

**CSE235**

**데이터베이스 시스템**

**(Database Systems)**

**Lecture 05: 데이터 모델링**

**담당교수: 전강욱(컴퓨터공학부)**

**kw.chon@koreatech.ac.kr**

# 지난 시간 복습

- 데이터 모델은 저장구조 + 데이터 연산
  - 관계형 모델, 네트워크 모델 등의 개요 학습
- 3단계 스키마: 외부 스키마, 개념 스키마, 내부 스키마
  - 데이터 독립성 제공
- 데이터 언어
  - 데이터정의어(DDL), 데이터조작어(DML), 데이터제어어(DCL)
- 데이터베이스 사용자
  - 데이터베이스 관리자, 최종 사용자, 응용 프로그래머
- DBMS 구조
  - 질의처리기(Query Processor), 저장 데이터 관리자(Stored Data Manager) 등
- DBMS 아키텍처
  - 중앙집중형 데이터베이스 시스템, 분산 데이터베이스 시스템, 클라이언트-서버 데이터베이스 시스템

# 복습 문제

- 자료는 현실세계에서 관찰을 통해 얻은 값을 가공 처리하여 의사 결정에 영향을 주는 것을 말한다 (   )
- 데이터의 참조는 저장되어 있는 데이터 레코드들의 주소나 위치에 의해서 이루어진다 (   )
- 데이터 정의 기능은 모든 응용 프로그램들이 요구하는 데이터 구조를 지원하기 위해 데이터베이스에 저장될 데이터의 Type과 구조에 대한 정의, 이용 방식, 제약 조건 등을 명시하는 기능이다 (   )
- 파일 시스템은 동일한 파일에 두 개 이상의 프로그램이 동시에 접근 할 수 있다 (   )

# 복습 문제 (계속)

- 외부 스키마는 데이터베이스 전체에서 특정 사용자 그룹이 관심을 가지고 있는 일부분만을 묘사한다 ( )
- 데이터베이스에 저장된 실제 데이터를 처리하는 작업에 대한 명세로서 데이터베이스를 조작하는 기본 도구를 구조라고 한다 ( )
- 데이터베이스의 논리적 구조 표현을 그래프 형태로 표현하며, 일 대 다 관계에 연관된 레코드 타입들을 각각 오너, 멤버라 하고, 이들의 관계를 오너-멤버라고도 일컫는 데이터 모델을 관계형 데이터 모델이라고 한다 ( )

# 세부 학습목표

- 1. 데이터 모델링이 무엇인지 정의할 수 있다.
- 2. 데이터 모델링의 3단계를 나열할 수 있다.
- 3. 데이터 모델을 구성하는 3가지 요소를 나열할 수 있다.
- 4. 개념적 데이터 모델이 무엇인지 설명할 수 있다.
- 5. 논리적 데이터 모델이 무엇인지 설명할 수 있다.

