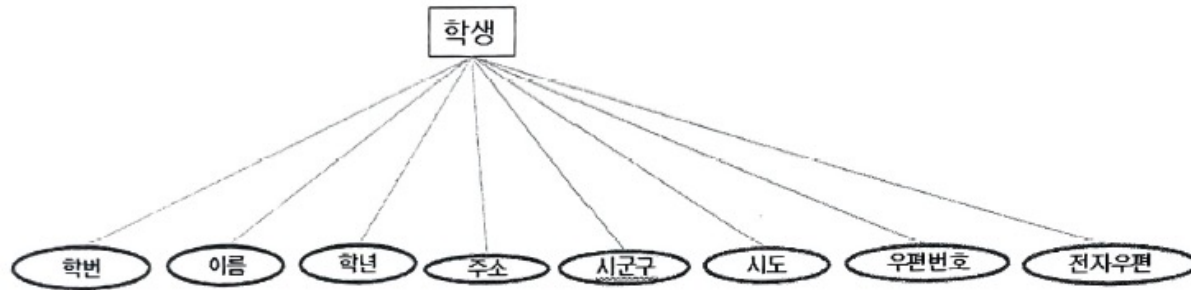
외래키

## 『수강신청』 데이터베이스 설계

- (1단계) 요구사항분석
  - 모든 과목은 하나의 반으로 운영되며, 한 교수가 담당한다.
  - 모든 과목은 강좌 개설이 되며, 매년 같은 교수가 담당한다.
  - 하나의 학과에 여러명의 교수와 학생이 소속된다.
  - 교수는 여러과목을 수업하고 학생은 여러 과목을 수강할 수 있다.
- (2단계) 개념적 설계(ERD 설계)
  - 개체, 속성, 관계 파악

