제 1장 데이터와 데이터베이스

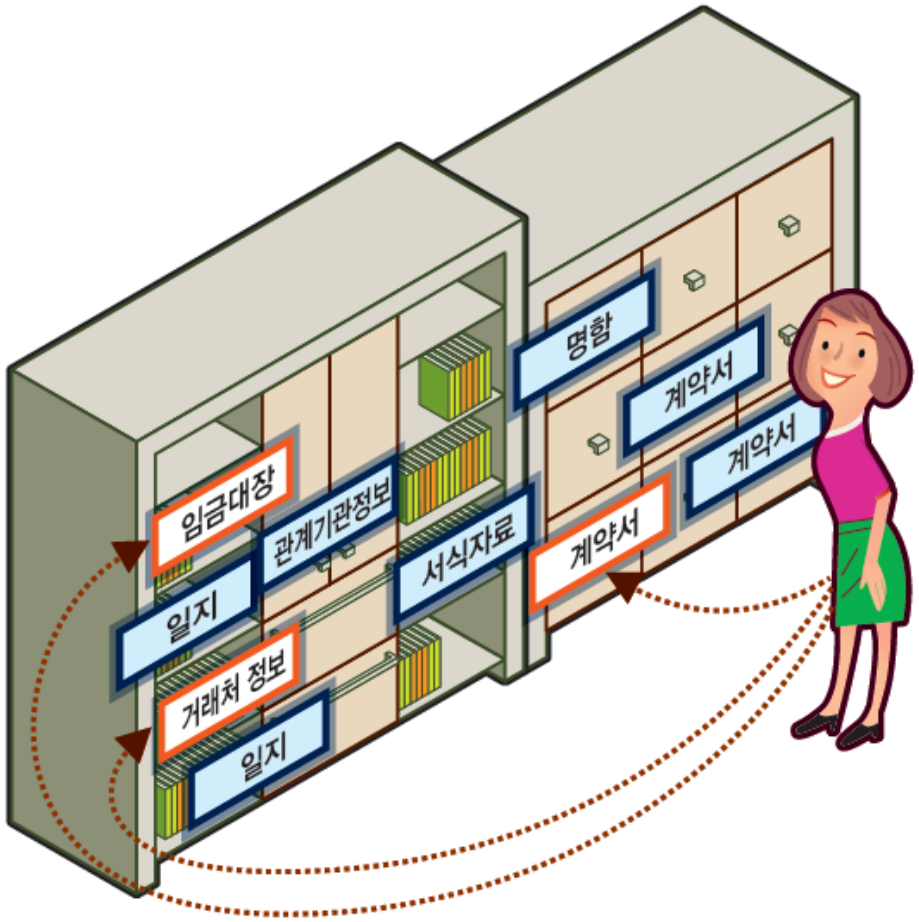
- 데이터
- 데이터베이스
- 정보 & 데이터

우리 생활 주변의 데이터베이스

- 데이터베이스 (database)
  - 정보를 필요에 따라 모아놓은 것
  - 조직이나 개인이 사용하는 조작 가능한, 저장된 데이터의 모임

  - 예) 사장실의 파일 캐비닛
    - 주소록, 계약서 등을 관리(각각 하나의 데이터베이스를 구성)
    - 편리한 사용을 위해 정렬, 분류
      - 파일 삽입, 삭제, 검색, 갱신 등
    - 컴퓨터를 이용하여 이러한 작업을 대행할 수 있음

사장실의 파일 캐비닛



데이터, 정보, 데이터베이스(1)

- 데이터, 정보, 그리고 지식
  - 데이터 (data)
    - 실세계의 실체를 묘사하는 값
    - 정형화되고 기록할 만한 가치가 있다고 판단되는 어떤 현상이나 사건, 아이디어에 대한 묘사

이름 : 홍길동

키 : 170 ㎝

몸무게 : 70 ㎏

결혼 : 미혼

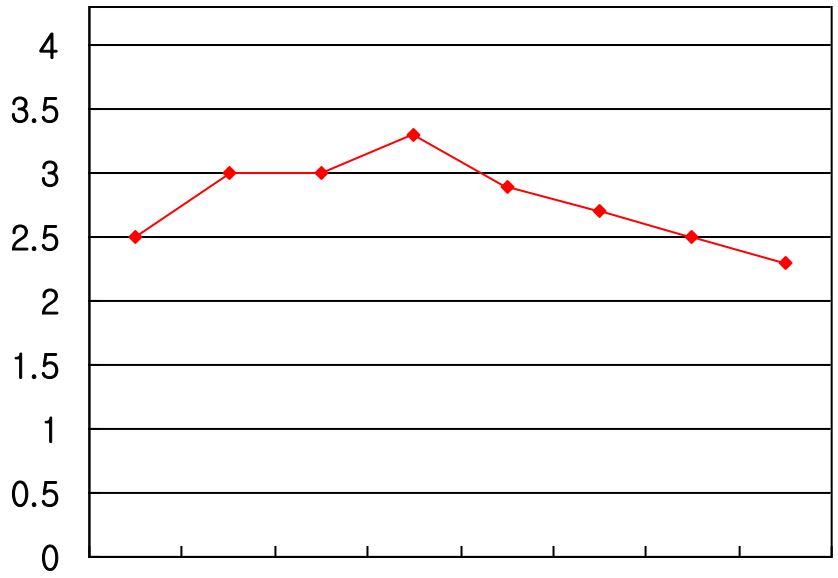
특기 : 무술

데이터, 정보, 데이터베이스(2)

- 정보 (information)
  - 데이터는 사실들 그 자체에 대한 일차적인 표현
  - 사실들과 이들로부터 유도될 수 있는 유추된 사실들

1학기	2.5
2학기	3.0
3학기	3.0
4학기	3.3
5학기	2.9
6학기	2.7
7학기	2.5
8학기	2.3

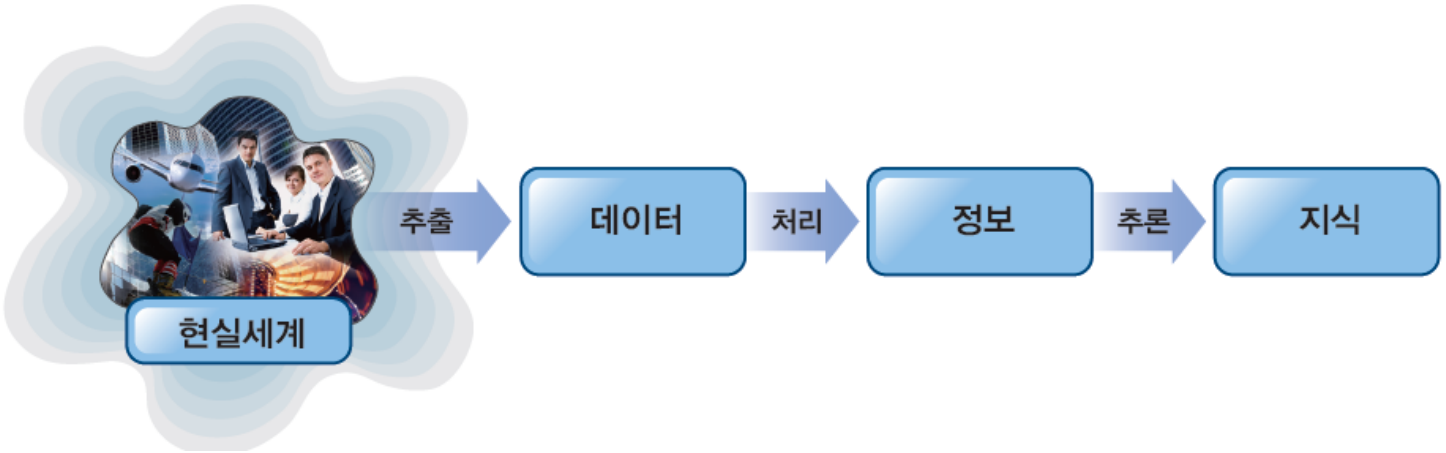
데이터



정보

데이터, 정보, 데이터베이스(3)

- 지식(knowledge)
  - 데이터와 정보에 비해 좀 더 상위 수준의 개념
  - 수동적이고 정적인 데이터나 정보에 비해, 이들을 처리하는 "**방법**"이나 어떤 근거에 의한 판단을 내리는데 필요한 분석과 판단에 관한 "**법칙**"등을 포함



데이터, 정보, 데이터베이스(4)

- 데이터베이스의 정의
  - 관련된 데이터의 모임 또는 집합
  - 정형화되고 조작 가능한 (처리하기에 용이한) 컴퓨터에 저장된 데이터의 모임
  - 특정 목적을 위해 계산, 저장, 검색, 정렬 등의 "데이터 처리" 작업을 수행
    - 예) 성적 처리, 마케팅을 위한 상품 판매 분석 등
  - 데이터베이스란 어떤 특정 조직의 응용 시스템에 사용되는 조작 가능한 저장 데이터의 모습

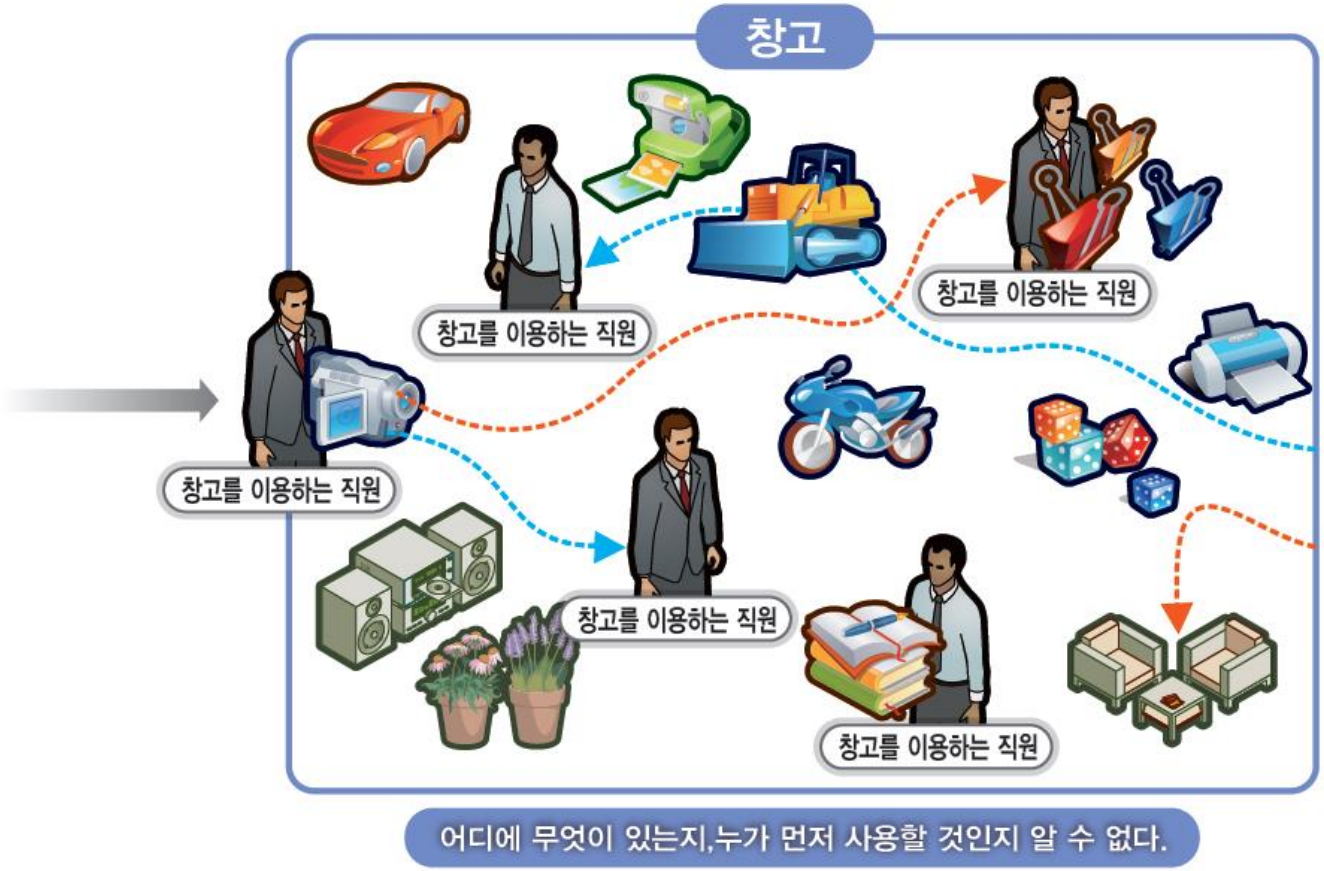
데이터, 정보, 데이터베이스(5)

- 일시적 데이터와 영구적 데이터
  - 일시적 (transient) 데이터
    - 해당 프로세스가 실행되는 동안만 일시적으로 존재
    - 예) 프로그램의 변수
  - 영구적 (persistent) 데이터
    - 어떤 프로세스의 생명주기에 종속적이지 않고 스스로 존재
    - 비휘발성 매체에 저장
  - 일반적인 데이터베이스는 지속적인 데이터의 모임을 뜻함

데이터베이스 관리 시스템(1)