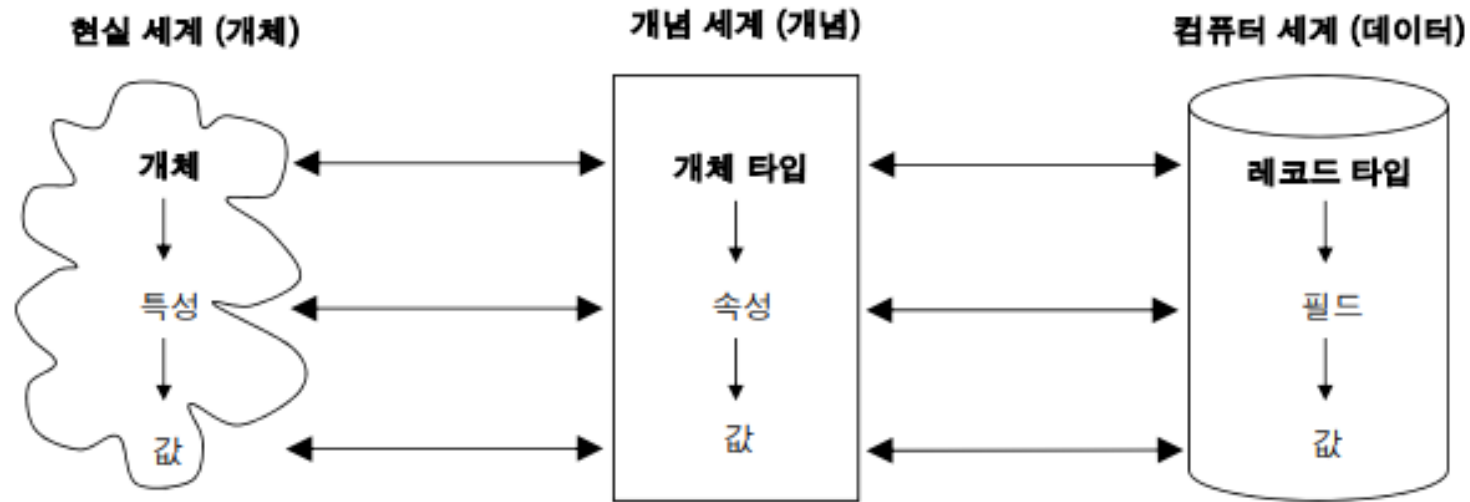
# 데이터 세계 구분

## ■ 데이터가 속한 세계와 데이터 구분

No.	세계 구분	데이터의 특징
1	현실 세계 (Real World)	오감으로 인지할 수 있는 실체로서, <b>둘 이상의 특성으로 구성된 개체(Entity)</b> 로 표현됨
2	개념 세계 (Conceptual World)	개체의 의미로부터 얻은 개념(Concept)으로서, <b>둘 이상의 속성으로 구성된 개체 타입</b> 으로 표현됨
3	컴퓨터 세계 (Computer World)	개념을 컴퓨터가 처리할 수 있도록 표현한 데이터(Data)로서, <b>둘 이상의 필드(field)로 구성된 레코드 타입</b> 으로 표현됨