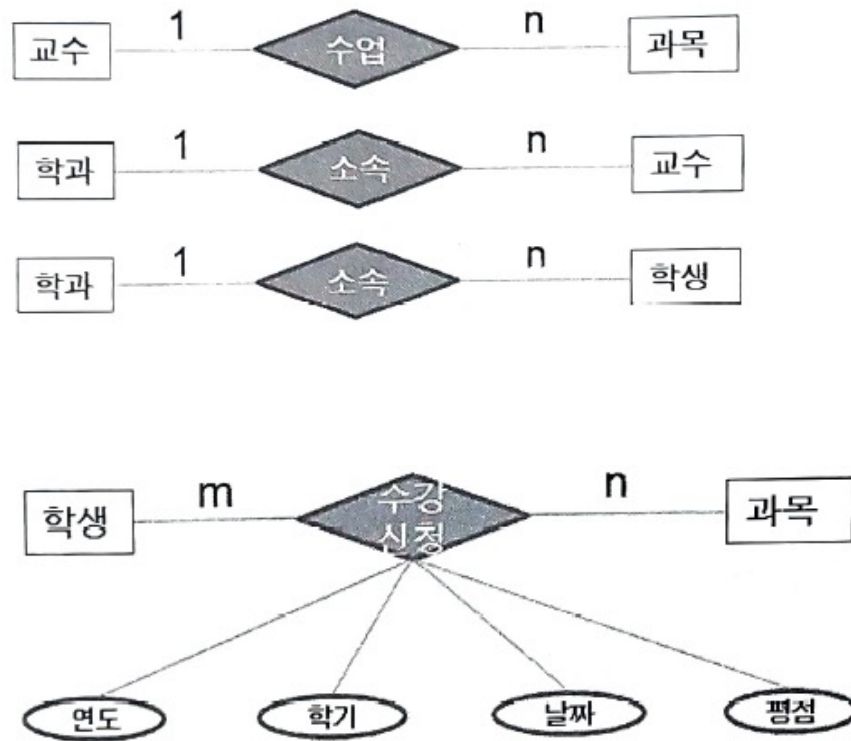
## 『수강신청』 데이터베이스 설계

- (2단계) 개념적 설계(ERD 설계)
- 개체
  - 학생, 과목, 교수, 학과



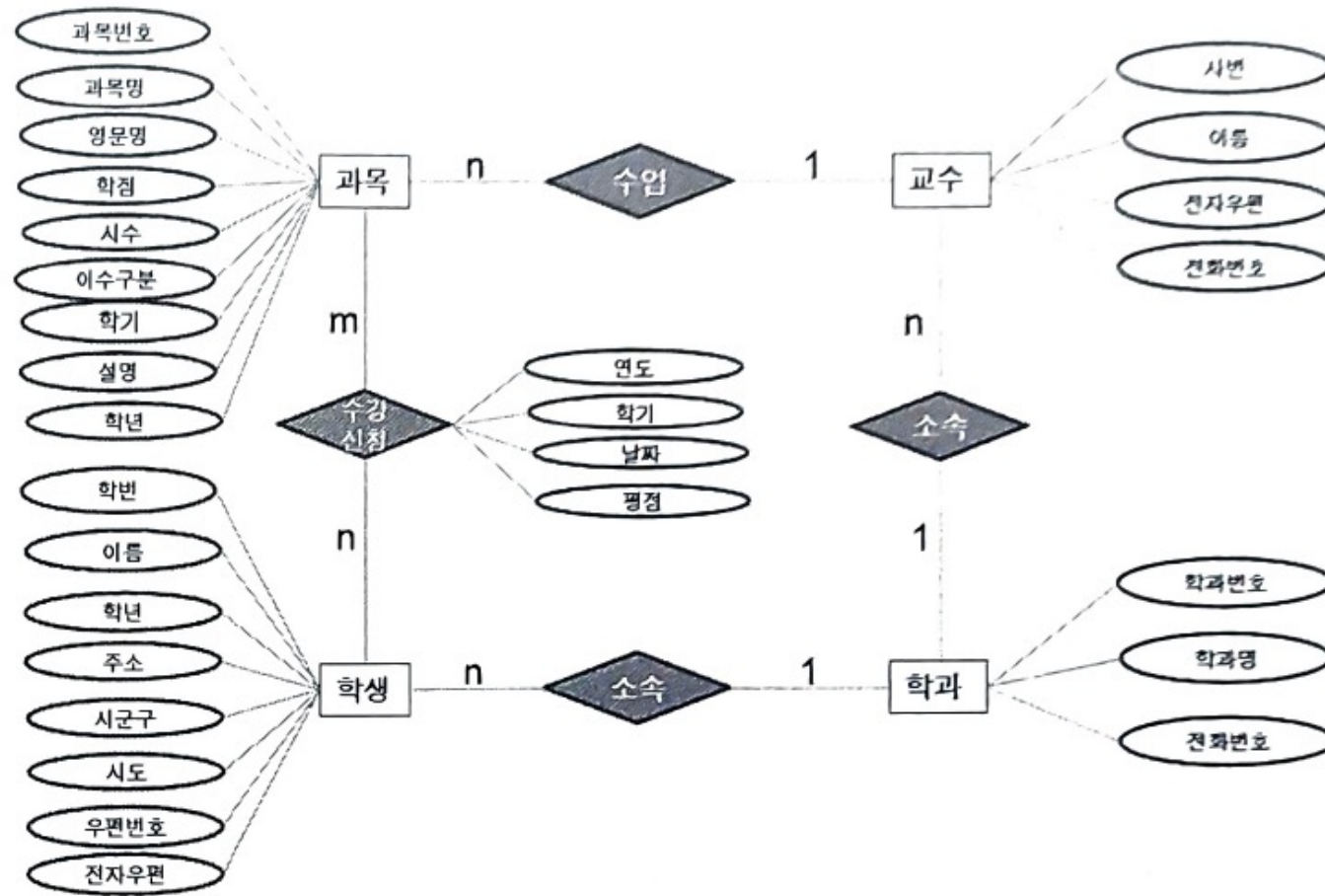
## 『수강신청』 데이터베이스 설계

- (2단계) 개념적 설계(ERD 설계)
  - 관계



## 『수강신청』 데이터베이스 설계

### ■ (2단계) 개념적 설계(ERD 설계)

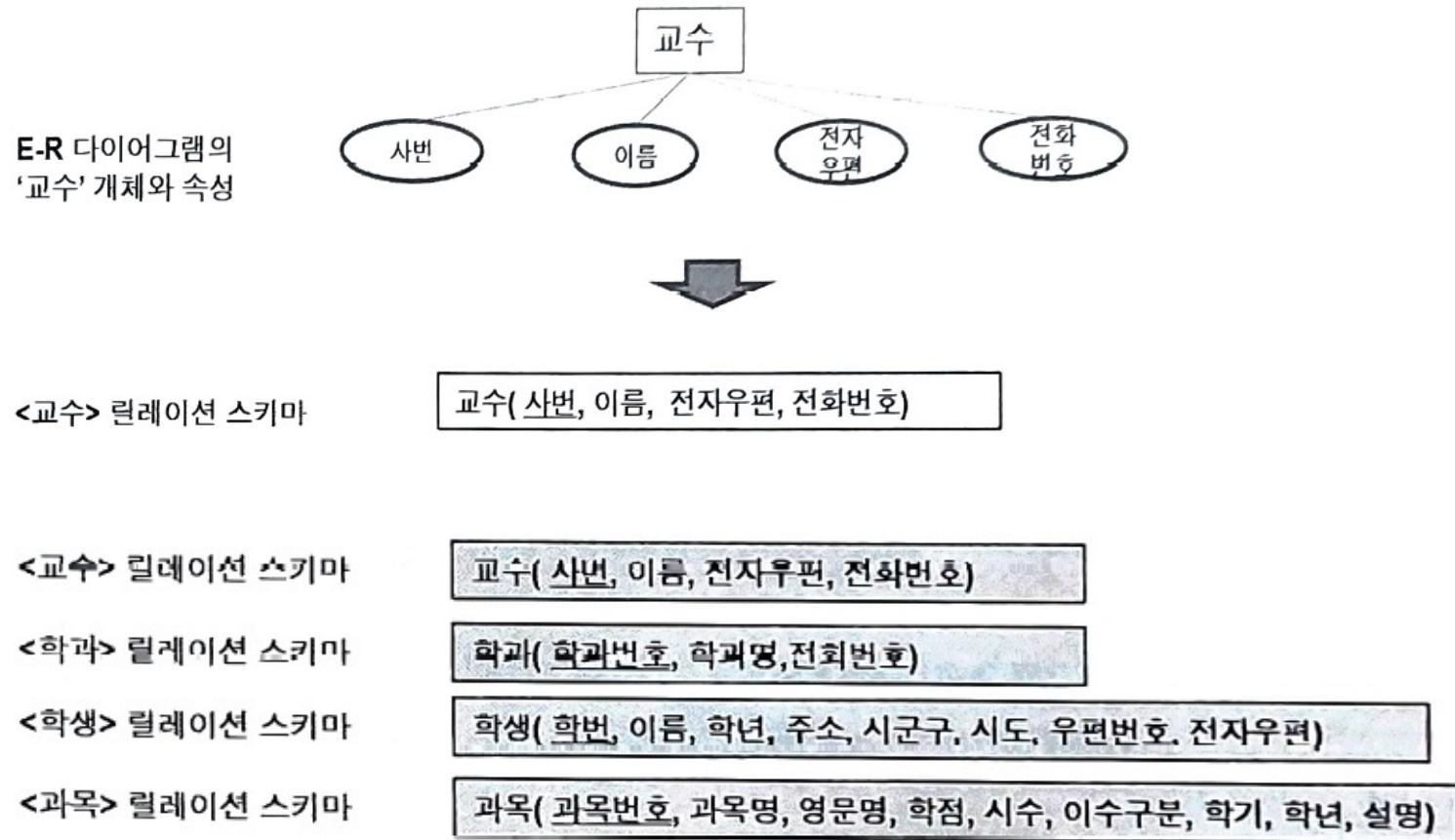


## 『수강신청』 데이터베이스 설계

- (3단계) 논리적 설계
  - 개념적 설계에서 도출한 ERD를 이용하여 릴레이션 스키마를 설계하는 과정
  - 규칙1: 모든 개체는 릴레이션으로 변환
  - 규칙2: 다대다(n:m) 관계는 릴레이션으로 변환
  - 규칙3: 일대일(1:1), 일대다(1:n)관계는 외래키로 표현
  - 규칙4: 다중값 속성은 릴레이션으로 변환
  - 릴레이션 스키마는 데이터의 중복을 막고 무결성을 강화하기 위해 하나의 릴레이션을 둘 이상으로 분리하는 정규화 과정을 거쳐 최종 릴레이션 스키마 완성

## 『수강신청』 데이터베이스 설계

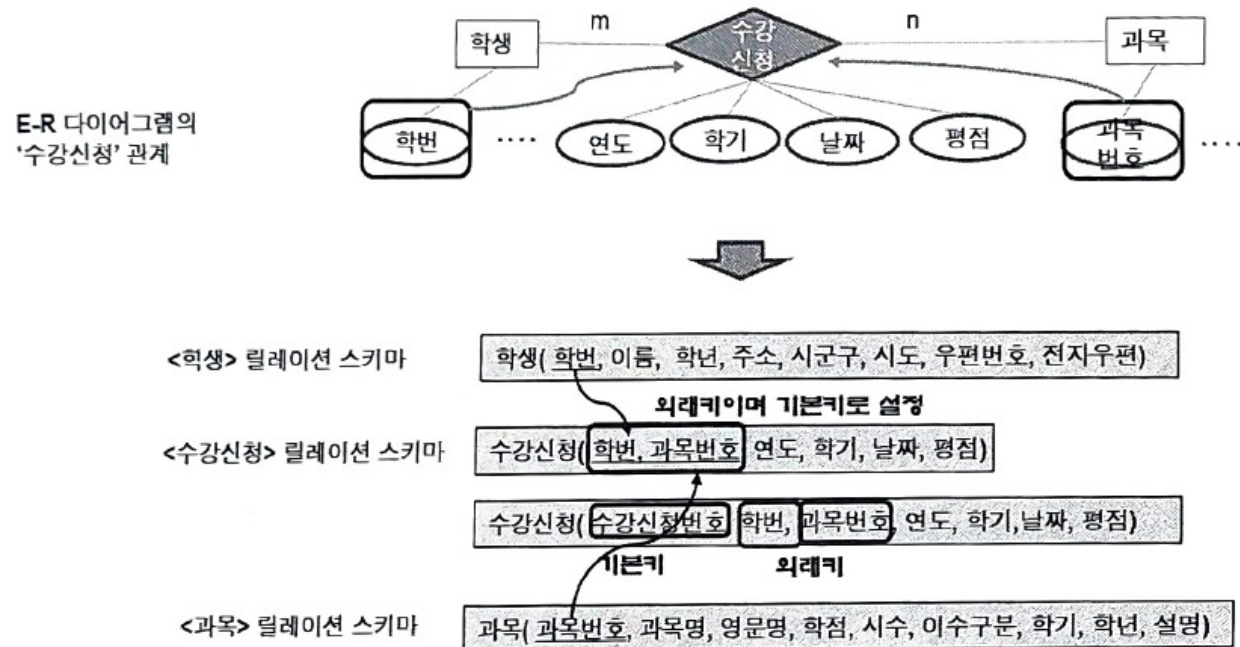
- 수강신청 시스템 릴레이션 스키마 설계
  - 규칙1: 모든 개체는 릴레이션으로 변환