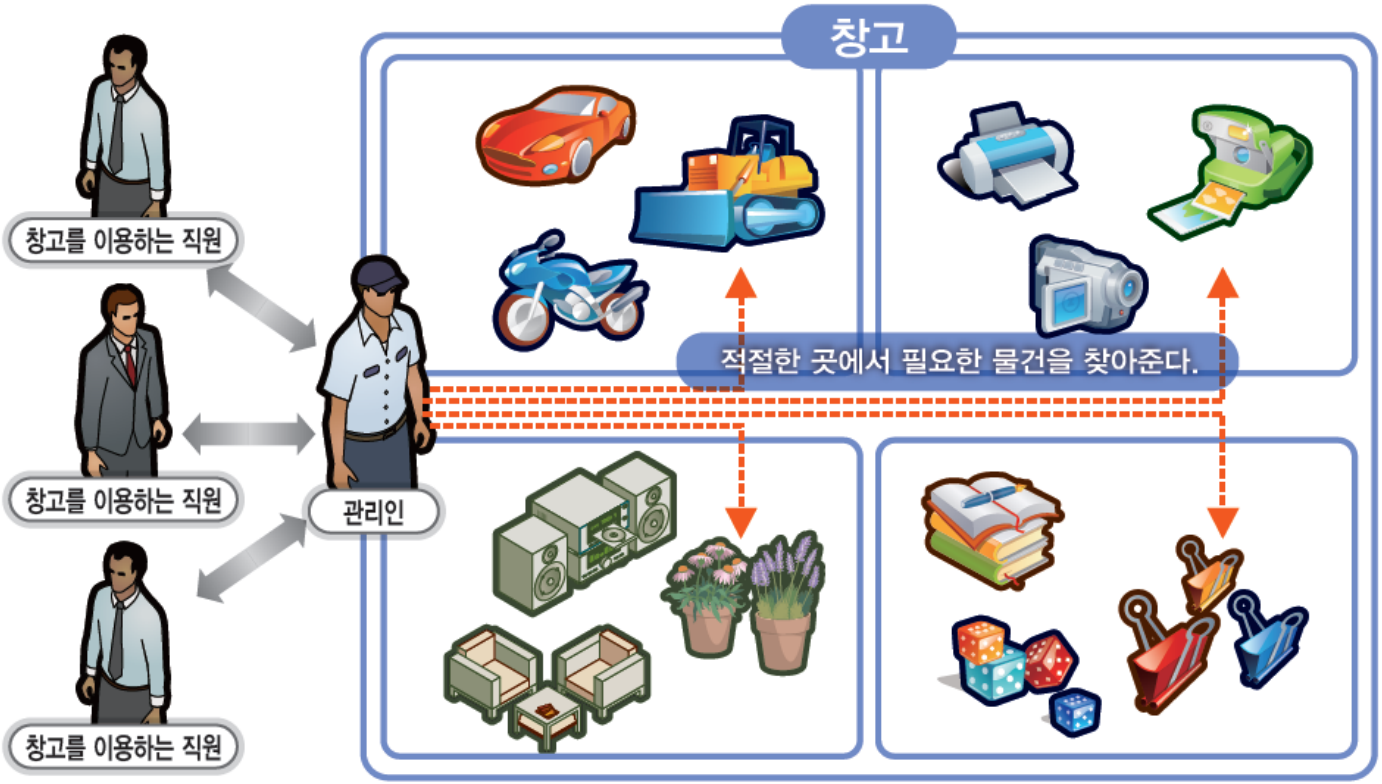
- 데이터베이스 관리 시스템
  - DBMS, DataBase Management System
    - 컴퓨터에 저장되는 데이터베이스를 관리해주는 소프트웨어 시스템
  - DBMS 종류들
    - 외산:
      - Oracle, MS SQL-Server, DB2, Sybase, dBase, FoxPro, MS Access
    - 국산
      - 큐브리드(CUBRID), 티베로(Tibero) – 티맥스소프트, ALTIBASE
    - 공개 S/W
      - MySQL, PostgresSQL, (큐브리드)

데이터베이스 관리 시스템(2)



<혼란스러운 창고>

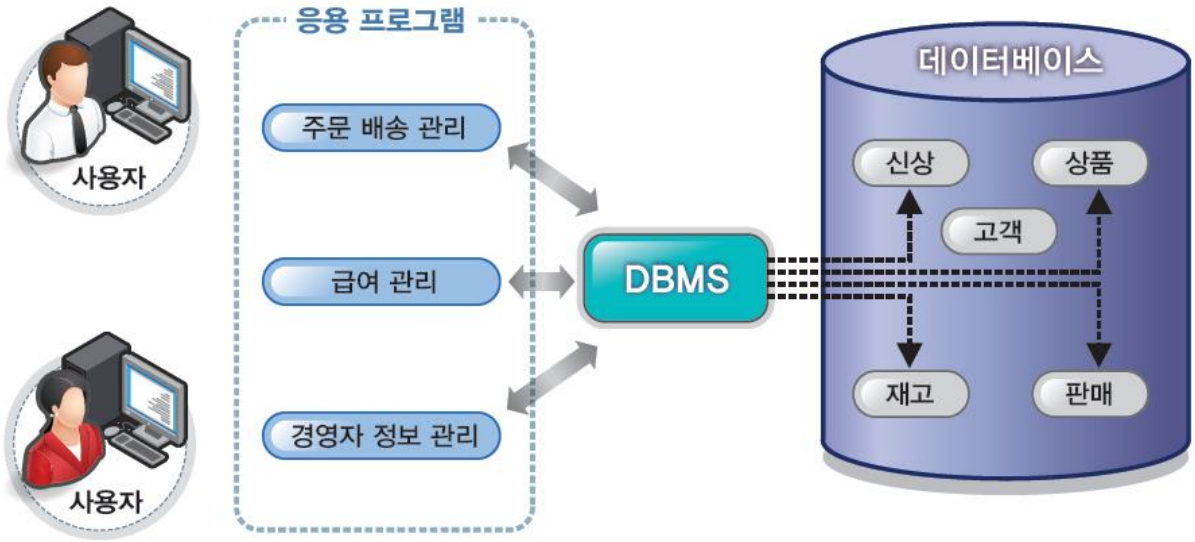
데이터베이스 관리 시스템(3)



<잘 정리된 창고>

데이터베이스 관리 시스템(3)

- 창고와 DBMS의 관계



물건	데이터
창고	데이터베이스(디스크)
창고관리인	DBMS
직원	응용 프로그램 또는 사용자

DBMS가 제공하는 기능(1)

- 정보를 표현할 수 있는 틀
  - DBMS가 정보를 관리할 수 있는 양식
  - 현실 세계의 정보를 컴퓨터에 저장시키는 양식이므로, 다양한 정보의 표현이 가능해야 함

- 데이터의 공유기능
  - DBMS가 관리하는 데이터는 여러 응용프로그램이 필요에 따라 이용할 수 있도록 제공됨

  → 데이터 중복의 제거 : 데이터를 공유함으로써 비효율성과 일관성(consistency) 문제를 제거
  - 동시성 문제(concurrency problem)
    - 다른 프로그램 또는 프로세스가 동시에 같은 데이터에 작업을 하려할 때

    → DBMS는 하나의 단위 프로그램이 일을 마칠 때까지 해당 데이터를 독점하도록 하는 방법 등으로 문제를 막음

DBMS가 제공하는 기능(2)

- 데이터 무결성 유지 기능
  - 데이터 무결성(無缺性; integrity)
    - 데이터베이스 내의 데이터가 얼마나 정확한가를 뜻함
      - 나이가 200 또는 –23 ?
      - 데이터 중복으로 인한 불일치 문제

- 데이터 독립성
  - 응용 프로그램과 데이터 간의 독립성
    - 응용 프로그램은 데이터가 디스크에 구체적으로 어떻게 저장되어 있는 지 몰라도 됨
    - 데이터에 종속적(data-dependent)
      - 데이터의 구조와 저장형태를 고려한 응용프로그램 구현으로 나중에 데이터의 저장형태나 구조를 바꾸려 할 때 응용프로그램도 바꾸지 않으면 안됨

DBMS가 제공하는 기능(3)

- 효율적인 자원관리 기능
  - 많은 양의 데이터를 다루는데 적합한 효율적이고 효과적인 방법들을 사용
  - 디스크 상에 데이터를 배치시키거나 디스크의 데이터를 처리를 위해 주기억장치로 불러들이는 작업

- 데이터 보안성과 안정성 유지 기능
  - DBMS가 관리하는 모든 데이터에 대해 자체적인 보안 기능 제공
  - 보안성 : 사람으로부터 데이터의 보호
  - 안정성 : 컴퓨터 장애나 고장 등으로부터의 보호
    - 예기치 못한 상황이 발생한 때 체계적인 수습이 가능해야 함

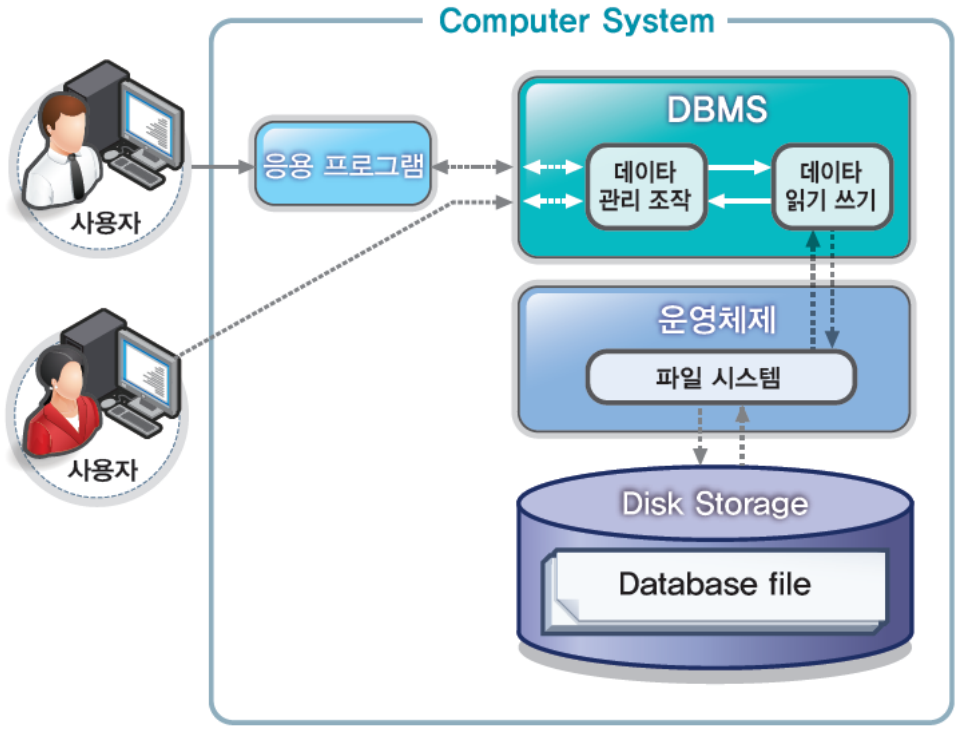
파일과 데이터베이스(1)

- 파일 시스템의 데이터 관리 기능
  - 파일 시스템 (file system)
    - 운영체제의 중요한 부분으로 데이터나 프로그램을 디스크에 읽고 쓸 수 있도록 해주는 프로그램
  - 프로그램에서 다루는 데이터를 직접 하나의 파일에 저장하여 관리하는 경우의 문제점
    - 프로그램 이외의 방법으로도 데이터 조작 가능
    - 프로그램과 데이터 형식이 묶여 있음
    - 동시 접근의 문제
    - 보안 문제
    - 장애 복구 문제

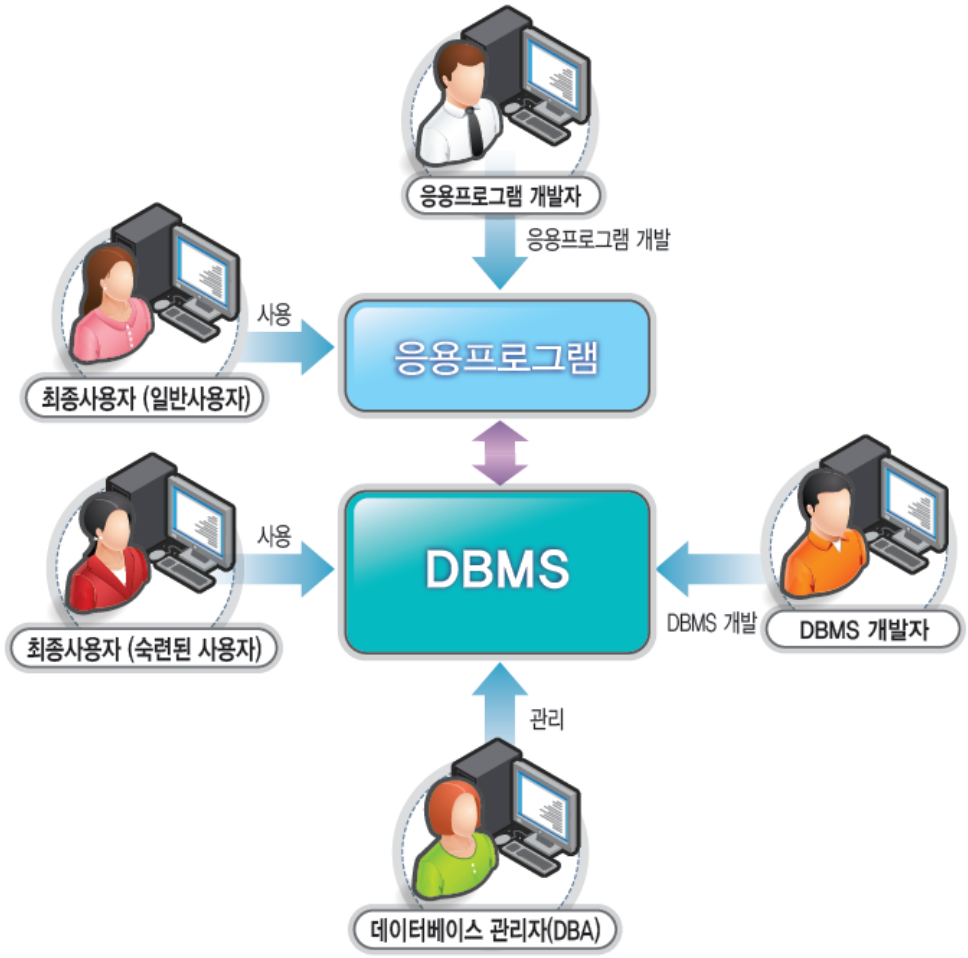
  ⇒ DBMS는 데이터베이스를 다루는 데 있어서 필요한 공통의 기능들을 제공하는 소프트웨어 시스템임

파일과 데이터베이스(2)

- 파일 시스템을 이용하는 DBMS
  - 파일 시스템 - 기본적인 저장 기능만을 제공
  - DBMS - 데이터베이스 관리에 필요한 다양한 기능



데이터베이스 시스템과 사용자



데이터베이스 시스템과 사용자(2)

- 최종사용자
  - end user, general user
  - 응용 프로그램이 제공하는 사용자 인터페이스(user interface)를 이용하며, 해당 응용분야의 업무를 처리하는 사람
  - 숙련된 최종 사용자는 응용프로그램을 이용하지 않고 DBMS에서 작접 사용
- 응용 프로그램 개발자
  - application programmer
  - DBMS를 이용한 응용프로그램을 개발하는 사람
- 데이터베이스 관리자
  - DataBase Administrator : DBA
  - DBMS 및 이와 관련된 하드웨어 또는 소프트웨어를 중앙에서 관리 감독하는 사람
- DBMS 개발자
  - DBMS developer
  - DBMS를 구성하는 모듈들을 설계하고 구현하는 사람

제 2장 관계형 데이터베이스

- 관계형 데이터 모델
- 관계형 데이터베이스
- 관계 대수

데이터 모델이란?