# 데이터 세계 구분 (계속)

- 3가지 세계에서 본 각 데이터의 구성 요소
  - 동일한 대상도 보는 관점에 따라서, 특성 및 특성을 표현하는 방법이 달라짐



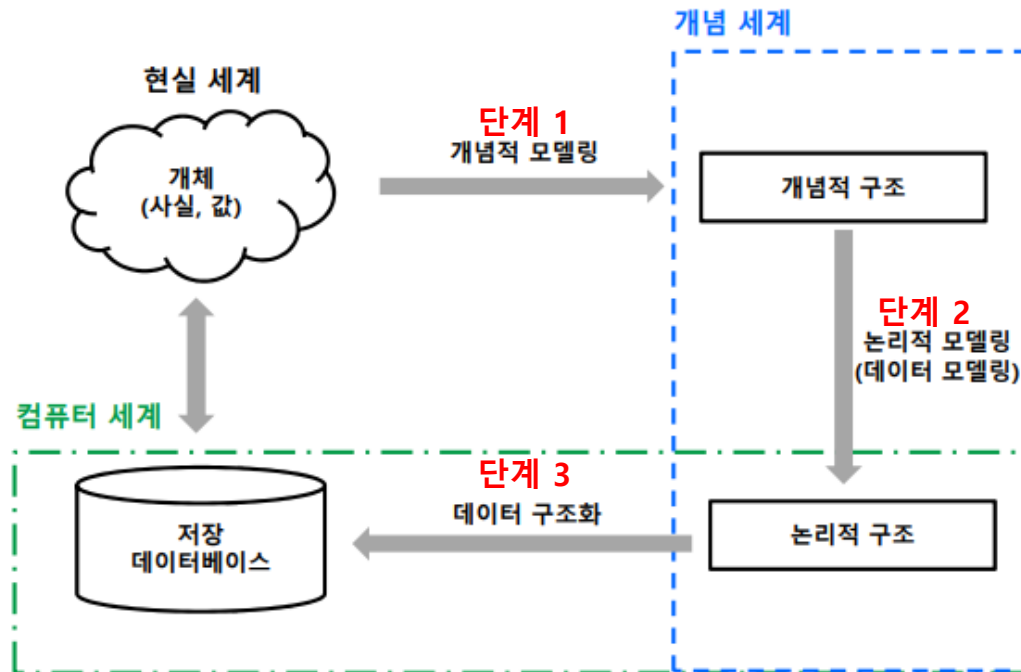
# 데이터 세계 구분 사례

구분	특징
현실 세계의 학생	모습, 목소리, 머리 색깔 등 신체적 특징, 이름, 주민등록번호, 고향, 학력, 취미, 별명, 연락처, 교우 관계, 가족 관계 등 수 없이 많은 특성으로 구성된다. ( <b>학생 개체</b> )
대학이라는 개념 세계 학생	이름, 학번, 전공, 지도교수, 이수 학점수, 평균평점 등 대학에서 필요로 하는 많은 속성들로 구성된다. ( <b>대학생 개체 타입</b> )
쇼핑몰이라는 개념 세계 학생	이름, 연락처, 주소, 배송지, 구매 실적 등 쇼핑몰의 고객 관리에 필요한 많은 속성들로 구성된다. ( <b>고객 개체 타입</b> )
대학이라는 개념 세계와 대응하는 컴퓨터 세계의 학생	이름(문자 10바이트), 학번(숫자 10자리), 전공(정수 2자리) 등 개념 세계의 속성에 대응하는 많은 필드로 구성된다. ( <b>대학생 레코드 타입</b> )



# 데이터 모델링의 개념

- 현실 세계에 존재하는 데이터를 컴퓨터의 DB로 옮기는 변환 과정
  - 데이터베이스 설계의 핵심 과정
  - 단계별로 나누어 데이터 모델링을 수행
    - 개념적 구조와 논리적 구조를 거쳐서 실제로 디스크에 저장할 수 있는 물리적 구조로 변환하는 일련의 DB 설계과정을 의미



데이터 모델링의 3단계.

# 데이터 모델링의 개념 (계속)

## ■ 개념적 데이터 모델링(Conceptual Modeling)

- 현실 세계의 중요 데이터 추출 후 개념 세계로 옮김
- 추상화(Abstraction)를 통해 DB에 저장하여 관리할 만한 가치가 있는 중요 데이터만 추출
  - 추상화: 세세하고 지엽적인 특징은 무시하고 핵심만을 추출하는 것
- 예: 학생의 경우, 학생임을 알 수 있는 데이터만 추출
  - 학교, 학년, 반 등

## ■ 개념적 모델링 방법

- 일반적으로 개체와 관계라는 추상적 개념을 이용해서 모델링함
- 주로 ER(Entity Relation) 모델이라는 개념적 모델을 사용하며, 모델링 결과를 ER 다이어그램으로 표현함

# 데이터 모델링의 개념 (계속)

## ■ 개념적 모델링 사례

현실 세계의 학생



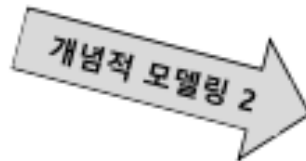
(얼굴형과 목소리, 머리 색깔, 이름, 전공 등 헤아릴 수 없을 정도로 많은 특성을 가짐)



대학이라는 개념 세계의 대학생



(여러 특성 가운데 대학 업무에 필요한 특성만 추출함)



인터넷 쇼핑몰이라는 개념 세계의 고객



(여러 특성 가운데 쇼핑몰에 필요한 특성만 추출함)

# 데이터 모델링의 개념 (계속)

## ■ 논리적 데이터 모델링(Logical Modeling)

- 개념 세계의 데이터를 **DB에 저장할 구조**로 표현하는 작업
  - DB에 어떤 구조로 저장할 것인지 결정
  - DBMS가 지원하는 논리적 데이터 모델이며, 관계형 모델, 네트워크형 모델, 계층형 모델 등이 존재
- 논리적 설계(Logical Design)와 동일한 개념