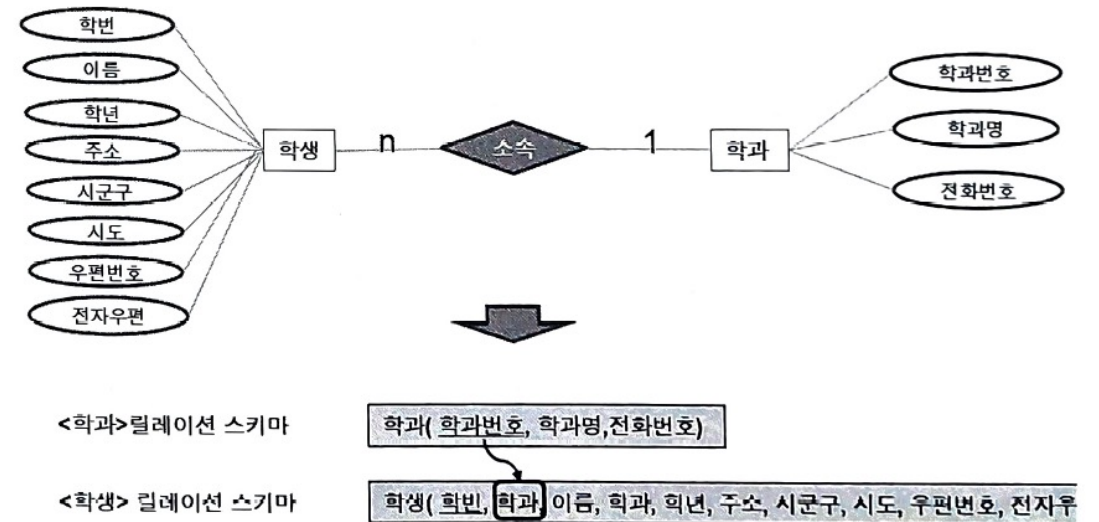
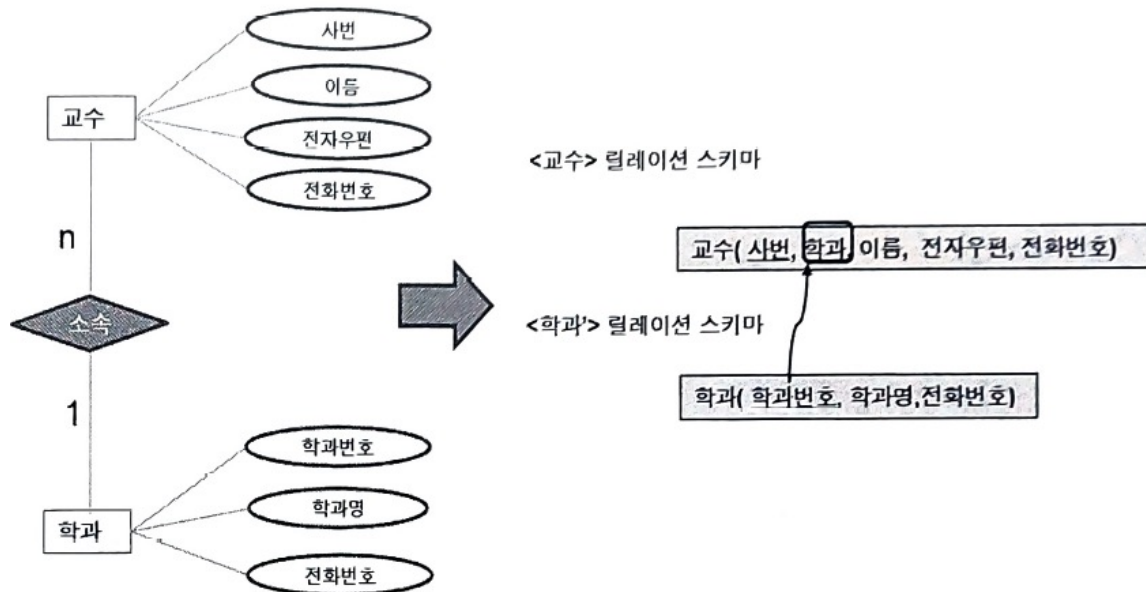
## 『수강신청』 데이터베이스 설계

- 수강신청 시스템 릴레이션 스키마 설계
  - 규칙2: 다대다(n:m) 관계는 릴레이션으로 변환
    - 기본키는 외래키 2개를 묶어서 기본키로 정하거나
    - 기본키로 사용할 새로운 속성(수강신청번호) 추가



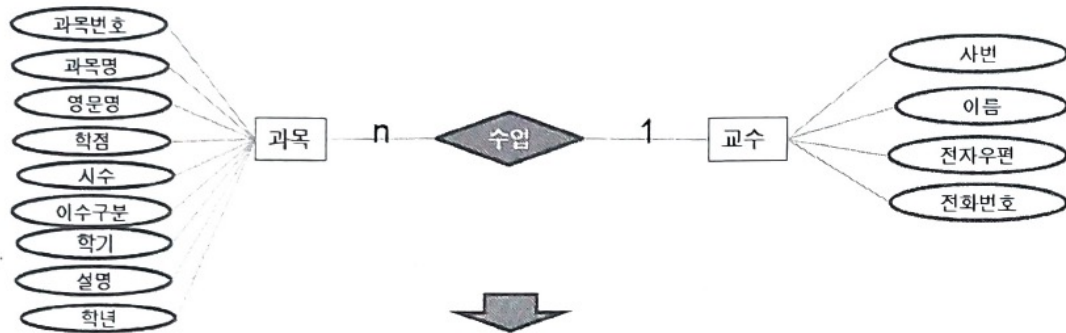
# 『수강신청』 데이터베이스 설계

- 수강신청 시스템 릴레이션 스키마 설계
- 규칙3: 일대일(1:1), 일대다(1:n)관계는 외래키로 표현



# 『수강신청』 데이터베이스 설계

- 수강신청 시스템 릴레이션 스키마 설계
- 규칙3: 일대일(1:1), 일대다(1:n)관계는 외래키로 표현



<교수> 릴레이션 스키마

교수( 사번, 이름, 전자우편, 전화번호 )

<과목> 릴레이션 스키마

과목( 과목번호, 담당교수, 과목명, 학점, 시수, 이수구분, 학기, 학년, 설명 )

<교수> 릴레이션 스키마

<학과> 릴레이션 스키마

<학생> 릴레이션 스키마

<수강신청> 릴레이션 스키마

<과목> 릴레이션 스키마

교수( 사번, 학과, 이름, 전자우편, 전화번호 )

학과( 학과번호, 학과명, 전화번호 )

학생( 학번, 학과, 이름, 학과, 학년, 주소, 시군구, 시도, 우편번호, 전자우편 )

수강신청( 수강신청번호, 학번, 과목번호, 연도, 학기, 날짜, 평점 )

과목( 과목번호, 담당교수, 과목명, 영문명, 학점, 시수, 이수구분, 학기, 학년, 설명 )

## 『수강신청』 데이터베이스 설계

### ■ 수강신청 시스템 릴레이션 스키마 설계

■ 정규화 : 관계형 데이터베이스 설계에서 데이터 중복을 최소화하도록 구조화하는 프로세스

### ■ 정규화 단계

- ① 제 1 정규화(First Normalization)
- ② 제 2 정규화(Second Normalization)
- ③ 제 3 정규화(Third Normalization)
- ④ 보이스-코드 정규화(Boyce-Codd Normalization)
- ⑤ 제 4 정규화(Fourth Normalization)
- ⑥ 제 5 정규화(Fifth Normalization)
- ⑦ 비정규화(De-normalization)

- 기본적으로 높은 차수의 정규화는 낮은 차수의 정규화를 모두 만족하여야 한다.
- 예) 제3정규형은 제1, 2 정규화의 조건을 모두 만족

## 『수강신청』 데이터베이스 설계

- 수강신청 시스템 릴레이션 스키마 설계
  - 제 1 정규화(First Normalization)
    - 릴레이션(테이블) 내의 속성이 단일 값을 가지도록 하는 것

'과목번호' 속성이 다중값을 가짐

수강신청번호	학번	과목번호	연도	학기	날짜	평점
00001	201703001	K20005, K20012, K20030, Y00132	2018	2	2018/8/29	-1
00002	201703002	K20004, K20012, Y00133	2018	2	2018/8/30	-1
00003	201503001	K20005, Y00132	2018	2	2018/8/31	-1

기본키      제1정규화

수강신청번호	일련번호	학번	과목번호	연도	학기	날짜	평점
00001	1	201703001	K20005,	2018	2	2018/8/29	-1
00001	2	201703001	K20012	2018	2	2018/8/29	-1
00001	3	201703001	K20030	2018	2	2018/8/29	-1
00001	4	201703001	Y00132	2018	2	2018/8/29	-1
00002	1	201703002	K20004	2018	2	2018/8/30	-1
00002	2	201703002	K20012	2018	2	2018/8/30	-1
00002	3	201703002	Y00133	2018	2	2018/8/30	-1
00003	1	201503001	K20005	2018	2	2018/8/31	-1
00003	2	201503001	Y00132	2018	2	2018/8/31	-1

수강신청 > 릴레이션의 제1정규화

## 『수강신청』 데이터베이스 설계

- 수강신청 시스템 릴레이션 스키마 설계

- 제 2 정규화(Second Normalization)

- 제2정규화는 기본키의 특정한 속성에 종속적인 속성이 없어야 한다.

- <수강신청> 릴레이션의 경우 기본키는 (수강신청번호, '일련번호') 두개의 속성으로 '수강신청번호'와 '일련번호' 속성값을 합쳐야만 하나의 튜플을 다른 튜플과 구분할 수 있다.