- 물리적 혹은 추상적으로 존재하는 현실세계를 단순화되고 정형화된 형태로 표현하는 하나의 방식 또는 규범

(a) 자전거 사진　　(b) 선으로 단순하게 그린 자전거

자전거의 추상화

- 실제 데이터가 갖는 특성을 살리면서, 목적에 맞게 관심있는 정보만을 단순화 하여 표현하는 방식
  - 데이터에 대한 조작이 가능해야함

릴레이션(relation)의 개념(1)

- 관계형 데이터 모델(relational data model)
  - 테이블 형식을 이용하여 데이터들을 정의하고 설명한 모델
  - 실세계의 데이터를 누구나 직관적으로 이해할 수 있는 형태로 기술할 수 있는 간단한 방식을 제공
  - 테이블을 릴레이션(relation)이라 부름

이 름	전화번호	주소	생일
홍길동	010-1234-5678	서울	3월 15일
이건우	010-2132-2345	서울	8월 23일
이몽룡	010-3245-4368	부산	12월 14일

표 형식으로 기술된 주소록

릴레이션의 개념(2)

- 릴레이션(relation)
  - 수학적으로, 두 개 이상의 집합으로부터 각 집합을 구성하는 원소들의 순서쌍에 대한 집합을 의미

이름 = {홍길동, 김광식, 박철수, 최용만}

주소 = {서울, 대전, 대구, 부산}

⇒ 순서쌍 : {<홍길동, 서울>, <김광식, 대전>, <박철수, 서울>, <최용만, 부산>}

이 름	주소
홍길동	서울
김광식	대구
박철수	서울
최용만	광주

순서쌍을 테이블로 표현한 예

릴레이션의 개념(3)

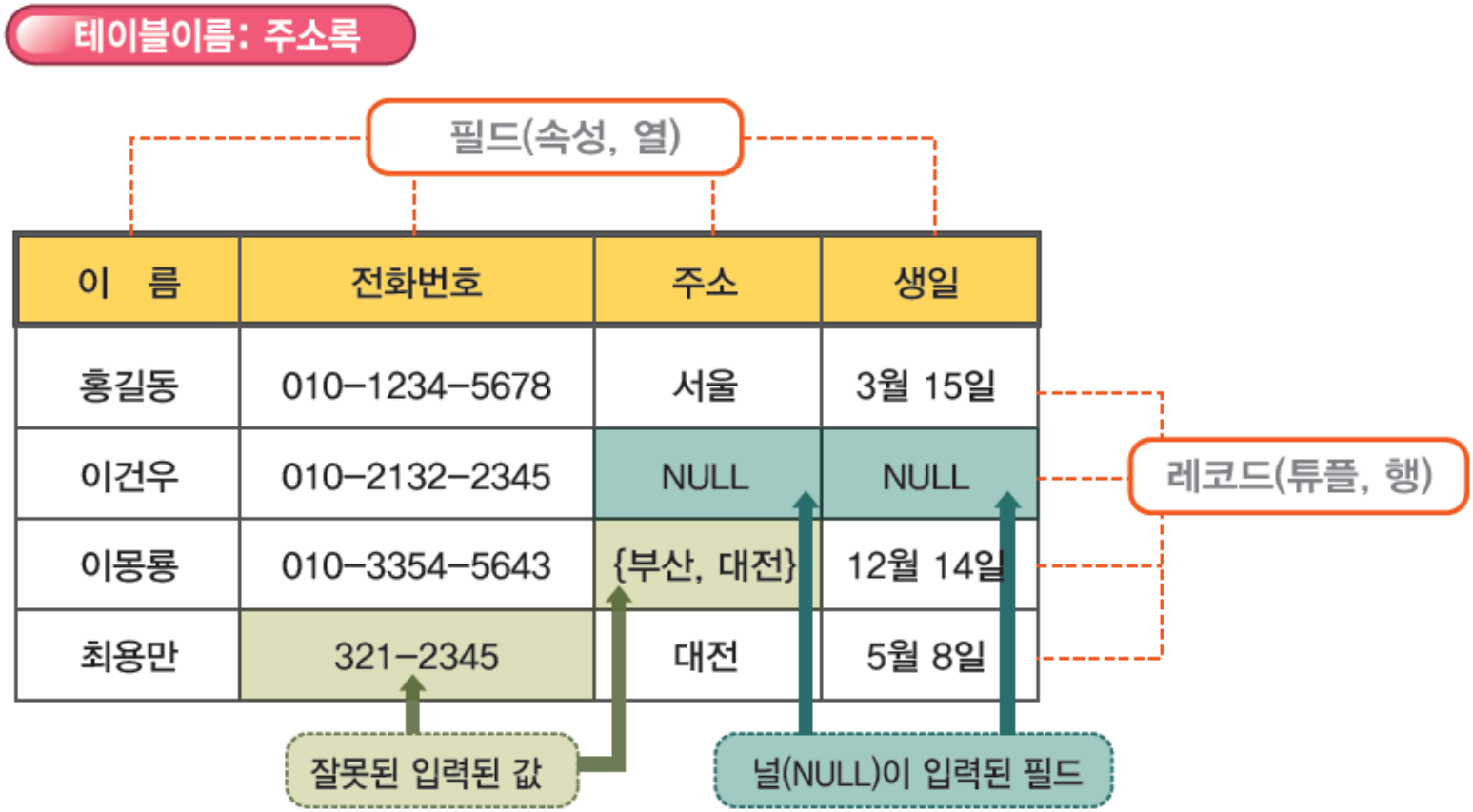
- 속성(attribute) – 필드, 컬럼
  - 릴레이션을 구성하는 각 열(column)의 이름
  - 예) 주소록 릴레이션을 구성하는 속성
    - 이름, 전화번호, 주소, 생일
- 튜플(tuple) – 레코드, 행
  - 릴레이션의 각 행
  - 예) 주소록 릴레이션의 한 튜플
    - <홍길동, 880-1234, 서울, 3월 15일>
- 이 책에서는 테이블, 필드, 레코드란 용어를 사용함

릴레이션	테이블
속성	필드(field), 컬럼(column)
튜플	레코드(record), 행(row)

릴레이션의 개념(4)

- 도메인(domain)
  - 각 필드에 입력 가능한 값들의 범위, 즉 각 필드가 가질 수 있는 모든 값들의 집합
  - 원자값(atomic value, 더 이상 분리되지 않는 값)이어야 함
  - 예) 주소록의 도메인
    - 이름: 개인 이름들로 구성된 문자열 집합
    - 전화번호: "ddd-ddd-dddd"의 형식으로 구성된 문자열의 집합 (d는 0부터9까지의 숫자)
    - 주소: 도시를 나타내는 문자열의 집합
    - 생일: "dd월dd일"로 구성된 문자열의 집합
- 널(null)
  - 특정 필드에 대한 값을 알지 못하거나 아직 정해지지 않아 입력하지 못한 경우의 필드의 값
  - 0이나 공백 문자와는 다름

릴레이션의 개념(5)


테이블이름: 주소록
필드(속성, 열)
이 름
전화번호
주소
생일
홍길동
010-1234-5678
서울
3월 15일
이건우
010-2132-2345
NULL
NULL
이몽룡
010-3354-5643
{부산, 대전}
12월 14일
최용만
321-2345
대전
5월 8일
레코드(튜플, 행)
잘못된 입력된 값
널(NULL)이 입력된 필드

테이블 스키마와 테이블 인스턴스

- 테이블 스키마(table schema, 스키마)
  - 테이블 정의에 따라 만들어진 데이터 구조
    - $R(A_1, A_2, \ldots, A_n)$

      R : 테이블의 이름

      $A_1, A_2, \ldots, A_n$: 필드들의 이름
    - 예)
      - 신입생(학번, 주민등록번호, 이름, 주소, 학과명)
- 차수(degree)
  - 테이블 스키마에 정의된 필드의 수
    - 차수 = 1 : 단항 테이블(unary relation)
    - 차수 = 2 : 이항 테이블(binary relation)
    - 차수 = n : n 항 테이블(n-ary relation)

테이블 스키마와 테이블 인스턴스(2)

- 테이블 인스턴스(table instance, 인스턴스)
  - 테이블 스키마에 현실 세계의 데이터를 레코드로 저장한 형태
  - 스키마는 한번 정의하면 거의 변함이 없지만 인스턴스는 수시로 바뀔 수 있음 – 레코드의 삽입, 삭제, 수정 등
- 기수(cardinality)
  - 테이블 인스턴스의 레코드의 수

학번	주민등록번호	이름	주소	학과명
1292001	900424-1825409	김광식	서울	컴퓨터공학과
1292002	900305-1730021	김정현	서울	컴퓨터공학과
1292003	891021-2308302	김현정	대전	컴퓨터공학과
1292301	890902-2704012	김현정	대구	산업공학과
1292303	910715-1524390	박광수	광주	산업공학과
1292305	921011-1809003	김우주	부산	산업공학과
1292501	900825-1506390	박철수	대전	전자공학과
1292502	911011-1809003	백태성	서울	전자공학과

예)신입생 테이블의 인스턴스

테이블의 특성

- 중복된 레코드가 존재하지 않음
  - 테이블 인스턴스는 레코드들의 “집합”임
- 레코드간의 순서는 의미가 없음
  - 테이블 인스턴스는 레코드들의 “집합”임
  - ‘첫번째 레코드’,‘두번째 레코드’란 표현은 의미 없음
- 레코드 내에서 필드의 순서는 의미가 없음
  - 테이블 스키마는 필드들의 집합으로 표현됨
  - ‘첫번째 필드’,‘두번째 필드’란 표현은 의미 없음
- 모든 필드는 원자값을 가짐

키(key)

- 키는 왜 필요한가?
  - 레코드간의 순서가 의미가 없으므로 레코드를 구분하기 위해서는 각 레코드의 값을 이용함
  - 키(key)
    - 필드들의 일부로 각 레코드들을 유일하게 식별해낼 수 있는 식별자(identifier)
    - 일반적으로 하나의 필드를 지정하여 키로 지정하나, 여러 개의 필드들로 키를 구성할 수 도 있음
      - 두 개 이상의 레코드로 구성된 키를 복합키(composite key)라고 함
    - 예를 들어 신입생 테이블의 학번 또는 주민등록번호 필드는 각 레코드간에 유일하므로 키가 될 수 있음
      - 그러나 학과명은 키가 될 수 없음
  - 관계형 데이터 모델에서 특정 레코드를 구별하거나 탐색하기 위한 유일한 방법

수퍼키, 후보키, 기본키의 개념

- 수퍼키(super key)
  - 아무런 제약 조건 없이 레코드들을 식별할 수 있는 필드의 집합
  - 예) (주민등록번호) (학번, 주민등록번호) (주민등록번호, 이름) (이름, 주소) 등
- 후보키(candidate key)
  - 최소한의 필드만으로 구성된 키
  - 예) (학번) (주민등록번호)(이름, 주소)(이름, 학과명)
- 기본키(primary key)
  - 후보키 중에서 식별자로 정의한 하나의 키
  - 되도록 하나의 필드로 구성된 후보키를 선정하는 것이 유리함
  - 예) (학번)
  - 그렇다면, (주민등록번호)(이름, 주소)(이름, 학과명) 모두 후보키 자격이 있는가?
    - 모든 가능한 인스턴스에 대해서도 키가 될 수 있어야 함

키가 널(null)이 될 수 있나?