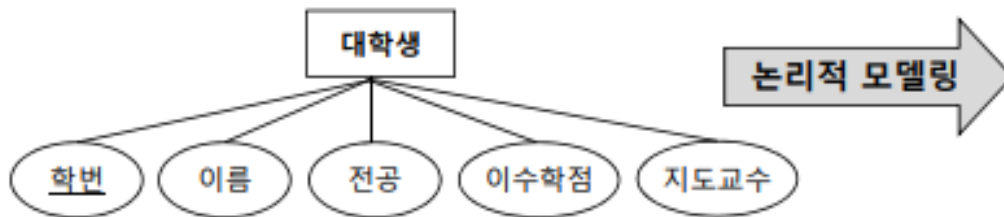
## ■ 논리적 모델링 방법

- ER 다이어그램으로 표현된 개념적 구조를 DBMS가 지원하는 논리적 데이터 모델로 변환함
- 논리적 데이터 모델의 종류에 따라 변환 방법이 다름

# 데이터 모델링의 개념 (계속)

## ■ 논리적 모델링 사례

[대학이라는 개념 세계의 대학생]



[관계형 데이터 모델]

대학생

학번	이름	전공	이수학점	지도교수

테이블  
형태 표현

혹은

대학생(학번, 이름, 전공, 이수학점, 지도교수)

리스트  
형태 표현

# 데이터 모델링의 개념 (계속)

## ■ 물리적 모델링

- 디스크에 데이터가 저장될 수 있도록 논리적 데이터 모델을 **물리적 데이터 구조**로 변환시키는 과정
- 물리적 설계(Physical Design)와 동일한 개념

## ■ 물리적 모델링 방법

- 저장 레코드 양식(각 필드의 이름, 데이터 타입, 크기 등) 정의 및 인덱스 (Index) 등을 설계함

# 데이터 모델(data model)

## ■ 데이터 모델이란?

- 데이터 모델링의 결과물을 표현하는 도구
- 데이터 및 데이터들 간의 관계 규정, 데이터에 대한 제약조건 등을 표현하는 개념적 도구

## ■ 데이터 모델의 구분

- 개념적 데이터 모델
  - DB를 개념적 구조로 표현하는 도구
  - 구성요소가 '개체'와 '관계'라는 추상적인 개념으로 구성
  - 예: Entity Relation (ER) 모델
- 논리적 데이터 모델
  - 데이터베이스를 논리적 구조로 표현하는 도구
  - 구성요소가 '레코드'와 '관계'라는 논리적 개념으로 구성된 데이터 모델
  - 예: 관계 데이터 모델, 네트워크 데이터 모델

# 데이터 모델 (계속)

- 데이터 모델은  $D = \langle S, O, C \rangle$ 로 표현
  - S (Structure): 데이터 구조
    - 개체 타입과 이들 간의 관계에 대한 명세: 현실 세계를 개념 세계로 추상화 하였을 때, 어떤 요소로 이루어져 있는지를 표현하는 개념적 구조
    - 자주 변하지 않은 정적인 특성
  - O (Operation): 연산
    - 데이터 조작 방법으로서, 개체 인스턴스를 처리하는 작업에 대한 명세: 데이터 구조에 따라 개념 세계나 컴퓨터 세계에서 실제로 표현한 값들을 처리
    - 값이 연산에 의해 계속 변경될 수 있어 동적인 특성을 가짐
  - C (Constraint): 제약조건
    - 개체 인스턴스의 존재 조건(구조적 제약과 의미상 제약)을 포함하며, 데이터 조작의 한계를 표현하는 규정에 해당함



# 데이터 모델 (계속)

## ■ 구조적 제약 및 의미상 제약 정의 사례

CREATE TABLE 부서

부서번호 NUMBER(2),

....

CONSTRAINT 부서번호\_check CHECK (부서번호 BETWEEN 10 AND 90),

... ;

구조적 제약

의미상 제약

# 개념적 데이터 모델

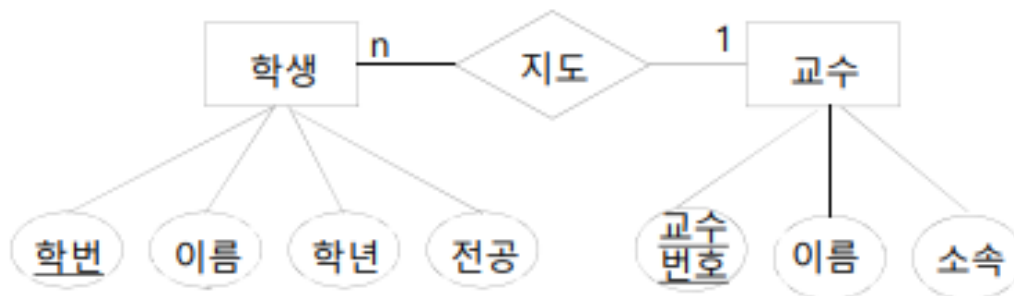
## ■ 개념적 데이터 모델이란?