기본키

수강신청번호	일련번호	학번	과목번호	연도	학기	날짜	평점
00001	1	201703001	K20005,	2018	2	2018/8/29	-1
00001	2	201703001	K20012	2018	2	2018/8/29	-1
00001	3	201703001	K20030	2018	2	2018/8/29	-1
00001	4	201703001	Y00132	2018	2	2018/8/29	-1
00002	1	201703002	K20004	2018	2	2018/8/30	-1
00002	2	201703002	K20012	2018	2	2018/8/30	-1
00002	3	201703002	Y00133	2018	2	2018/8/30	-1
00003	1	201503001	K20005	2018	2	2018/8/31	-1
00003	2	201503001	Y00132	2018	2	2018/8/31	-1

수강신청번호' 속성에 종속적인 학번, 연도, 학기, 날짜) 속성



## 『수강신청』 데이터베이스 설계

### ■ 수강신청 시스템 릴레이션 스키마 설계

#### ■ 제 2 정규화 (Second Normalization)

- (학번, 연도, 학기, 날짜) 속성은 기본키 중 '수강신청번호' 속성에만 종속되어 있다.
- '수강신청번호' 속성값을 알면 (학번, 연도, 학기, 날짜) 속성값을 알 수 있다는 뜻으로 (학번, 연도, 학기, 날짜) 속성값은 중복 입력되어 있는 것으로 볼 수 있다.

기본키

수강신청번호	학번	연도	학기	날짜
00001	201703001	2018	2	2018/8/29
00002	201703002	2018	2	2018/8/30
00003	201503001	2018	2	2018/8/31

↓ <수강신청내역> 릴레이션

기본키

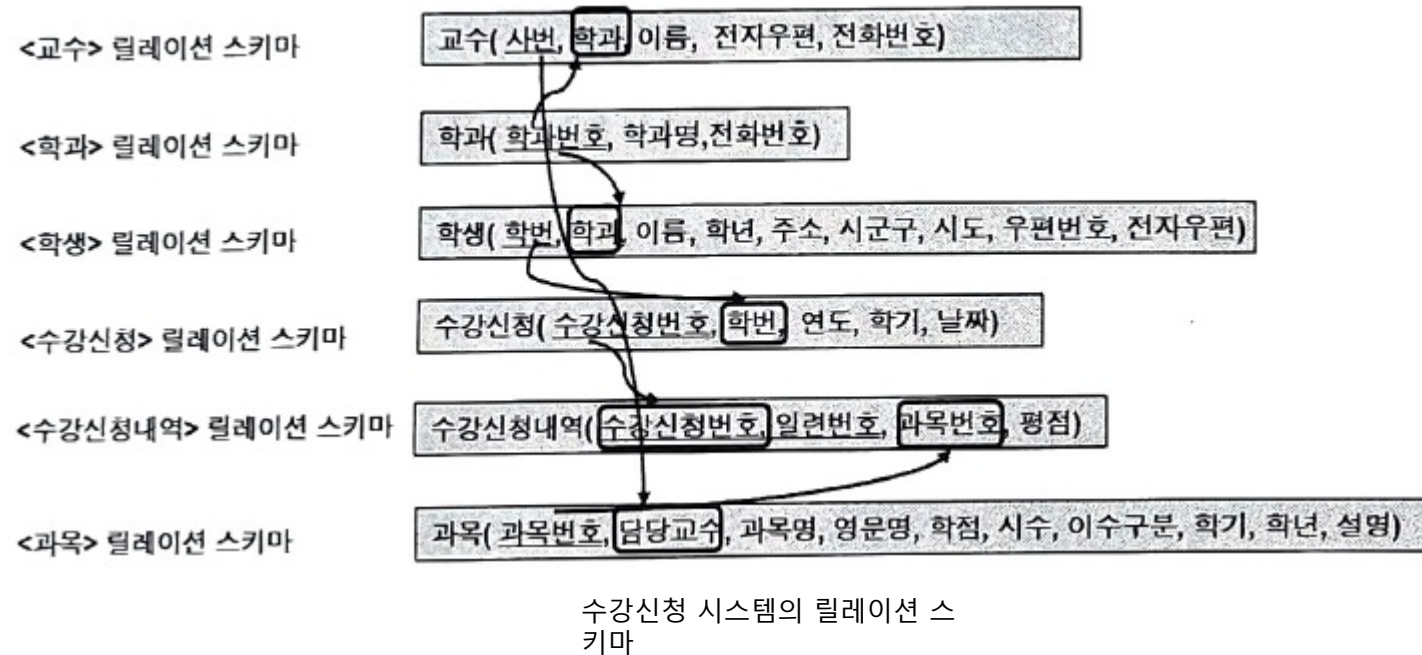
수강신청번호	일련번호	과목번호	평점
00001	1	K20005,	-1
00001	2	K20012	-1
00001	3	K20030	-1
00001	4	Y00132	-1
00002	1	K20004	-1
00002	2	K20012	-1
00002	3	Y00133	-1
00003	1	K20005	-1
00003	2	Y00132	-1

외래키

제2정규화 : 릴레이션의 분리

## 『수강신청』 데이터베이스 설계

- 수강신청 시스템 릴레이션 스키마 설계
- 제 3 정규화(Third Normalization)
  - 특정 속성의 값들이 키가 아닌 다른 속성에 종속적일 경우 이를 다른 릴레이션으로 분리





## 『수강신청』 데이터베이스 설계

- 수강신청 시스템 테이블 명세서 작성
- 수강신청 시스템 테이블 명세서 작성

- <학과> 테이블

테이블 이름	필드 이름	데이터 형식	NULL 유무	기본값	기본 키	외래 키	FK 테이블 이름	FK 열 이름	제약조건	설명
학과	학과번호	CHAR(2)	N		PK					학과 고유 번호
	학과명	CHAR(20)	N							학과 이름
	전화번호	CHAR(20)	N							학과사무실 전화번호

- <교수> 테이블

테이블 이름	필드 이름	데이터 형식	NULL 유무	기본값	기본 키	외래 키	FK 테이블 이름	FK 열 이름	제약조건	설명
교수	사번	CHAR(7)	N		PK					교수 사원 번호
	이름	CHAR(20)	N							교수 이름
	학과	CHAR(2)	N			FK	학과	학과번호		소속 학과
	전자우편	CHAR(50)	N							교수 e-mail
	전화번호	CHAR(20)	Y							교수 전화번호

## 『수강신청』 데이터베이스 설계

- 수강신청 시스템 테이블 명세서 작성
- 수강신청 시스템 테이블 명세서 작성

- <학생> 테이블

테이블 이름	필드 이름	데이터 형식	NULL 유무	기본 값	기본 키	외래 키	FK 테이블 이름	FK 열 이름	제약조건	설명
학생	학번	CHAR(7)	N		PK					학생 번호
	학과	CHAR(2)	N			FK	학과	학과번호		소속 학과
	이름	CHAR(20)	N							학생 이름
	학년	CHAR(1)	N						1,2,3,4 만 허용	소속 학년
	주소	CHAR(200)	Y							주소
	시군구	CHAR(20)	Y							주소 시군구
	시도	CHAR(20)	Y							주소 도/시
	우편번호	CHAR(20)	Y							우편번호
	전자우편	CHAR(50)	Y							E-메일 주소

- <과목> 테이블

테이블 이름	필드 이름	데이터 형식	NULL 유무	기본 값	기본 키	외래 키	FK 테이블 이름	FK 열 이름	제약조건	설명
과목	과목번호	CHAR(6)	N		PK					교과목 고유 번호
	과목명	CHAR(50)	N							교과목 명
	영문명	CHAR(50)	N							교과목 영문명
	담당교수	CHAR(7)	N			FK	교수	사번		과목 담당 교수
	학점	INT	N							학점수
	시수	INT	N							주당 시간수
	이수구분	CHAR(20)	N						교양, 전공 만 허용	이수구분
	학기	CHAR(10)	N						1,여름,2,겨울 만 허용	개설 학기
	학년	CHAR(1)	N						1,2,3,4 만 허용	해당 학년
	설명	TEXT	Y							교과목 설명

## 『수강신청』 데이터베이스 설계

- 수강신청 시스템 테이블 명세서 작성
- 수강신청 시스템 테이블 명세서 작성
  - <수강신청> 테이블

테이블 이름	필드 이름	데이터 형식	NULL 유무	기본 값	기본 키	외래 키	FK 테이블 이름	FK 열 이름	제약조건	설명
수강신청	수강신청 번호	CHAR(7)	N		PK					수강신청 번호
	학번	CHAR(7)	N			FK	학생	학번		수강 학생
	날짜	DATE TIME	N							수강신청 날짜
	연도	CHAR(4)	N							수강 연도
	학기	CHAR(10)	N						1,여름,2,겨울 만 허용	수강 학기

- <수강신청내역> 테이블

테이블 이름	필드 이름	데이터 형식	NULL 유무	기본 값	기본 키	외래 키	FK 테이블 이름	FK 열 이름	제약조건	설명
수강신청 내역	수강신청 번호	CHAR(7)	N		PK	FK	수강신청	수강신청 번호		수강신청번호
	일련번호	INT	N		PK					일련번호
	과목번호	CHAR(6)	N			FK	과목	과목번호		신청 과목
	평점	INT	N	-1						취득 평점