- 기본키는 식별자의 기능을 함
- 기본키로 정의된 필드가 널을 갖게 되면 이러한 식별 기능을 상실
  - 예를 들어 두 개의 레코드에 대한 기본키 값이 동시에 널이면 그들은 서로 구별할 수 없음
  - 따라서 기본키는 널이 될 수 없음

외래 키(foreign key)(2)

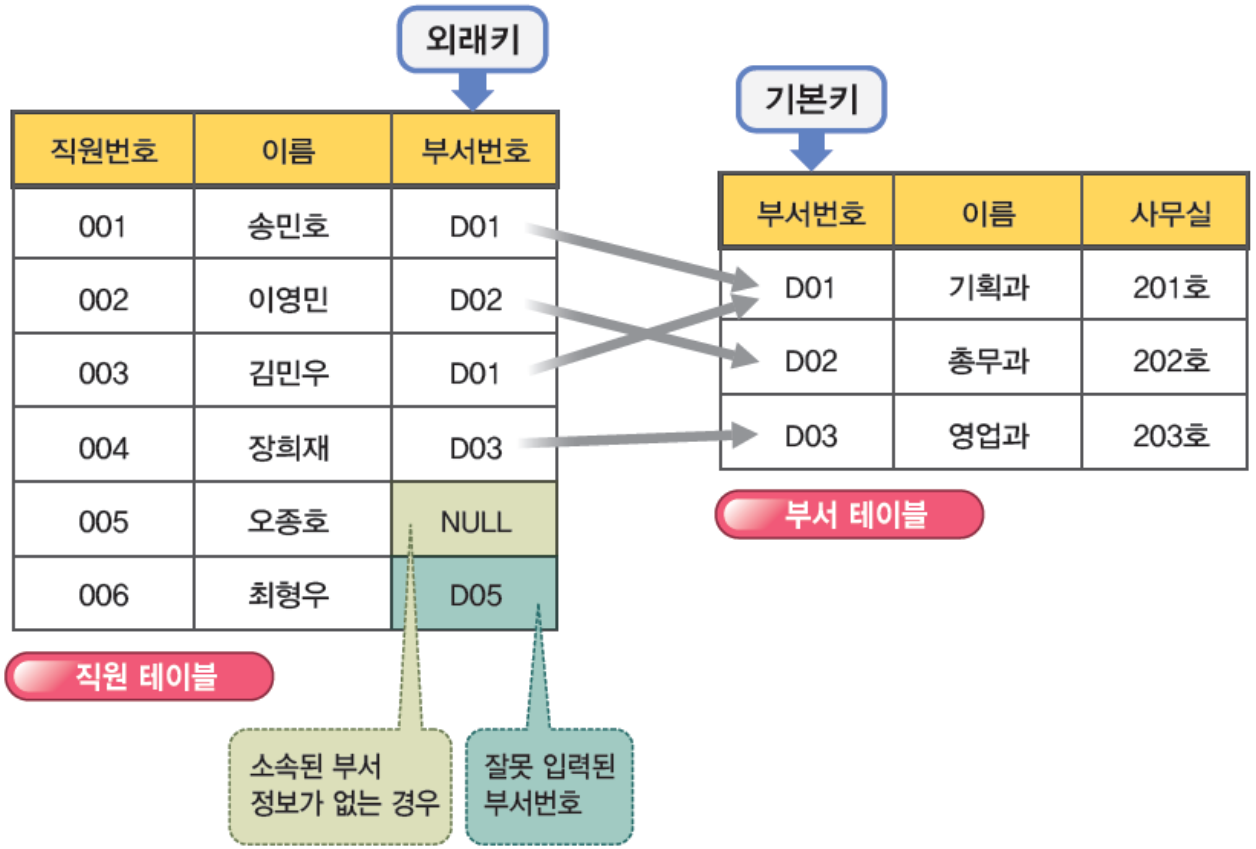
- 다른 테이블의 기본 키를 참조하는 필드집합
- 두 테이블 스키마 R_1, R_2에 대해,
  - R_1의 어떤 필드집합 FK가 다음 두 조건을 만족하면, FK는 R_2 의 기본키인 PK를 참조하는 R_1의 외래키임
    - FK의 필드들은 테이블 스키마 R_2 의 기본 키 PK와 동일한 도메인을 가짐
    - R_1의 각 레코드의 FK값은 R_2 의 레코드 중 하나의 PK값과 일치하거나 널이 됨
  - 여기서 R_1 레코드의 FK값이 널이 된다는 것은 알지 못하거나 아직 결정되지 않았다는 것을 의미함

이때, R_1 : 참조하는 테이블(referencing table)

R_2 : 참조되는 테이블(referenced table)

외래 키(foreign key)(3)

직원(<u>직원번호</u>, 이름, 부서번호)
부서(<u>부서번호</u>, 부서명, 사무실)



관계형 데이터베이스(relational database)

- 정의
  - 관계형 데이터 모델에 기반하여 하나 이상의 테이블로 실세계를 표현한 데이터베이스
    - 실세계를 관계형 데이터 모델이라는 추상적인 도구를 이용하여 표현한 것
    - 테이블들을 컴퓨터의 기억 장치에 어떠한 방법으로 저장할 것인가에 대한 물리적인 구조까지 정의한 것은 아님
- 관계형 데이터베이스가 하나 이상의 테이블로 구성되어 있을 때
  - 데이터베이스 스키마(database schema)
    : 테이블 스키마의 집합
  - 데이터베이스 인스턴스(database instance)
    : 테이블 스키마들에 대한 테이블 인스턴스의 집합

예제: 학사 데이터베이스 (2)

stu_id	resident_id	name	year	dept_id
1292001	900424-1825409	김광식	2	920
1292002	900305-1730021	김정현	2	920
1292003	891021-2308302	김현정	2	920
1292301	890902-2704012	김현정	2	923
1292303	910715-1524390	박광수	1	923
1292305	921011-1809003	김우주	2	923
1292501	900825-1506390	박철수	1	925
1292502	911011-1809003	백태성	2	925

(a) student

dept_id	dept_name	office
920	컴퓨터공학과	201호
923	산업공학과	207호
925	전자공학과	308호

(b) department

3장. 데이터모델링의 주요 개념

- ❑ 개요
- ❑ 엔티티
- ❑ 속성
- ❑ 관계
- ❑ 주식별자와 외래 식별자
- ❑ ERD 표기법

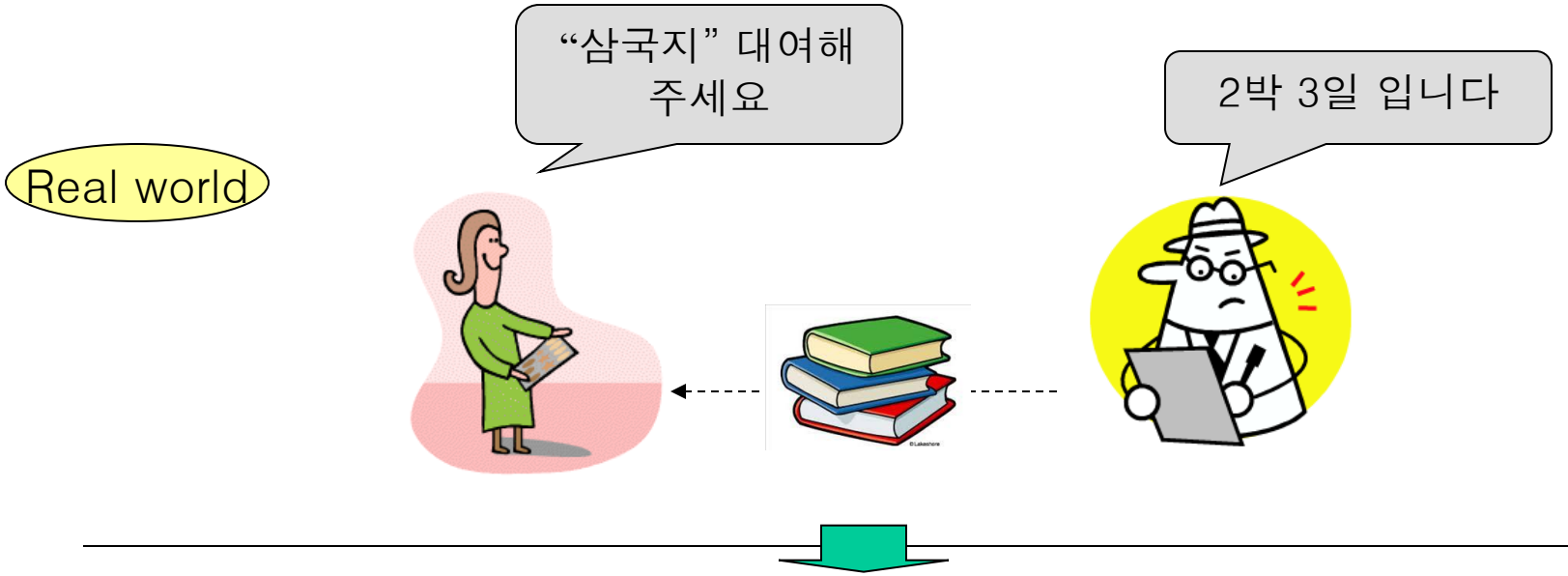
3.1 개요

❑ 데이터 모델링의 목적

- 정보 시스템을 구축하는데 필요한 정보(데이터)를 약속된 표기법에 의해 표현함으로써 시스템 구축 대상이 되는 업무 내용을 정확하게 분석하고자 함
- 분석된 모델을 가지고 실제 데이터베이스를 생성하여 개발및 관리에 이용하고자 함

<그림 3.1> 데이터 모델링의 개념

3.1 개요


Real world
"삼국지" 대여해 주세요
2박 3일 입니다
Computer world
고객
◆고객번호
이름
주소
전화번호
대여
◆대여번호
대여일
고객번호
도서번호
반납여부
도서
◆도서번호
제목
분류번호
출판사
대여기간

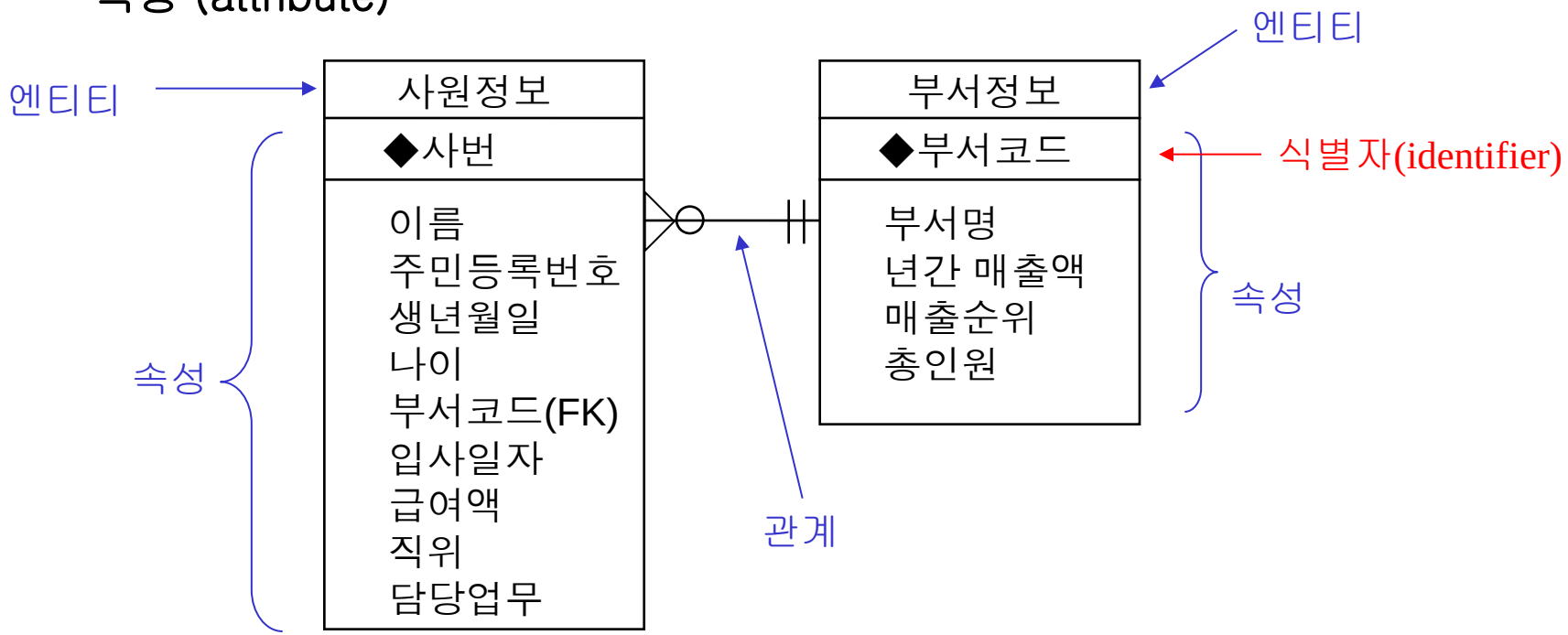
3.1 개요

❑ *Note*

- 논리적 데이터베이스 설계를 다른 말로 데이터 모델링 이라 한다.
  - '데이터 모델링'이 일반적으로 많이 쓰임
- 데이터 모델링은 전체 데이터베이스 설계에서 핵심적인 부분
- 데이터 모델링을 할 수 있기 위해서는 모델링에 사용되는 주요 개념들에 대해 알고 있어야 한다
- 데이터 모델링은 종이와 연필을 가지고도 진행할 수 있지만 효율적인 모델링을 위해 지원 도구를 사용하는 경우가 많다.
- 데이터 모델링을 수행하게 되면 최종적인 산출물(output)은 ERD 이다

3.1 개요

- 데이터 모델링의 세가지 개념
  - 엔티티 (Entity)
  - 관계 (Relationship)
  - 속성 (attribute)



3.1 개요

❑ 데이터베이스 용어 vs 모델링 용어

데이터베이스 용어	모델링 용어
테이블(table)	엔티티(entity)
컬럼(column), 열	속성(attribute)
튜플(tuple), 행(row)	인스턴스(instance)
기본키(primary key)	주식별자(primary identifier)
외래키(foreign key)	외래 식별자(foreign identifier)

<표 3.1> 데이터베이스 용어 vs 모델링 용어

3.2 엔티티(Entity)

❑ 엔티티란?

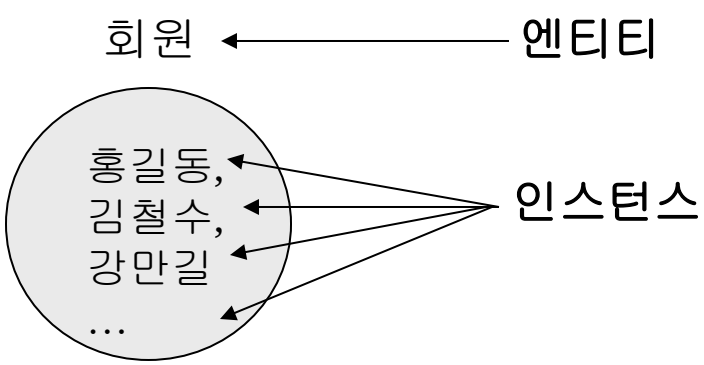
> Entity 란 업무의 관심 대상이 되는 정보를 갖고 있거나 그에 대한 정보를 알아야하는 유형, 무형의 사물이나 객체를 말한다.