- 개념적 모델링을 통해서 얻을 수 있는 추상화된 데이터 구조로서, 구성요소가 '개체'와 '관계'라는 추상적인 개념으로 구성된 데이터 모델

## ■ 특징

- DB 설계자의 이해를 돕기 위한 것으로, DBMS는 개념적 데이터 모델을 이해할 수 없음

## ■ 예: ER 모델



# 개체관계 모델

- 개체관계 모델 (ER Model, Entity-Relationship Model)
  - Peter Chen이 제안한 개념적 데이터 모델
  - 개체와 개체 간 관계를 이용해 현실 세계를 개념적 구조로 표현
  - 핵심 요소: 개체, 속성, 관계
- 개체관계 다이어그램 (ER Diagram)
  - ER 모델을 이용해 개념적으로 모델링한 결과물을 그림으로 표현한 것



Chen, Peter Pin-Shan. "The entity-relationship model—toward a unified view of data." *ACM transactions on database systems (TODS)* 1.1 (1976): 9-36.

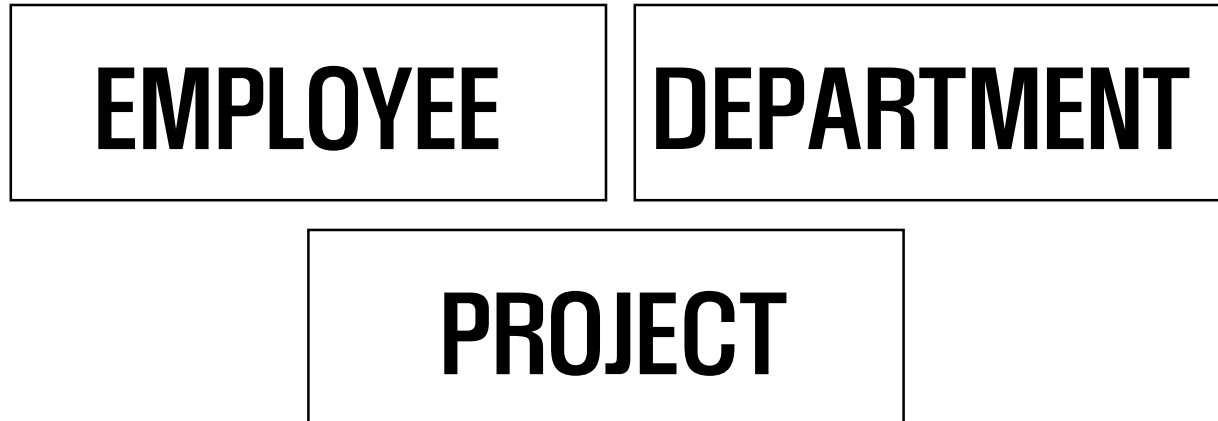
**인용수: 13,164회** (google scholar 기준)

Peter Chen 박사

(출처: <https://www.csc.lsu.edu/~chen/>).

# 개체-관계 모델: 개체 (Entity)

- 개체: 실세계에서 독립적으로 존재하는 실체
  - 저장할 가치가 있는 중요 데이터를 포함하는 사람, 사물, 사건 등
  - 다른 개체와 구별되는 이름 포함
  - 각 개체만의 고유한 특성이나 상태를 하나 이상 포함
  - 파일 구조의 레코드(Record)와 대응
  - ER 다이어그램에서 사각형으로 표현



# 개체-관계 모델: 개체 (계속)

## ■ 개체 타입 (Entity Type)

- 개체를 고유의 이름과 속성으로 정의
- 파일 구조의 레코드 타입(Record Type)에 대응

## ■ 개체 인스턴스 (Entity Instance)

- 개체를 구성하고 있는 속성이 실제 값을 가짐으로써 실체화된 개체
- 파일 구조의 레코드 인스턴스(Record Instance)에 대응
- Entity Occurrence라고도 불림