## 『수강신청』 데이터베이스 설계

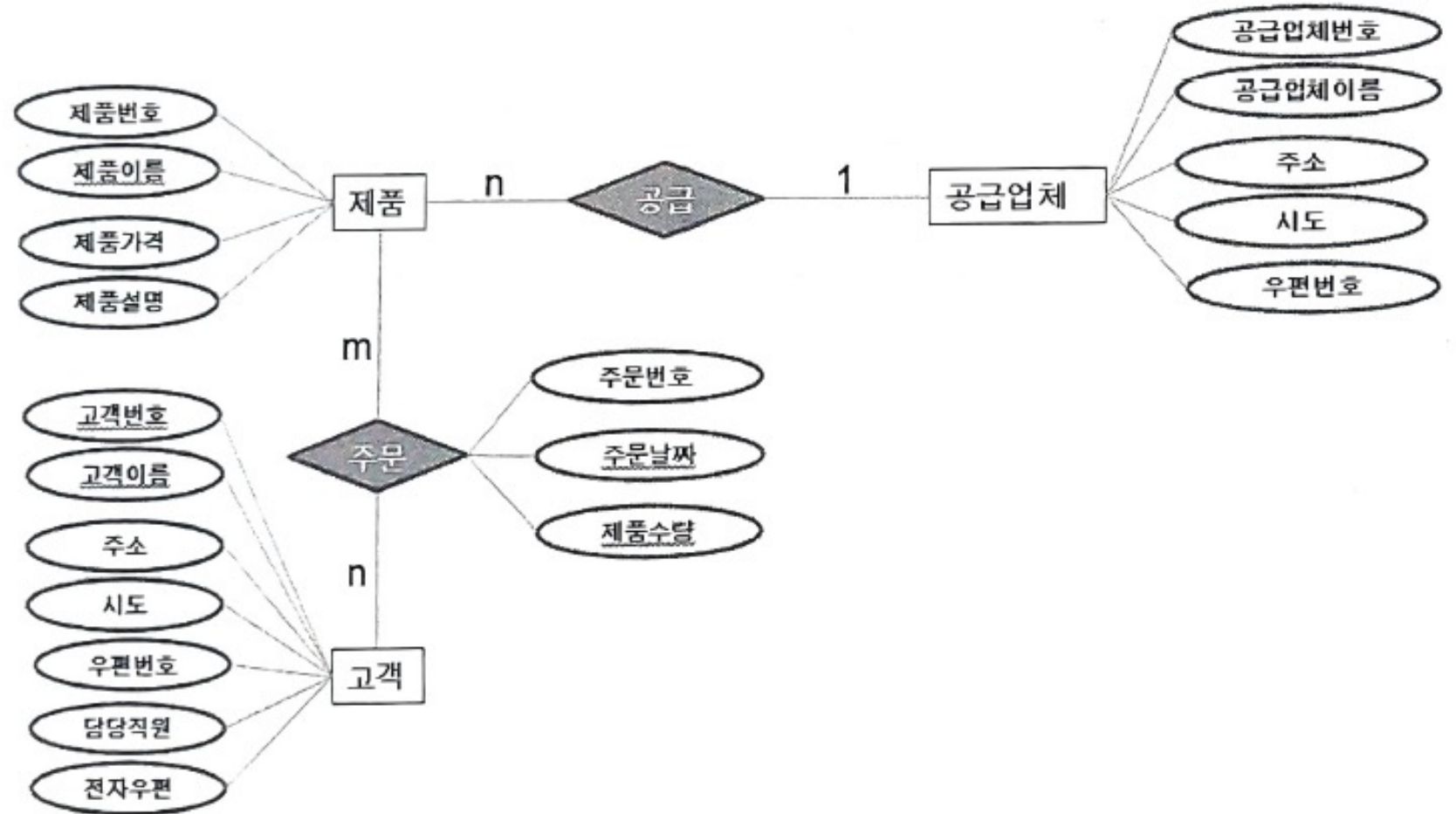
### ■ 4단계 : 물리적 설계와 구현

- 물리적 설계 단계는 논리적 구조로 표현된 데이터를 디스크 등의 물리적 저장장치에 저장할 수 있는 물리적 구조의 데이터로 변환하는 과정
- 다양한 데이터베이스 응용에 대해 처리 성능을 얻기 위해 데이터베이스 파일의 저장 구조 및 액세스 경로를 결정하는 등 데이터베이스 시스템의 성능에 중대한 영향을 미친다.
- 구현 단계는 논리적 설계 단계와 물리적 설계 단계에서 도출된 데이터베이스 스키마를 SQL로 작성한 후 명령문을 DBMS에서 실행하여 데이터베이스를 실제로 생성하는 작업

## 『장난감 가게』 데이터베이스 설계

- 장난감가게는 장난감을 판매하는 도매 업체로 공급업체로부터 장난감을 공급받아, 소매업체에 판매한다. 한 공급업체로부터 여러 제품을 납품받으며, 한번 주문에 고객에게 여러 종류의 장난감을 판매하게 된다. 이의 ER 다이어그램은 다음과 같다.

- 논리적 설계 단계를 수행해보세요
- 제출: 1114\_이름\_논리적설계.docx



## SQL(Structured Query Language)

- SQL은 관계형 데이터베이스 관리 시스템(RDBMS)의 데이터를 관리하기 위해 설계된 특수한 목적의 프로그래밍 언어
- 관계형 데이터베이스 관리 시스템에서 자료의 검색과 관리, 데이터베이스 스키마 생성과 수정. 데이터베이스 객체 접근 조정 관리를 위해 고안
- 기능별 분류
  - 데이터 정의 언어(DDL:Data Definition Language)
    - CREATE, ALTER, RENAME, DROP, TRUNCATE
  - 데이터 조작 언어(DML:Data Manipulation Language)
    - SELECT, INSERT, UPDATE, DELETE
  - 데이터 제어 언어(DCL: Data Control Language)
    - GRANT, BEGIN TRANSACTION, ROLLBACK, COMMIT

## SQL(Structured Query Language)

### ■ SELECT 문

```
SELECT <필드 이름>  
FROM <테이블 이름>;
```

```
SELECT 과목명  
FROM 과목;
```

### ☞ SQL문장에서 알아두어야 할 것

- SQL문장 안에 있는 불필요한 공백들은 실행 시 무시된다. 한 문장을 한 줄에 쓸 수 있으나, 읽기 쉽게 하기 위하여 여러 줄로 나누어 쓰는 방법이 유용하다.
- SQL 문장의 끝은 세미콜론(;)으로 끝난다..
- 일반적으로 SQL문장의 키워드는 대소문자를 구별하지 않는다. 그러나, 필드 이름과 테이블 이름은 대소문자를 구별하니 입력할 때 조심하기 바란다.

## SQL(Structured Query Language)

### ■ SELECT 문

```
SELECT 과목번호, 과목명, 학점  
FROM 과목;
```

```
SELECT *  
FROM 과목 ;
```

```
SELECT 담당교수, 과목명 학점  
FROM 과목  
ORDER BY 담당교수, 과목명;
```

```
SELECT 담당교수, 과목명 학점  
FROM 과목  
ORDER BY 1,2;
```

```
SELECT 과목명  
FROM 과목  
ORDER BY DESC;
```

```
SELECT 담당교수, 과목명, 학점  
FROM 과목  
ORDER BY 담당교수 DESC, 과목명 DESC;
```



## SQL(Structured Query Language)

### ■ SELECT 문

```
SELECT 과목명, 학점 시수  
FROM 과목  
WHERE 시수 = 4;
```

```
SELECT 과목명, 학점  
FROM 과목  
WHERE BETWEEN 2 AND 3 ;
```

```
SELECT 과목명 학점  
FROM 과목  
WHERE 학점>3;
```

```
SELECT 학번, 이름, 전자우편  
FROM 학생  
WHERE 전자우편 IS NULL ;
```

```
SELECT 과목명 학점  
FROM 과목  
WHERE 학점 >=3;
```

```
SELECT 학번, 이름, 전자우편  
FROM 학생  
WHERE 전자우편 IS NOT NULL;
```

```
SELECT 과목명, 이수구분  
FROM 과목  
WHERE 이수구분 <>'교양';
```

```
SELECT 과목명, 이수구분  
FROM 과목  
WHERE 이수구분 !='교양';
```