- 엔티티의 예
  - 고객, 사원정보, 부서, 제품
  - 주문서, 성적표, 입고전표, 금전출납부
  - 생산계획, 공정

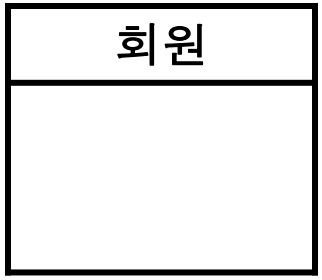


3.2 엔티티(Entity)

❑ 엔티티의 표현



<그림 3.3> 엔티티와 인스턴스



<그림 3.4> ERD에서 엔티티의 표현

3.2 엔티티(Entity)

- 엔티티의 분류
  - 유형 엔티티 : 물리적인 형태가 있고 쉽게 엔티티임을 알 수 있다. (예: 고객, 사원, 상품, 거래처, 학생, 교수, ...)
  - 무형 엔티티 : 물리적인 형태가 없고 개념적으로 존재하는 엔티티이다. (예: 생산계획, 부서조직, 색상별선호도, ...)
  - 문서 엔티티 : 업무 절차상에서 사용되는 문서나 장부, 전표에 대한 엔티티이다. (예: 거래명세서, 입출금전표, 주문서, 금전출납부, ..)
  - 이력 엔티티 : 업무상 반복적으로 이루어지는 행위나 사건의 내용을 일자별, 시간별로 저장하기 위한 엔티티이다. (예: 입고이력, 출고이력, ..)
  - 코드 엔티티 : 무형 엔티티의 일종으로 각종 코드를 관리하기 위한 엔티티이다. (예: 국가코드, 색상코드, 직급분류코드, 상태코드, ...)

3.2 엔티티(Entity)

- 엔티티의 특징 (1)
  - 시스템 구축 대상이 되는 업무에서 필요하고 관리하고자 하는 정보이어야 한다.
    - 예) 환자 : 병원 정보 시스템에서는 꼭 필요한 엔티티
      일반 회사의 정보 시스템에서는 필요하지 않음
  - 일반적으로 엔티티는 2개 이상의 인스턴스(instance)가 존재해야 의미가 있다.
    - 예) 과목
      - » 영어, 수학, 과학, ...
      - » 수학만 가르치는 보습학원에서 과목이라는 엔티티를 만드는 것이 의미가 있을까?

3.2 엔티티(Entity)

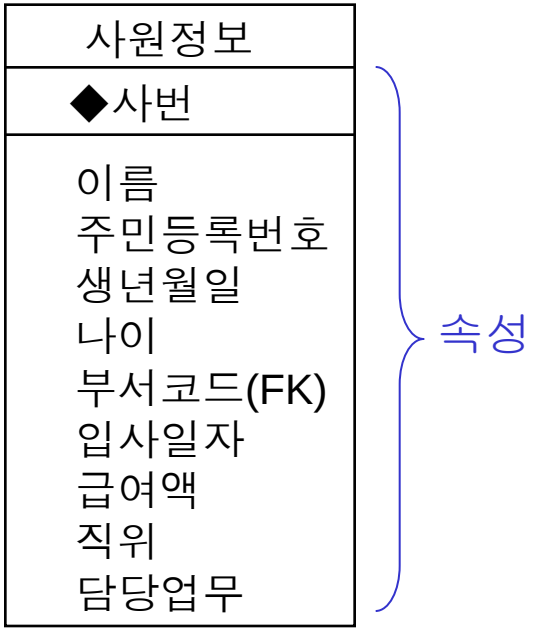
- 엔티티의 특징 (2)
  - 엔티티는 반드시 하나 이상의 속성을 가져야 한다.
    - 예) 과목
      - » 과목 코드, 과목명, 학점,..
    - 만일 속성을 찾을 수 없다면 엔티티이기 보다는 다른 엔티티의 속성일 가능성이 높다.
      - » '이름' 은 엔티티이기 보다는 속성임

3.2 엔티티(Entity)

- 엔티티의 명명(naming)
  - 어떤 엔티티에 적절한 이름을 붙이는 것은 쉬운 일이 아니다.
  - 예: 고객이 어떤 제품을 주문했는지를 관리하는 엔티티
    - 고객제품 : 고객이 주문한 제품? 고객의 제품 ?
  - 일반적인 명명 기준
    - 현업 업무에서 사용하는 용어를 사용
    - 약어를 가능하면 사용하지 않는다
    - 단수명사를 사용
    - 모든 엔티티명은 유일해야
    - 엔티티 생성 의미대로 이름을 부여

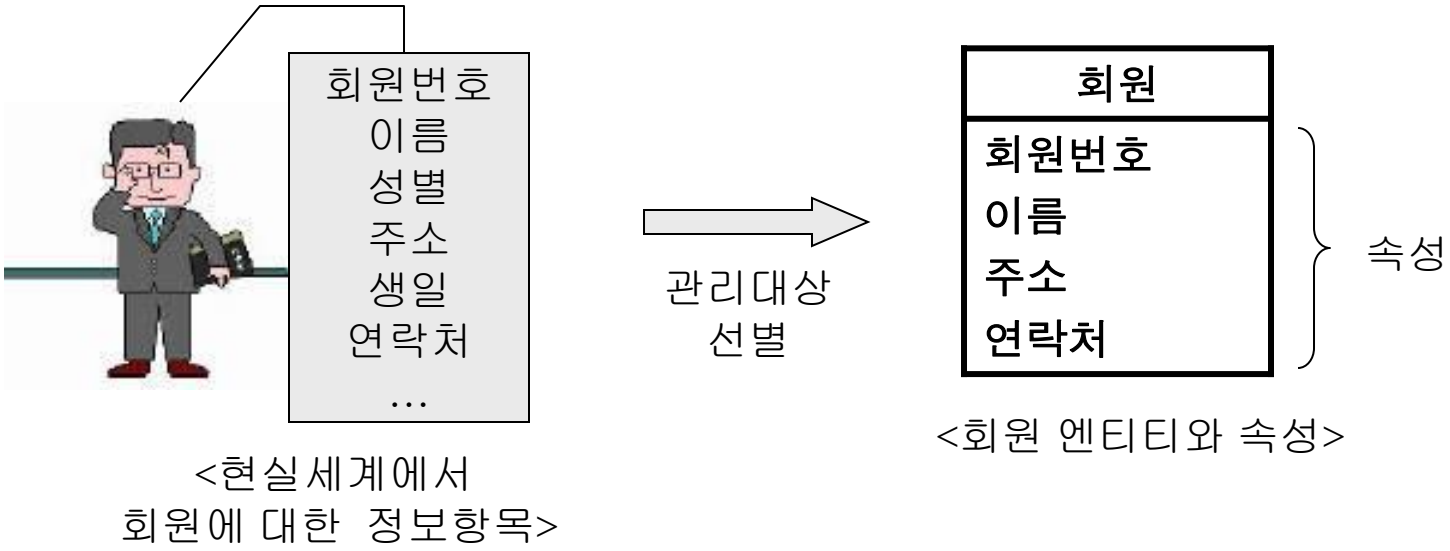
3.3 속성(Attribute)

❑ 속성이란
- 엔티티에서 관리하고자 하는 더 이상 분리되지 않는 최소 단위의 데이터
- 엔티티는 한 개 혹은 한 개 이상의 속성을 가진다
- 엔티티는 '속성들의 집합' 으로 정의될 수 있다.



3.3 속성(Attribute)

❑ 속성이란



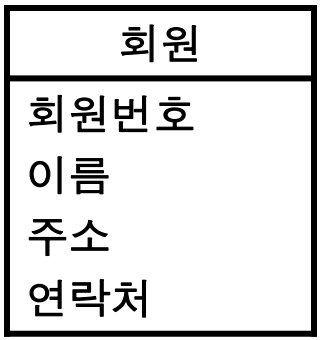
<그림 3.6> 속성의 결정

- **현실세계의 정보 항목중 업무에서 관심이 있는 정보 항목만을 속성으로 취한다**

3.3 속성(Attribute)

❑ 속성이란

- 엔터티의 속성은 나중에 테이블의 컬럼에 대응된다.



회원 엔티티

member

member_id	name	address	tel
1001	홍길동	서울	011-123-4561
1002	김우영	충남	019-555-1325
1003	김재일	부산	018-671-4435
1004	정태선	대전	011-801-1433

<그림 3.7> member 테이블

3.3 속성(Attribute)

❑ 속성의 분류

- 기본 속성 : 업무분석을 통해 현실세계로 부터 얻어낸 속성
  - 제품이름, 제조년월, 원가
- 설계속성 : 원래 현실세계에는 존재하지 않지만 설계 과정에서 만들어진 속성
  - 국가코드, 색깔 코드, 일련번호
- 유도 속성 : 다른 속성으로 부터 계산이나 변형에 의해 나온 속성
  - 금액 (= 수량 x 단가)
  - 평균 (= 합계/인원수)

➡ 7장에서 자세히 설명

3.3 속성(Attribute)

❑ 속성의 명명 규칙

- 속성의 의미가 분명히 드러나게 이름을 부여 ***
- 해당 업무에서 사용하는 이름을 부여한다.
- 서술식 속성명은 사용하지 않는다. 수식어, 소유격 자제
- 약어의 사용은 가급적 피한다.
- 엔티티에서 유일하게 식별 가능하도록 지정

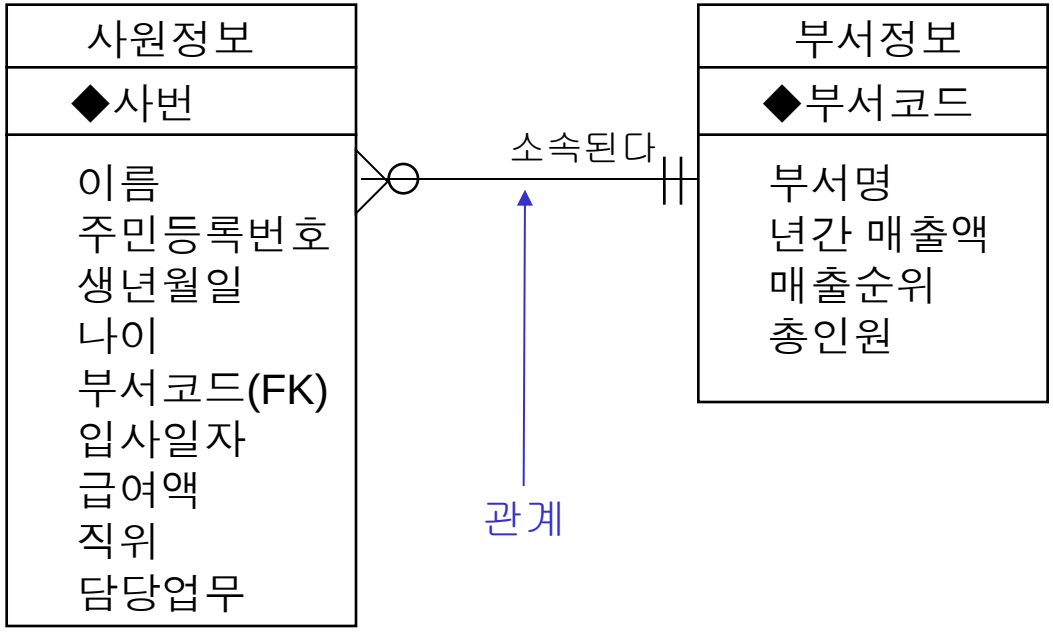
- 용어상의 혼란을 피하기 위해 사전에 용어사전(data dictionary)을 정의해 쓰는 경우도 많다.

9장에서 설명

3.4 관계(Relationship)

❑ 관계란?

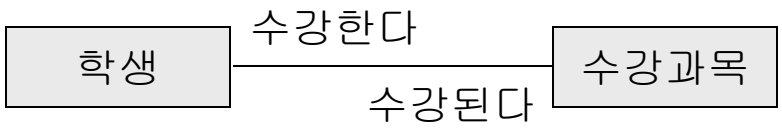
- 두개 혹은 그 이상의 엔티티들 간의 의미있는 연결
- 엔티티와 엔티티가 존재의 형태나 행위로 서로에게 영향을 주는 형태



3.4 관계(Relationship)

❑ 관계란?

– ‘학생’과 ‘수강과목’의 예



<그림 3.8> 두 엔티티 사이의 관계의 표현

– 두 엔티티가 관계가 있다는 의미는 상호 공유하는 속성이 있다는 의미이다

3.4 관계(Relationship)

학생

학번	이름
21001	김철수
21002	양길현
21003	임영수
21004	박한나

수강과목

과목
전산학개론
이산수학
웹디자인

두 엔티티는
관계가 없음

<그림 3.9> 학생과 수강과목 테이블 (공유 속성 없음)

학생

학번	이름
21001	김철수
21002	양길현
21003	임영수
21004	박한나

수강과목

학번	과목
2001	전산학개론
2001	이산수학
2002	전산학개론
2003	웹디자인
2003	이산수학

두 엔티티는
관계 있음

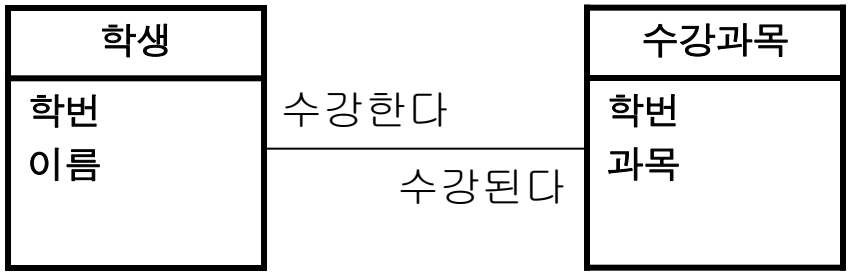
공유하는 속성

<그림 3.10> 학생과 수강과목 테이블 (공유 속성 있음)

3.4 관계(Relationship)

3.4 관계(Relationship)

❑ 관계의 명명
- 두 엔티티 사이의 관계에 대해 이름을 붙이는 것
- 어느 엔티티의 관점에서 보느냐에 따라 이름이 다르다.