## ■ 개체 집합 (Entity Set)

- 특정 개체 타입에 대한 개체 인스턴스들을 모아 놓은 것

개체 타입의 이름:

**EMPLOYEE**

**COMPANY**

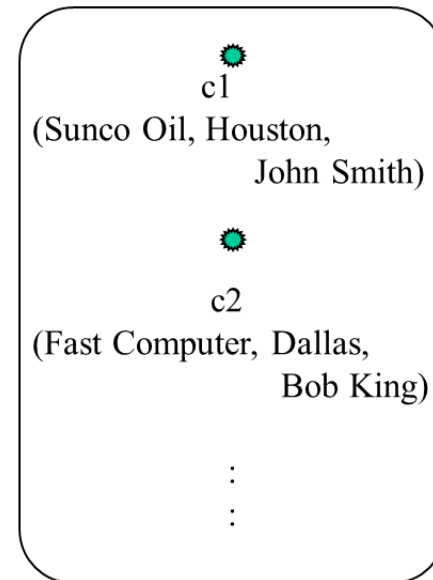
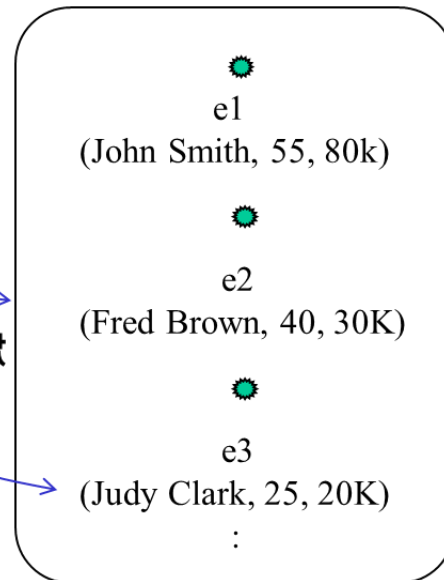
애틀리뷰트 이름

Name, Age, Salary

Name, Headquarters, President

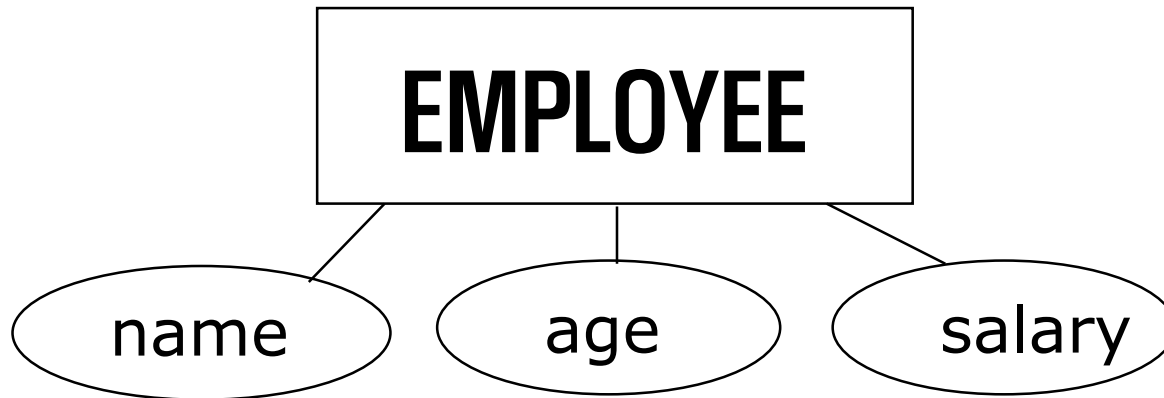
개체 집합  
[외연]