## SQL(Structured Query Language)

### ■ SELECT 문

```
SELECT 과목명, 담당교수, 학점  
FROM 과목  
WHERE 학점 >= 1 AND 학점 <= 2;
```

```
SELECT 과목명, 담당교수, 학점, 학기  
FROM 과목  
WHERE 학기 = '2' AND 학점 >= 3;
```

```
SELECT 과목명, 담당교수, 학점  
FROM 과목  
WHERE 학점 = 1 OR 학점=2;
```

```
SELECT 과목명, 담당교수, 학점, 학기  
FROM 과목  
WHERE 학기 = '2' OR 학점 >= 3;
```

```
SELECT 과목명, 담당교수, 학점, 학기  
FROM 과목  
WHERE (학점 = 3 OR 학점 = 4) AND 학기 = '2';
```

```
ELECT 과목명, 담당교수, 학점, 학기  
FROM 과목  
WHERE NOT 담당교수='1000001';
```

```
SELECT 과목명, 담당교수, 학점, 학기  
FROM 과목  
WHERE 담당교수 <> '1000001';
```

## SQL(Structured Query Language)

### ■ SELECT 문

```
SELECT 과목명, 담당교수, 학점, 학기  
FROM 과목  
WHERE 담당교수 IN ('1000001', '1000002');
```

```
SELECT 과목명, 담당교수, 학점, 학기  
FROM 과목  
WHERE 담당교수='1000001' OR 담당교수='1000002';
```

- 목록에 여러 값을 나열할 때, IN 연산자가 OR 연산자보다 쓰기 쉽고 이해하기 쉽다.
- 연산자 수를 줄일 수 있다.
- OR 연산자보다 실행 속도가 빠르다.
- IN 연산자에 다른 SELECT문을 넣을 수 있다.

## SQL(Structured Query Language)

### ■ SELECT 문

```
SELECT 과목명, 담당교수, 학점  
FROM 과목  
WHERE 과목명 LIKE '컴퓨터%';
```

```
SELECT 과목명, 담당교수, 학점  
FROM 과목  
WHERE 과목명 LIKE '%보호%';
```

```
SELECT 과목명, 담당교수, 학점  
FROM 과목  
WHERE 과목명 LIKE '컴퓨터%개론';
```

```
SELECT 과목명, 담당교수, 학점  
FROM 과목  
WHERE 과목명 LIKE ' _ _ _개론';
```

```
SELECT 과목명, 담당교수, 학점  
FROM 과목  
WHERE 과목명 LIKE '%개론';
```

## SQL(Structured Query Language)

### ■ SELECT 문

```
SELECT 과목명, 영문명, CONCAT(과목명, 영문명)  
FROM 과목  
ORDER BY 과목명;
```

```
SELECT 과목명, 영문명, CONCAT(과목명, '(', 영문명, ')')  
FROM 과목  
ORDER BY 과목명;
```

```
SELECT 과목명 AS 이름, CONCAT(과목명, '(', 영문명, ')') AS 국영문  
명  
FROM 과목  
ORDER BY 과목명;
```

## SQL(Structured Query Language)

### ■ SELECT 문

```
SELECT 과목명, 학점, 시수  
FROM 과목  
ORDER BY 과목명;
```

```
SELECT 과목명, 학점, 시수, 시수-학점, 시수*15 AS 총시간수  
FROM 과목  
ORDER BY 과목명;
```

```
SELECT 과목명 AS 이름, CONCAT(과목명, '(', 영문명, ')') AS 국영문  
명  
FROM 과목  
ORDER BY 과목명;
```

# SQL(Structured Query Language)

## ■ SELECT 문

```
SELECT 영문명, LOWER(영문명), UPPER(영문명)  
FROM 과목;
```

```
SELECT 과목명, LEFT(과목명,2), RIGHT(과목명,2), SUBSTRING(과목명,2,3), LENGTH(과목  
명)  
FROM 과목;
```

```
SELECT 학번, 날짜, YEAR(날짜), MONTH(날짜), DAY(날짜)  
FROM 수강신청;
```

```
SELECT 학번, 날짜  
FROM 수강신청  
WHERE MONTH(날짜)=8;
```

```
SELECT CURDATE(), CURTIME(), NOW();
```

```
SELECT DATE(NOW()), TIME(NOW());
```

# SQL(Structured Query Language)

## ■ SELECT 문

```
SELECT 학점, if(학점 >= 3, '3학점 이상', '3학점 미만')  
FROM 과목;
```

```
SELECT 과목번호, 평점, CASE 평점  
                        WHEN 1 THEN 'D'  
                        WHEN 2 THEN 'C'  
                        WHEN 3 THEN 'B'  
                        WHEN 4 THEN 'A'  
                        WHEN 0 THEN 'F'  
                        ELSE '입력오류'  
                        END AS 학점  
FROM 수강신청내역  
WHERE 수강신청번호 = '1810002';
```



## SQL(Structured Query Language)

### ■ SELECT 문

```
SELECT *  
FROM 학생;
```

```
SELECT COUNT(*) AS 학생수  
FROM 학생;
```

```
SELECT 과목번호, 평점  
FROM 수강신청내역  
WHERE 수강신청번호='1810003';
```

```
SELECT SUM(평점) AS 합계, AVG(평점) AS 평균  
FROM 수강신청내역  
WHERE 수강신청번호='1810003';
```

```
SELECT 시수  
FROM 과목 ;
```

```
SELECT MAX(시수) AS '최대 시수', MIN(수) AS '최소 시수'  
FROM 과목;
```

## SQL(Structured Query Language)

- SELECT 문

```
SELECT 담당교수  
FROM 과목 ;
```

```
SELECT DISTINCT 담당교수  
FROM 과목;
```

```
SELECT COUNT(DISTINCT 담당교수) AS '강의 교수 수'  
FROM 과목;
```

## SQL(Structured Query Language)

### ■ SELECT 문

```
SELECT 담당교수, 과목명, 학점  
FROM 과목
```

```
SELECT 담당교수  
FROM 과목  
GROUP BY 담당교수
```

```
SELECT 담당교수, 학점  
FROM 과목  
GROUP BY 담당교수, 학점
```

```
SELECT 담당교수, COUNT(*) AS 과목수, SUM(학점) AS 학점  
수  
FROM 과목  
GROUP BY 담당교수  
ORDER BY 담당교수;
```

```
SELECT 담당교수, 학점, COUNT(*) AS 과목수, SUM(학점) AS  
학점수  
FROM 과목  
GROUP BY 담당교수, 학점  
ORDER BY 담당교수, 학점
```

# SQL(Structured Query Language)

## ■ SELECT 문

```
SELECT 담당교수, COUNT(*) AS 과목수, SUM(학점) AS 학점수
FROM 과목
WHERE 학기='1'
GROUP BY 담당교수
ORDER BY 담당교수;
```

담당교수	과목수	학점수
1000001	1	2
1000002	1	3
1000003	1	2
1000004	1	3

```
SELECT 담당교수, COUNT(*) AS 과목수, SUM(학점) AS 학점수
FROM 과목
WHERE 학기='1'
GROUP BY 담당교수
HAVING 학점수>2
ORDER BY 담당교수;
```