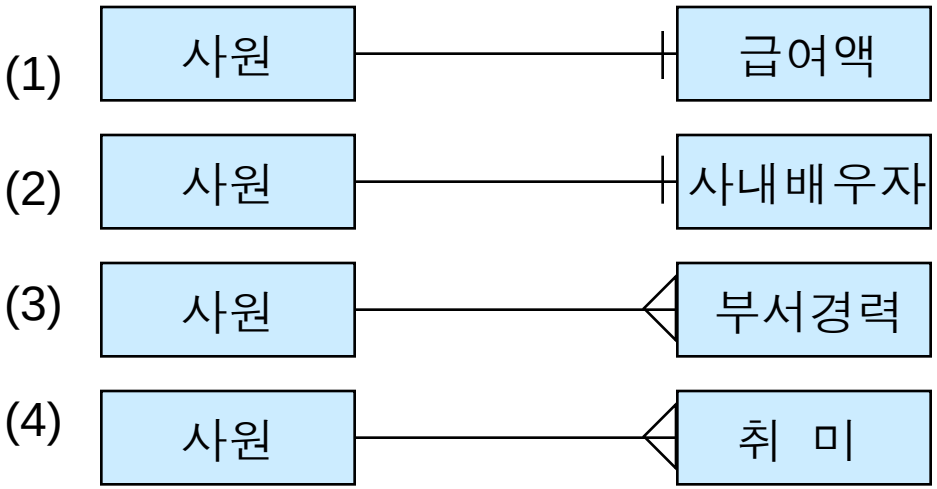


<그림 3.11> 학생과 수강과목 ERD

3.4 관계(Relationship)

❑ 관계의 카디널리티(Cardinality)

- 두개의 엔티티간의 관계에서 참여자(인스턴스)의 수를 표현한 것
  - 단일 : |
  - 다중 : <



3.4 관계(Relationship)

❑ 예제

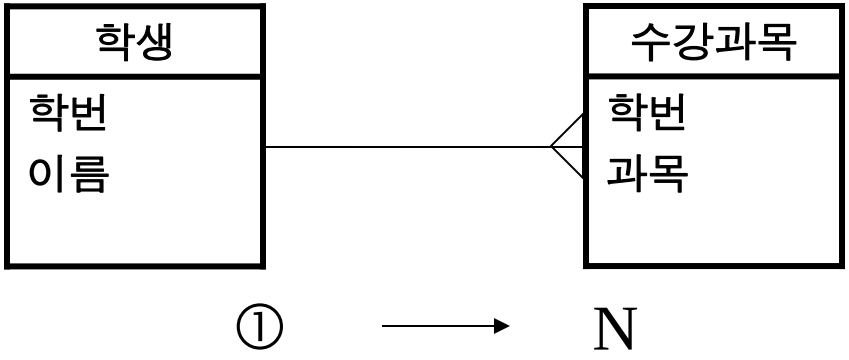
학생

학번	이름
21001	김철수
21002	양길현
21003	임영수
21004	박한나

수강과목

학번	과목
2001	전산학개론
2001	이산수학
2002	전산학개론
2003	웹디자인
2003	이산수학

<그림 3.12> 학생쪽에서 수강과목과의 카디널러티



<그림 3.13> 학생 1명이 N개의 과목과 관련이 있음

3.4 관계(Relationship)

❑ 예제

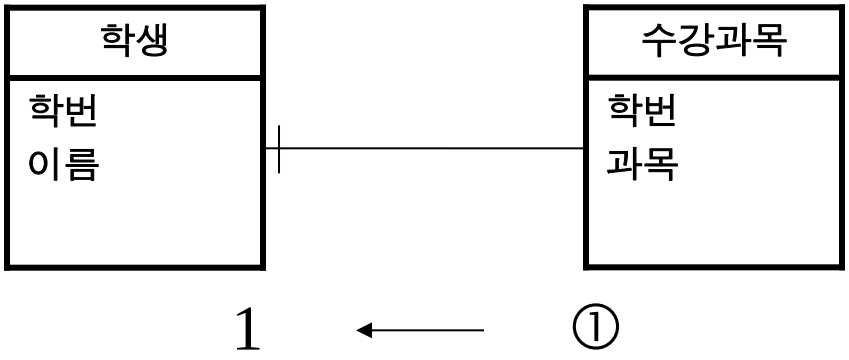
학생

학번	이름
21001	김철수
21002	양길현
21003	임영수
21004	박한나

수강과목

학번	과목
2001	전산학개론
2001	이산수학
2002	전산학개론
2003	웹디자인
2003	이산수학

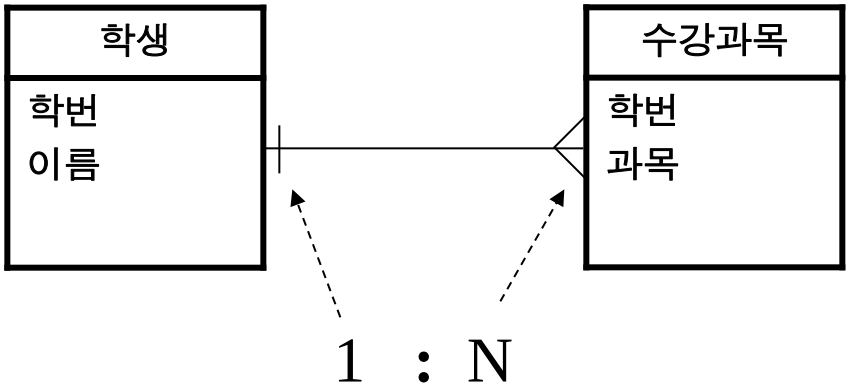
<그림 3.14> 수강과목쪽에서 학생과의 카디낼러티



<그림 3.15> 수강과목 1개의 정보는 학생 1명과 관계를 가짐

3.4 관계(Relationship)

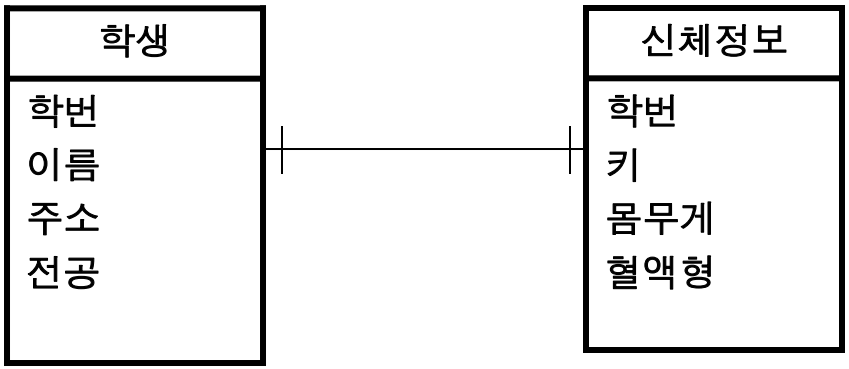
❑ 예제



<그림 3.16> 학생~수강과목간의 카디널러티

3.4 관계(Relationship)

➢ 1:1 관계



1:1 관계에 있는 두 엔티티는 하나로 합칠 수 있다.

학생

학번	이름	주소	전공
21001	김철수	서울	영문학
21002	양길현	인천	컴퓨터
21003	임영수	광주	화학
21004	박한나	부산	수학

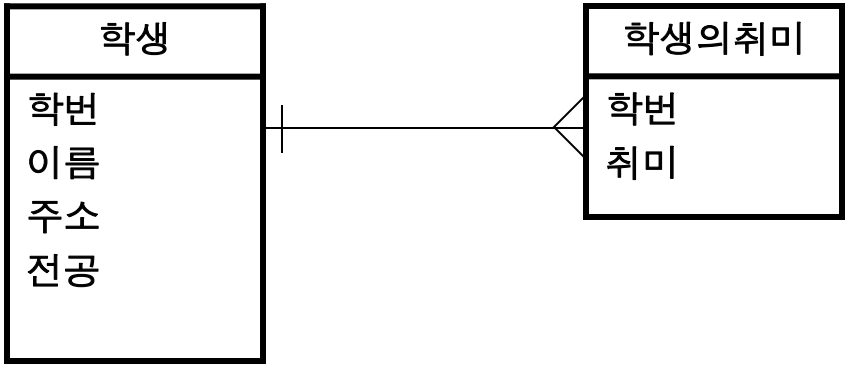
신체정보

학번	키	몸무게	혈액형
21001	175	70	A
21002	169	65	B
21003	180	60	O
21004	170	85	B

<그림 3.17> 1:1 카디낼러티의 예

3.4 관계(Relationship)

➢ 1:N 관계



대부분의 관계는
1:N 관계이다

학생

학번	이름	주소	전공
21001	김철수	서울	영문학
21002	양길현	인천	컴퓨터
21003	임영수	광주	화학
21004	박한나	부산	수학

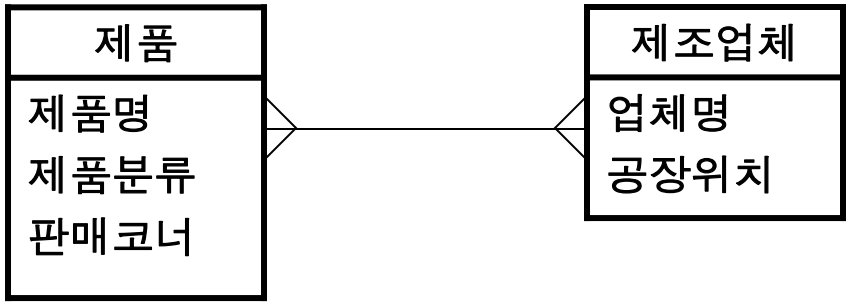
학생의취미

학번	취미
21002	낚시
21002	등산
21003	낚시
21004	여행

<그림 3.18> 1:N 카디낼러티의 예

3.4 관계(Relationship)

➢ M:N 관계



M:N 관계는 아직 완성되지 않은 작업으로 간주되며, M:N 관계를 해소하디 위한 추가 작업이 필요하다

제품

제품명	제품분류	판매코너
TV	가전	1-C
냉장고	가전	1-D
MP3	미디어	2-A
세탁기	가전	1-F

제조업체

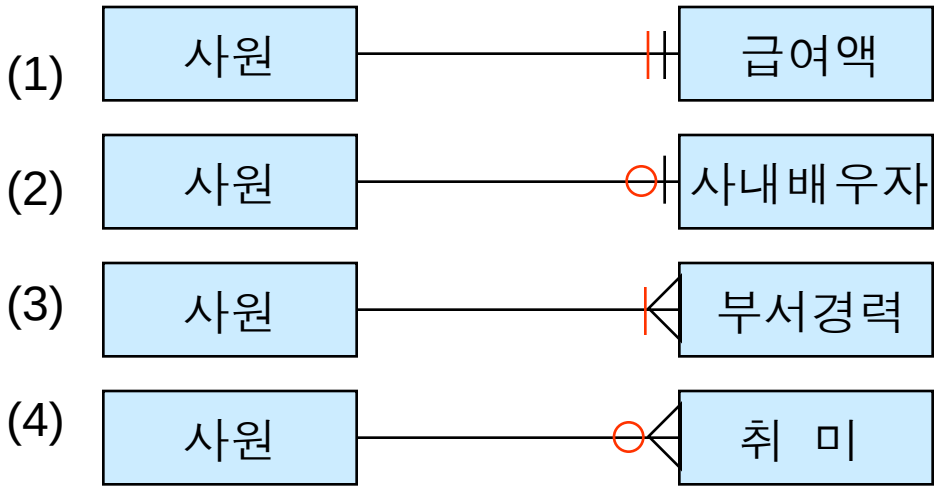
업체명	공장위치
삼성	수원
LG	강릉
대우	부산
현대	광주

<그림 3.19> M:N 카디낼러티의 예

3.4 관계(Relationship)

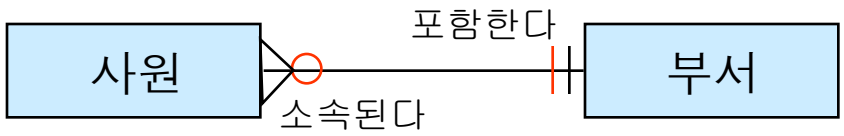
❑ 관계의 참여도

- 관계가 있는 두 엔티티의 인스턴스들이 항상 관계에 참여 하는지, 아니면 경우에 따라 관계에 참여 하는지 여부
  - 필수 : |
  - 선택 : o



3.4 관계(Relationship)

❑ 예제

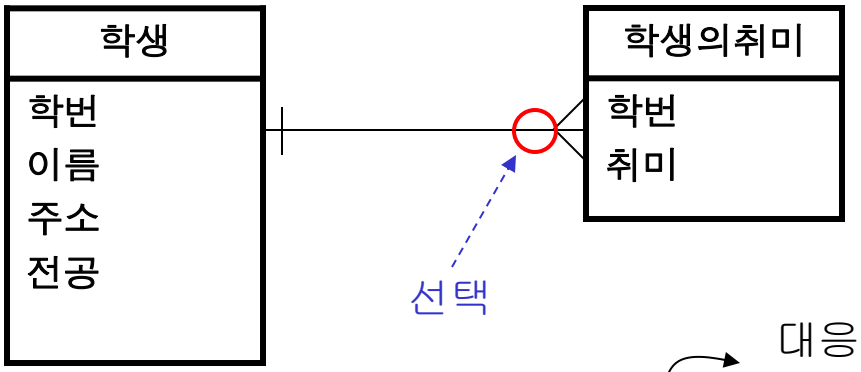


- 모든 사원은 반드시 부서를 가지며, 단 하나의 부서에 소속된다.
- 어떤 부서는 특정 시점에서 사원이 없을수도 있고, 있는 경우 여러 사원을 포함할 수 있다.

** 두 엔티티 사이의 관계가 어떠한지는 전적으로 현실세계의 의미를 따른다.

3.4 관계(Relationship)

❑ 예제



대응되는 튜플이 없음

학생

학번	이름	주소	전공
21001	김철수	서울	영문학
21002	양길현	인천	컴퓨터
21003	임영수	광주	화학
21004	박한나	부산	수학

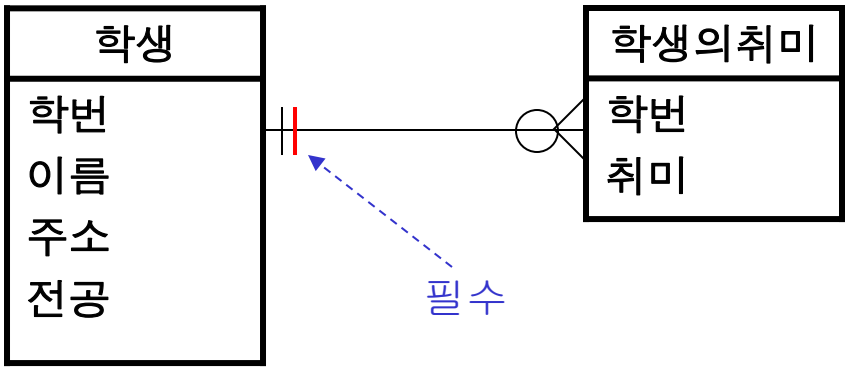
학생의취미

학번	취미
21002	낚시
21002	등산
21003	낚시
21004	여행

<그림 3.20> 학생은 취미를 선택적으로 갖는다.

3.4 관계(Relationship)

❑ 예제



학생

학번	이름	주소	전공
21001	김철수	서울	영문학
21002	양길현	인천	컴퓨터
21003	임영수	광주	화학
21004	박한나	부산	수학

학생의취미

학번	취미
21002	낚시
21002	등산
21003	낚시
21004	여행

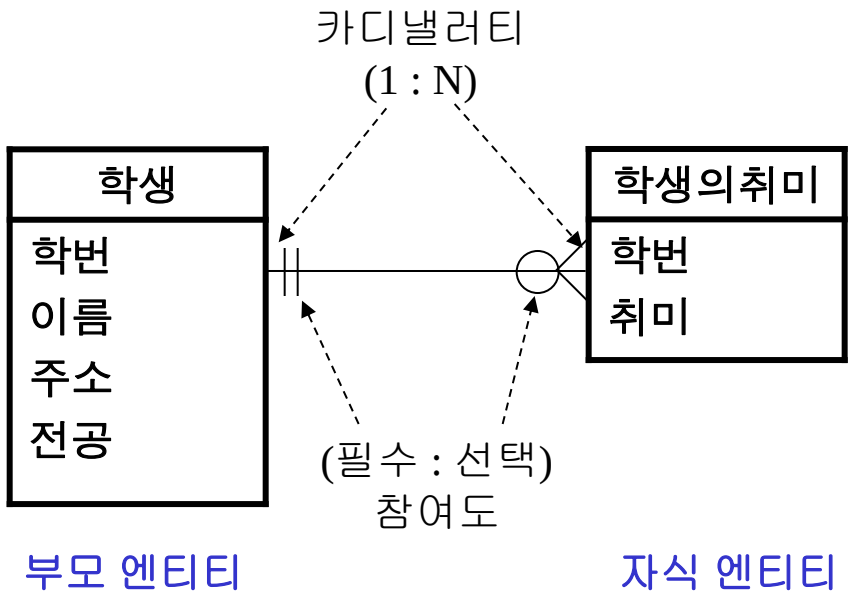
<그림 3.21> 학생의 취미정보는 관련된 학생정보를 필수적으로 갖는다.

3.4 관계(Relationship)

- 부모 엔티티와 자식 엔티티
  - 상호 관계가 있는 두 엔티티는 부모-자식의 관계에 있는 경우가 많다
  - 부모, 자식 여부는 어느쪽에 정보가 먼저 생성이 되는가에 따라 결정 된다
    - 정보가 먼저 생성되는 쪽이 부모, 가져다 쓰는 쪽이 자식
  - 두 엔티티가 부모-자식의 관계가 있다면 일반적으로 부모 엔티티와 자식 엔티티의 카디낼러티는 1:N 이고 참여도는 부모쪽이 필수, 자식쪽이 선택으로 나타난다

3.4 관계(Relationship)