담당교수	과목수	학점수
1000002	1	3
1000004	1	3

WHERE → GROUP BY → HAVING → ORDER BY → SELECT

## SQL(Structured Query Language)

### ■ SELECT 문

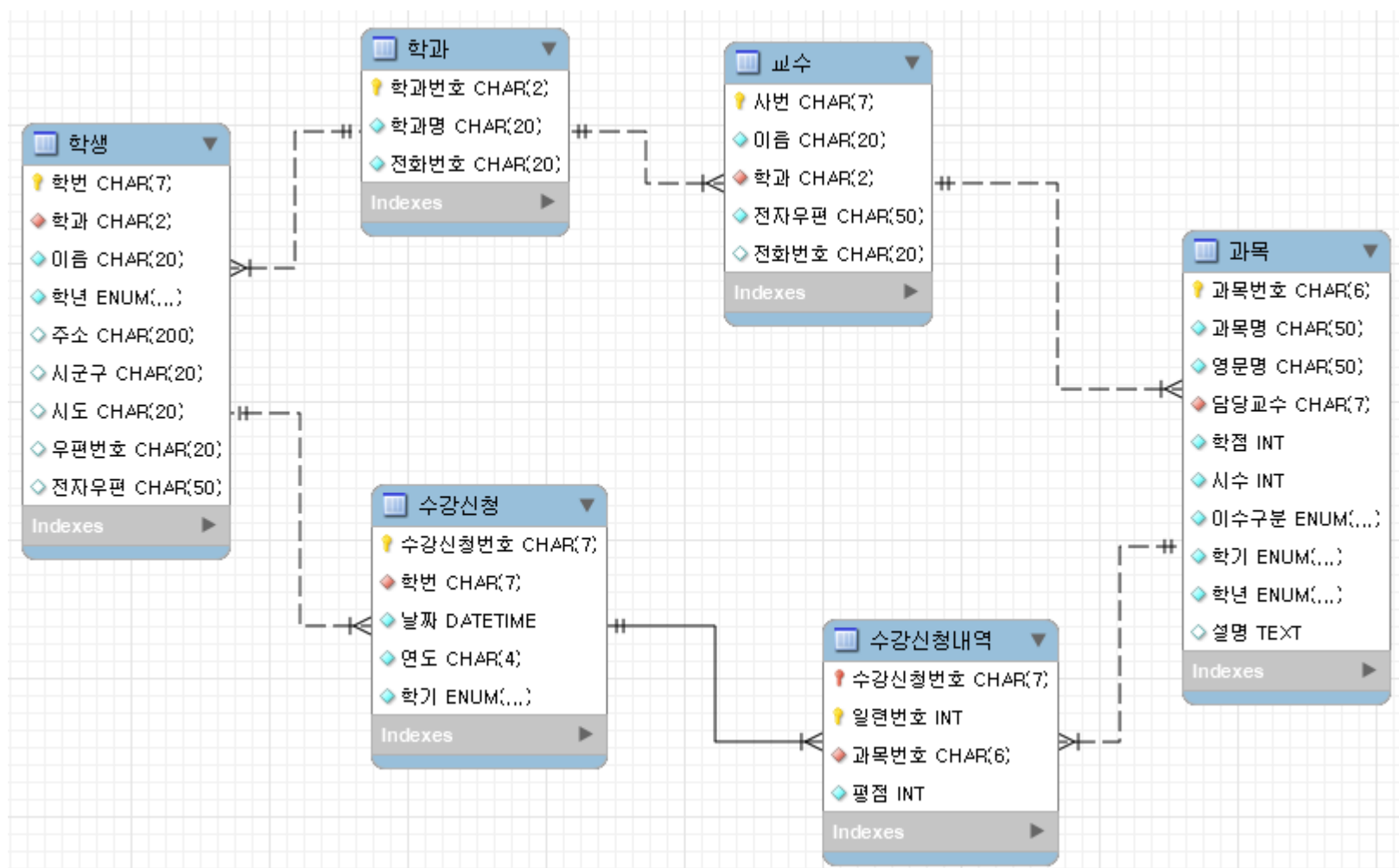
```
SELECT 담당교수, COUNT(*) AS 과목수, SUM(학점) AS 학점  
수  
FROM 과목  
GROUP BY 담당교수  
ORDER BY 담당교수;
```

```
SELECT 담당교수, COUNT(*) AS 과목수, SUM(학점) AS 학점  
수  
FROM 과목  
GROUP BY 담당교수  
HAVING COUNT(*) >= 3  
ORDER BY 담당교수;
```

담당교수	과목수	학점수
1000001	3	9
1000002	3	10
1000003	1	2
1000004	3	7

담당교수	과목수	학점수
1000001	3	9
1000002	3	10
1000004	3	7

# EERD



# SQL(Structured Query Language)

- 하위 쿼리로 두 개의 테이블 연결하기

```
SELECT 사번, 학과, 이름,  
      (SELECT 학과명  
       FROM 학과  
       WHERE 학과.학과번호=교수. 학과) AS 학과명 -- WHERE<A>.기본키=<B>.외래키  
FROM 교수;
```

학과 Table

학과번호	학과명	전화번호
01	기초키 컴퓨터정보학과	022-200-3000
02	전자공학과	022-200-4000
03	전기공학과	022-200-5000
04	정보통신공학과	022-200-6000

교수 Table

사번	이름	학과	전자우편	전화번호
1000001	김교수	01	외래키 kim@school.ac.kr	010-000-0001
1000002	이교수	01	lee@school.ac.kr	010-000-0002
1000003	박교수	01	park@school.ac.kr	010-000-0003
1000004	최교수	01	choi@school.ac.kr	
1000005	한교수	04	han@school.ac.kr	

## SQL(Structured Query Language)

- 하위 쿼리로 두 개의 테이블 연결하기

```
SELECT 사번, 학과, 이름,  
      (SELECT 학과명  
       FROM 학과  
       WHERE 학과.학과번호=교수.학과) AS 학과명,  
      (SELECT 전화번호  
       FROM 학과  
       WHERE 학과.학과번호=교수.학과) AS 학과전화번호  
FROM 교수;
```

**하나의 하위 쿼리에서는 하나의 필드 밖에 조회할 수 없으므로, 여러 필드를 조회하기 위해서는 여러 개의 하위 쿼리를 사용해야 한다.**



# SQL(Structured Query Language)

- 하위 쿼리로 두 개의 테이블 연결하기

```
SELECT 과목번호, 과목명, 담당교수, 학점,  
      (SELECT 이름  
       FROM 교수  
       WHERE 교수.사번=과목.담당교수)AS 교수명  
FROM 과목;
```

교수 Table					
사번	이름	학과	전자우편	전화번호	
1000001	김교수	01	kim@school.ac.kr	010-000-0001	
1000002	이교수	01	lee@school.ac.kr	010-000-0002	
1000003	박교수	01	park@school.ac.kr	010-000-0003	
1000004	최교수	01	choi@school.ac.kr		
1000005	한교수	04	han@school.ac.kr		

과목 Table					
과목번호	과목명	영문명	담당교수	학점	
K20002	컴퓨터네트워크	Computer Network	외래키 1000001	2	
K20012	정보보호개론	Introduction to Infomatio...	1000001	4	
K20013	인터넷프로그래밍	Internet Programming	1000001	3	
K20023	프로그래밍언어	Programming Language	1000002	4	
K20033	데이터베이스	Database	1000002	3	
K20035	운영체제	Operating System	1000004	3	
K20045	컴퓨터소프트웨어개론	Introduction to Computer ...	1000002	3	


## SQL(Structured Query Language)

- 하위 쿼리 필터링하기

- 과목번호가 'K20002'인 과목을 수강 신청한 모든 학생의 정보가 필요하다면...?


- <수강신청내역> 테이블에서 'K20002' 과목 수강신청한 '수강신청번호'를 얻는다.

```
SELECT 수강신청번호  
FROM 수강신청내역  
WHERE 과목번호='K20002';
```



- <수강신청> 테이블에서 얻어진 '수강신청번호'를 기반으로 각 수강신청을 한 학생의 '학번'을 얻는다.

```
SELECT 학번  
FROM 수강신청  
WHERE 수강신청번호 IN('1810001', '1810002', '1810003');
```



- <학생> 테이블에서 얻어진 '학번'을 기반으로 학생의 정보를 얻는다.

```
SELECT 학과, 이름, 학년  
FROM 학생  
WHERE 학번 IN('1810001', '1810002', '1810003');
```

### 실행결과

학과	이름	학년
01	김민준	1
01	박주원	1
04	윤서연	1

## SQL(Structured Query Language)

- 하위 쿼리를 계산 필드로 이용
  - <학생> 테이블에 있는 모든 학생들의 수강신청 횟수를 검색하려면...
    - <학생> 테이블에서 모든 레코드를 읽어 온다.
    - <수강신청> 테이블에서 얻어진 '학번'에 대한 수강신청 횟수(레코드 수)를 센다.

```
SELECT 학번, 이름,  
       (SELECT COUNT(*)  
        FROM 수강신청  
        WHERE 수강신청.학번=학생.학번) AS 수강신청횟수  
FROM 학생;
```

# SQL(Structured Query Language)

## ■ 조인

```
SELECT *  
FROM 교수, 학과  
WHERE 교수.학과=학과.학과번호;
```

```
SELECT 사번, 학과, 이름,  
    (SELECT 학과명  
     FROM 학과  
     WHERE 학과.학과번호=교수.학과) AS 학과명,  
    (SELECT 전화번호  
     FROM 학과  
     WHERE 학과.학과번호=교수.학과) AS 학과전화번호  
FROM 교수;
```

하위 쿼리

```
SELECT 사번, 학과, 이름, 학과명, 학과.전화번호 AS 학과전화번호  
FROM 교수, 학과  
WHERE 교수.학과 = 학과.학과번호;
```

조인 쿼리

## SQL(Structured Query Language)

- 내부조인(INNER JOIN)
  - 내부 조인은 둘 이상의 테이블에 존재하는 공통 필드(기본키와 외래키)의 값이 같은 것을 결과로 추출

```
SELECT 사번, 학과, 이름, 학과명, 학과.전화번호 AS 학과전화번호  
FROM 교수, 학과  
WHERE 교수.학과 = 학과.학과번호;
```

```
SELECT 사번, 학과, 이름, 학과명, 학과.전화번호 AS 학과전화번호  
FROM 교수 INNER JOIN 학과  
ON 교수.학과=학과.학과번호;
```