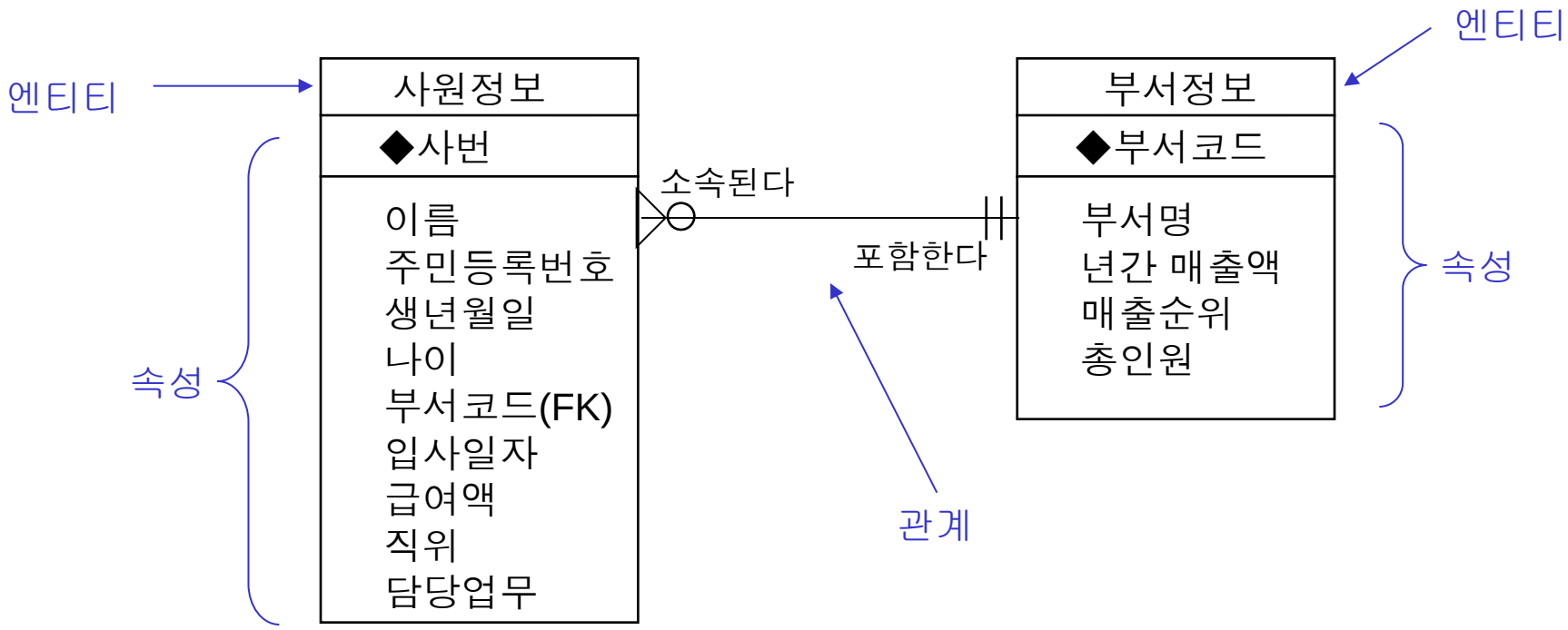
❑ 부모 엔티티와 자식 엔티티



<그림 3.22> 부모-자식 관계에서 일반적인 카디낼러티와 참여도

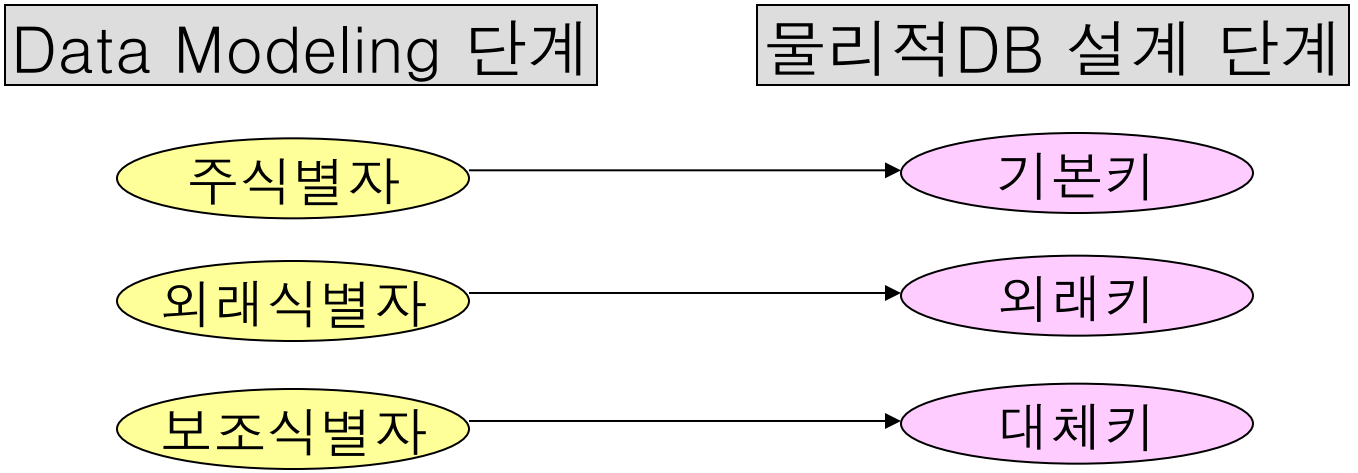
중간 정리

❑ 엔티티 / 관계 / 속성



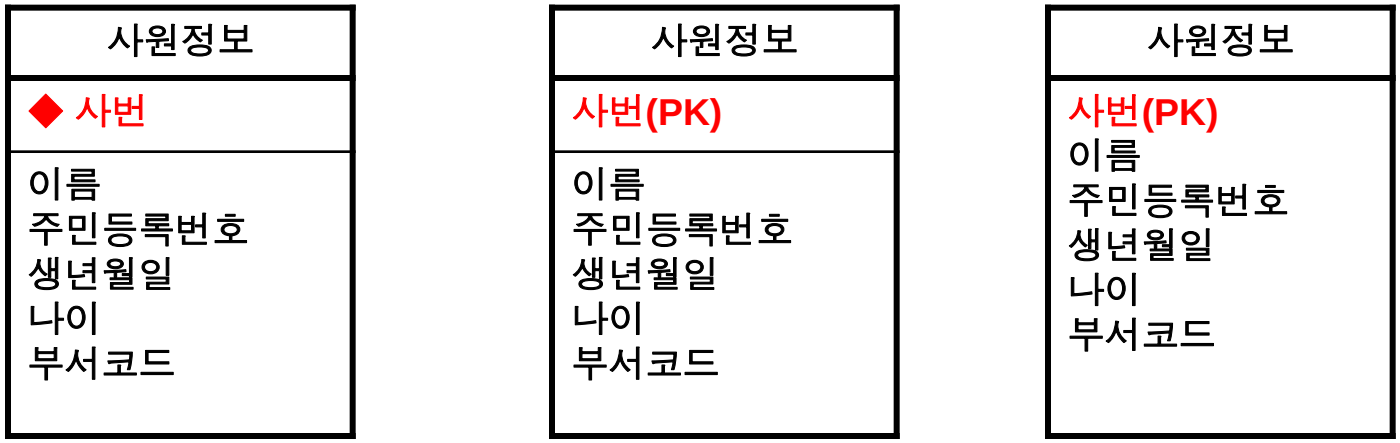
3.5 주식별자와 외래 식별자

❑ 식별자 vs 키



3.5 주식별자와 외래 식별자

❑ 주식별자의 표현 방법

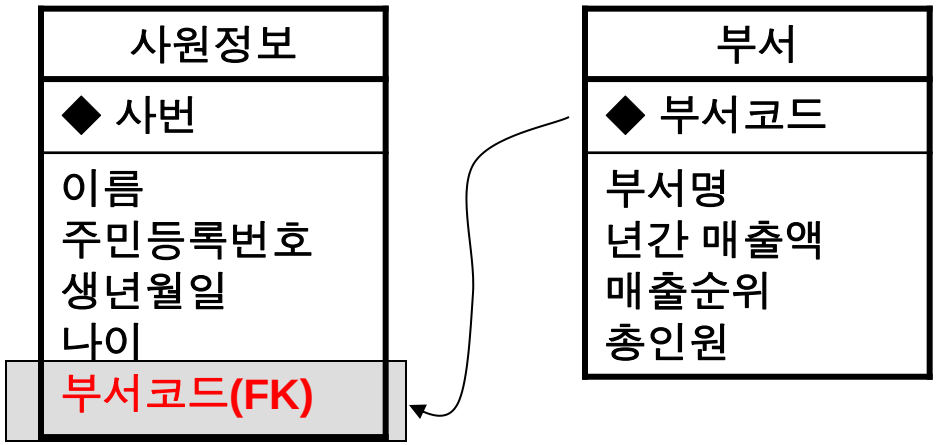


이것을 사용

엔티티 내에서 인스턴스와 인스턴스를 구별하는 기준

3.5 주식별자와 외래 식별자

❑ 외래 식별자의 표현



엔티티와 엔티티를 연결해 주는 고리 역할

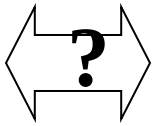
3.5 주식별자와 외래 식별자

❑ 외래 식별자의 역할

사원정보
◆ 사번
이름 주민등록번호 생년월일 나이

부서정보
◆ 부서코드
부서명 년간 매출액 매출순위 총인원

98001	홍길동	770316-1203123	77.3.16	25
98002	김철수	760519-1205431	76.5.19	26

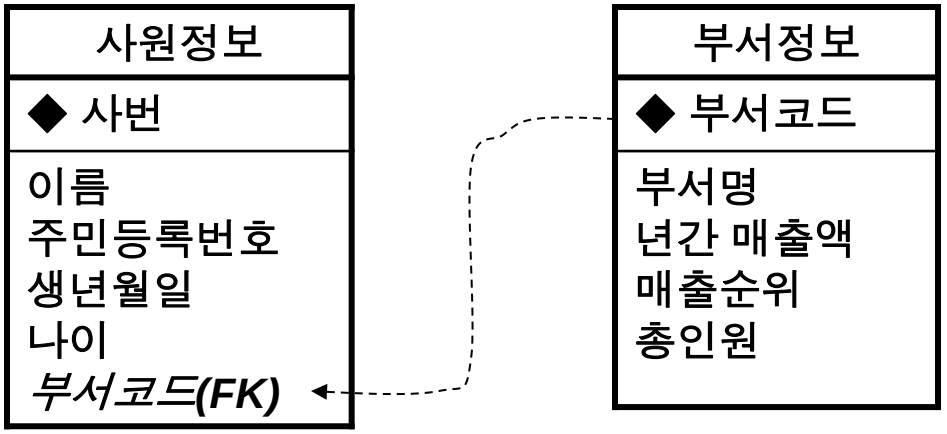


D1	영업1부	5600	2	40
D2	영업2부	7300	1	25
D3	영업3부	2500	5	10

<그림 3.24> 데이터 관점에서는 상호 관련성이 없는 두 엔티티

3.5 주식별자와 외래 식별자

❑ 외래 식별자의 역할



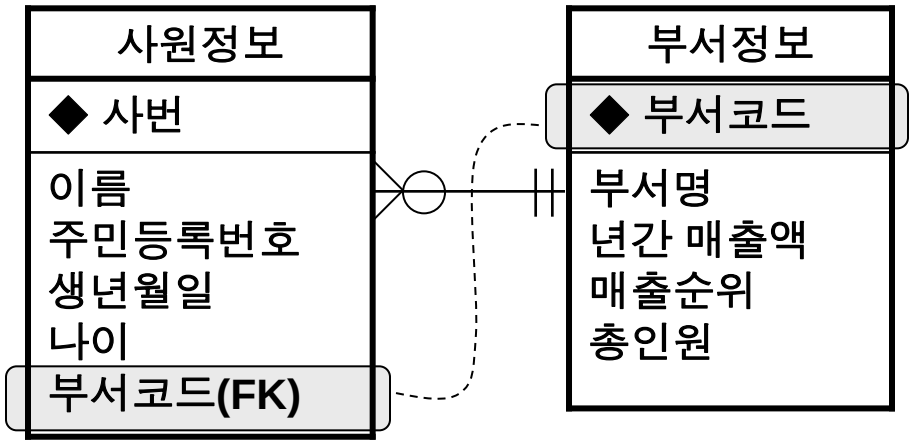
98001	홍길동	770316-1203123	77.3.16	25	D1
98002	김철수	760519-1205431	76.5.19	26	D2

D1	영업1부	5600	2	40
D2	영업2부	7300	1	25
D3	영업3부	2500	5	10

<그림 3.25> 외래식별자에 의해 연결된 두 엔티티

3.5 주식별자와 외래 식별자

❑ 부모 엔티티와 자식 엔티티에서의 식별자



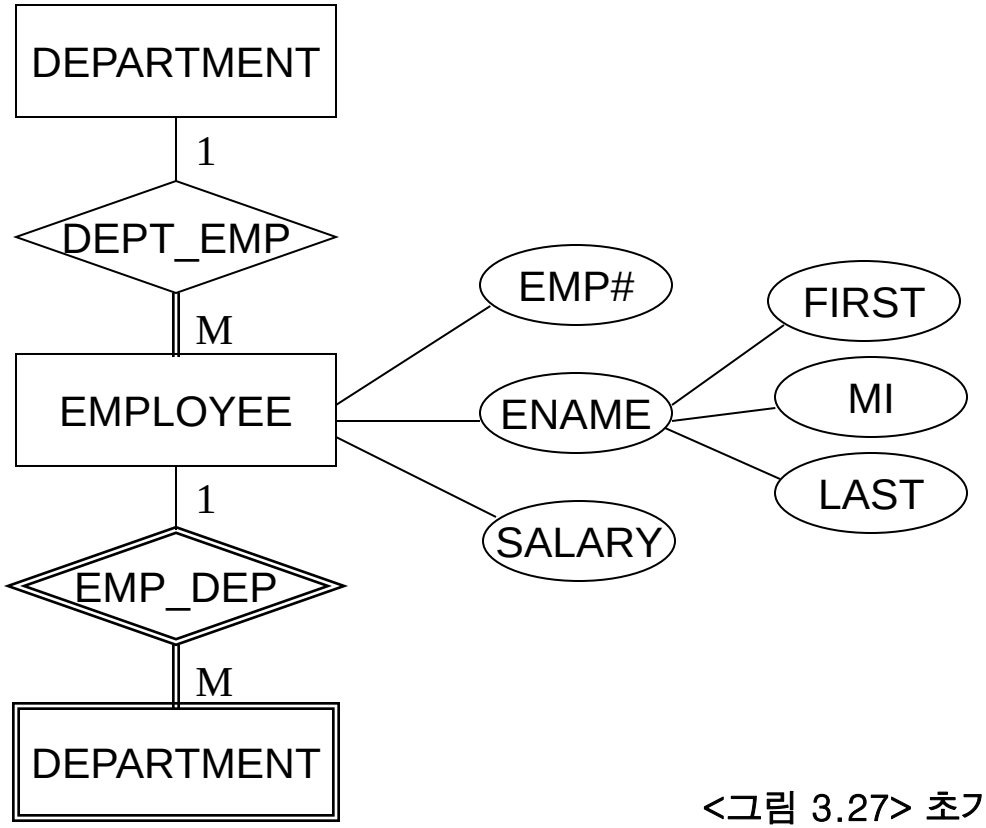
부모 엔티티의 주식별자는 자식 엔티티의 외래식별자와 연결된다.

3.5 주식별자와 외래 식별자

- *Note*
  - 관계형 데이터베이스에서
    - 주식별자 속성은 null 값을 가질 수 없다.
    - 외래식별자 속성은 null 값을 가질 수 있다.

3.6 ERD 표기법

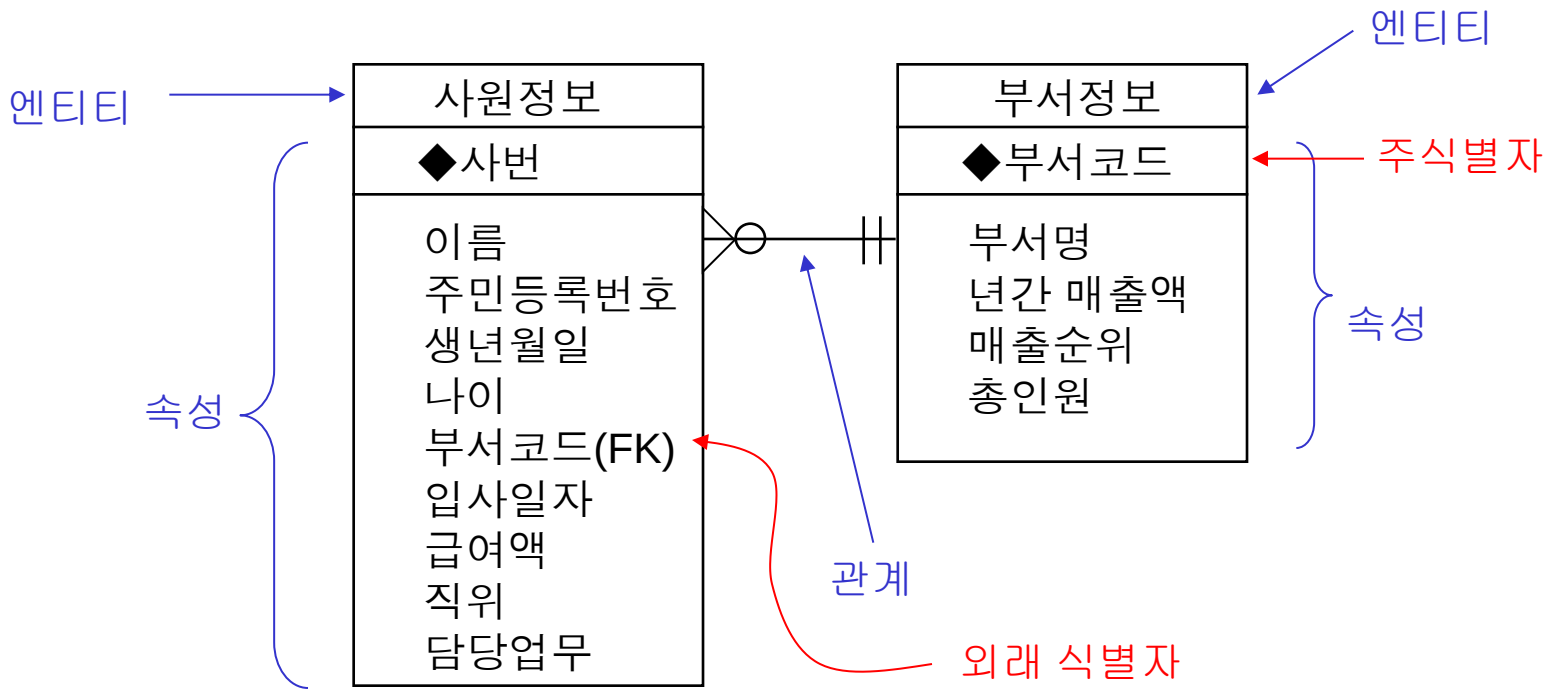
❑ 초기의 ERD



<그림 3.27> 초기의 ERD

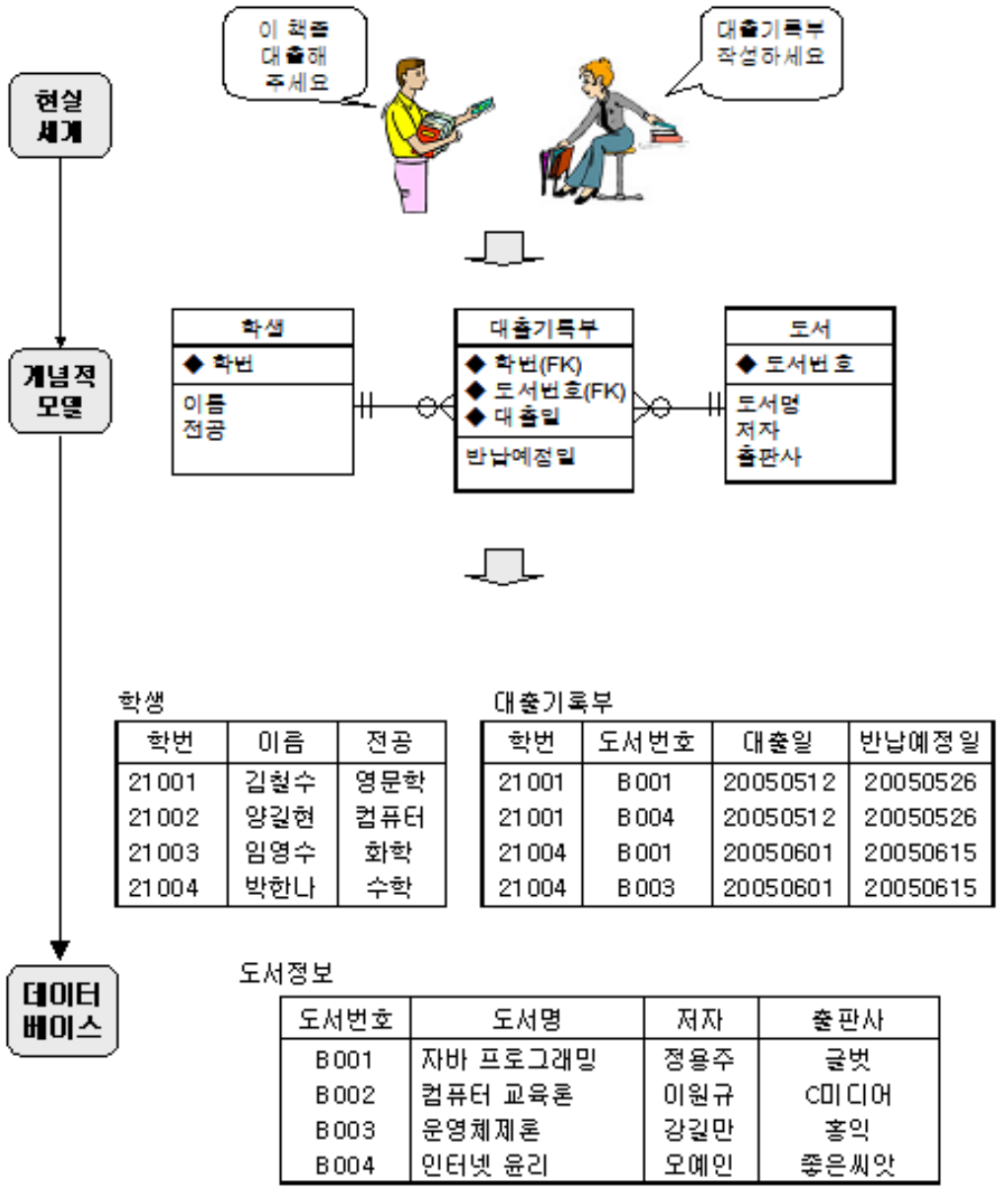
3.6 ERD 표기법

❑ 일반적으로 사용되는 ERD



단원정리

– 데이터 모델링



학생

학번	이름	전공
21001	김철수	영문학
21002	양길현	컴퓨터
21003	임영수	화학
21004	박한나	수학

대출기록부

학번	도서번호	대출일	반납예정일
21001	B001	20050512	20050526
21001	B004	20050512	20050526
21004	B001	20050601	20050615
21004	B003	20050601	20050615

도서정보

도서번호	도서명	저자	출판사
B001	자바 프로그래밍	정용주	글벗
B002	컴퓨터 교육론	이원규	C미디어
B003	운영체제론	강길만	홍익
B004	인터넷 윤리	오예인	좋은씨앗