# SQL(Structured Query Language)

- 내부조인(INNER JOIN)
  - 여러 테이블 조인
    - 과목번호가 'K20002'인 과목을 수강 신청한 모든 학생의 학과,이름, 학년을 조회하는 SQL문

```
SELECT 학과, 이름, 학년
FROM 학생, 수강신청, 수강신청내역
WHERE 학생.학번=수강신청.학번AND 수강신청.수강신청번호=수강신청내역.수강신청번호AND 과목번호 = 'K20002';
```

[illegible]

## SQL(Structured Query Language)

- 내부조인(INNER JOIN)
  - 여러 테이블 조인
    - '수강신청번호'가 '1810001'(<수강신청> 테이블)인 수강신청의 신청 학생 이름(<학생> 테이블), 과목 이름, 학점(<과목> 테이블)과 이 과목의 담당교수 이름(<교수> 테이블), 평점(<수강신청내역> 테이블)를 출력하고자 한다면 ?

```
SELECT 학생.이름, 과목명, 학점, 교수.이름, 평점
FROM 수강신청, 학생, 과목, 교수, 수강신청내역
WHERE 수강신청.학번=학생.학번
AND 수강신청.수강신청번호 = 수강신청내역,수강신청번호
AND 수강신청내역과목번호 = 과목과목번호
AND 과목 담당교수 = 교수.사번
AND 수강신청.수강신청번호 = '1810001';
```

## SQL(Structured Query Language)

- 외부조인(OUTER JOIN)
  - 내부 조인은 조인 대상 테이블에서 공통인 필드 값을 기반으로 결과를 생성
    - <교수> 테이블의 모든 교수에 대하여 교수 정보와 그 교수가 담당하고 있는 교과목 이름을 알고자 한다면...?

```
SELECT 교수.사번, 교수.이름, 과목과목명  
FROM 교수 INNER JOIN 과목  
ON 교수.사번 = 과목.담당교수;
```

- 담당 과목이 없는 교수의 자료를 조회한다면...?

```
SELECT 교수.사번, 교수.이름, 과목.과목명  
FROM 교수 LEFT OUTER JOIN 과목  
ON 교수.사번 = 과목.담당교수;
```

```
SELECT 교수,사번, 교수.이름, 과목.과목명  
FROM 과목 RIGHT OUTER JOIN 교수  
ON 교수.사번 = 과목.담당교수;
```



## SQL(Structured Query Language)

- 조인에서 집계 함수 사용하기

```
SELECT 교수. 사번, 교수.이름, COUNT(과목.과목번호) AS 담당과목수  
FROM 교수 INNER JOIN 과목  
ON 교수.사번 = 과목.담당교수  
GROUP BY 교수.사번;
```

```
SELECT 교수. 사번, 교수. 이름, COUNT(과목.과목번호) AS 담당과목수  
FROM 교수 LEFT OUTER JOIN 과목  
ON 교수. 사번 = 과목. 담당교수  
GROUP BY 교수.사번;
```

## SQL(Structured Query Language)

### ■ 쿼리 결합(UNION)

- 여러 SELECT 문을 결합하여 하나의 쿼리로 만드는 것이 가능
  - 예) <교수> 테이블에 '전자우편' 필드가 있고, <학생> 테이블에도 학생들의 '전자우편'
  - 필드를 가지고 있다, 교수와 학생 모두에게 메일을 보내려 할 때, 테이블 별로 쿼리를 실행시켜 전자우편을 검색해서 그 결과를 병합해야 한다

```
SELECT 사번 AS 번호, 학과, 이름, 전자우편  
FROM 교수  
UNION  
SELECT 학번 AS 번호, 학과, 이름, 전자우편  
FROM 학생;
```

- UNION 키워드는 각각의 SELECT 문에 의한 검색 결과 중 중복되는 것은 제거
  - UNION ALL

## SQL(Structured Query Language)

- 저장 프로시저(Stored Procedure)

- 저장 프로시저는 일련의 쿼리를 마치 하나의 함수처럼 실행하기 위한 쿼리의 집합
  - 저장 프로시저는 데이터베이스에서 처리해야 하는 어떤 논리적 순서를 구성하고 그것을 하나의 명령어로 처리할 수 있게 한 것이며, 복잡한 처리 및 조회 등의 쿼리를 작성할 때 사용한다
- 저장 프로시저를 만들기 위한 SQL문

```
CREATE PROCEDURE 저장프로시저_이름(IN 매개변수, OUT 매개  
변수)  
BEGIN  
    SQL문장들  
END
```

# SQL(Structured Query Language)

- 제어문을 사용하는 프로시저
  - 저장 프로그램의 제어문은 어떤 조건에서 어떤 코드가 실행되어야 하는지를 제어하기 위한 문법으로, 절차적 언어의 구성요소를 포함함

구문	의미	문법
DELIMITER	<ul style="list-style-type: none"><li>구문 종료 기호 설정</li></ul>	DELIMITER {기호}
BEGIN-END	<ul style="list-style-type: none"><li>프로그램 문을 블록화시킴</li><li>중첩 가능</li></ul>	BEGIN {SQL 문} END
IF-ELSE	<ul style="list-style-type: none"><li>조건을 검사 결과에 따라 문장을 선택적으로 수행</li></ul>	IF <조건> THEN {SQL 문} [ELSE {SQL 문}] END IF;
LOOP	<ul style="list-style-type: none"><li>LEAVE 문을 만나기 전까지 LOOP을 반복</li></ul>	[label:] LOOP {SQL 문   LEAVE [label]} END LOOP
WHILE	<ul style="list-style-type: none"><li>조건이 참일 경우 WHILE 문의 블록을 실행</li></ul>	WHILE <조건> DO {SQL 문   BREAK   CONTINUE} END WHILE
REPEAT	<ul style="list-style-type: none"><li>조건이 참일 경우 REPEAT 문의 블록을 실행</li></ul>	[label:] REPEAT {SQL 문   BREAK   CONTINUE} UNTIL <조건> END REPEAT [label:]
RETURN	<ul style="list-style-type: none"><li>프로시저를 종료</li><li>상태값을 반환 가능</li></ul>	RETURN [<식>]

# SQL(Structured Query Language)

- 저장 프로시저(Stored Procedure)
- DB없는 저장 프로시저

```
/* 저장 프로시저 dorepeat가 있으면 삭제 */
drop procedure if exists dorepeat;
/* 마침기호 변경 */
delimiter //
CREATE PROCEDURE dorepeat(IN p1 INT, OUT y INT)
BEGIN
    declare x int;
    SET x=1;
    REPEAT
        SET x=x+1;
        SET y=x;
    UNTIL x>=p1
    END REPEAT;
END //
/* 마침기호 취소 */
delimiter;
```



## SQL(Structured Query Language)

- 저장 프로시저(Stored Procedure)
- DB있는 저장 프로시저

```
drop procedure if exists 새수강신청;
delimiter //
CREATE PROCEDURE 새수강신청(IN 학번 CHAR(7), OUT 수강신청_번호 INT)
BEGIN
    SELECT MAX(수강신청번호) INTO 수강신청_번호 FROM 수강신청;
    SET 수강신청_번호=수강신청_번호+1;
    INSERT INTO 수강신청(수강신청번호,학번,날짜,연도, 학기)
    VALUES(수강신청_번호, 학번, CURDATE(), '2020', '2');

END//
delimiter;
```

```
CALL 새수강신청('1804004', @수강신청_번호);
SELECT @수강신청_번호;
```

## SQL(Structured Query Language)

- 저장 프로시저(Stored Procedure) 실습
  - <학과> 테이블에 새로운 레코드를 삽입하고 삽입한 레코드를 보여주는 '새학과' stored procedure를 만드시오. 아래 실행 결과는 이 프로시저를 이용하여 출력한 예이다.

```
CALL 새학과('08', '컴퓨터보안학과', '022-200-7000');
```

실행결과	학과번호	학과명	전화번호
	08	컴퓨터보안학과	022-200-7000

## SQL(Structured Query Language)

- 저장 프로시저(Stored Procedure) 실습
  - "수강신청" 데이터베이스의 총 학생 수, 교수 수, 과목 수를 계산하는 "통계" stored procedure를 만드시오. 아래 실행 결과는 이 프로시저를 이용하여 출력한 예이다.

```
CALL 통계(@a, @b, @c);
```

```
SELECT @a AS 학생수, @b AS 교수수, @c AS 과목수;
```

실행결과	학생수	교수수	과목수
	10	7	10