애틀리뷰터 값



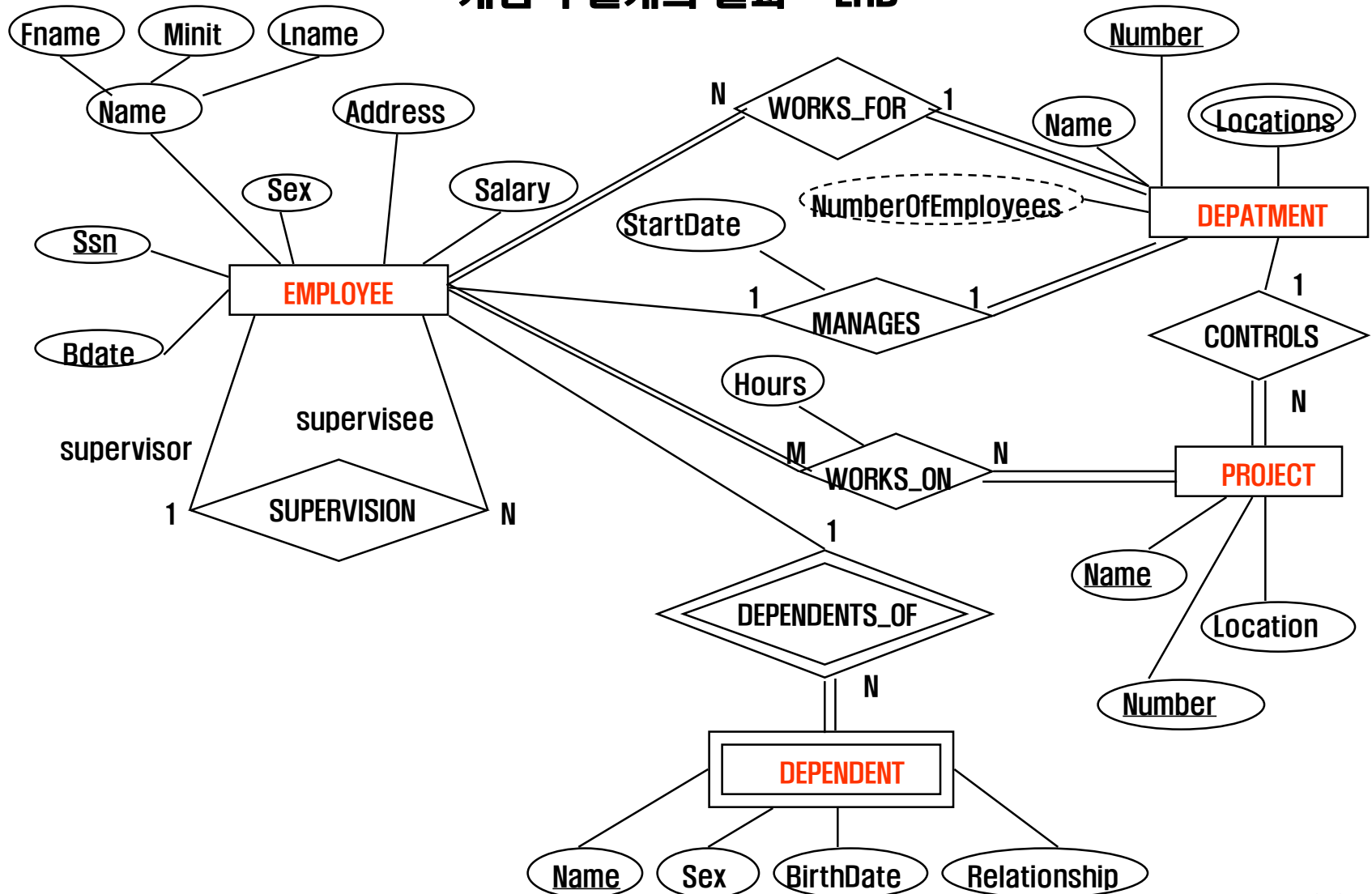
# 개체-관계 모델: 속성

- 개체나 관계가 가지고 있는 고유의 특성
  - 개체는 반드시 하나 이상의 특성을 포함
- 의미 있는 데이터의 가장 작은 논리적 단위
- 파일 구조의 필드(Field)와 대응
- ER 다이어그램에서 타원으로 표현
  - 예: EMPLOYEE의 속성: Name, Age, Salary



# 개체-관계 모델: 속성 (계속)

개념적 설계의 결과 - ERD



# 개체-관계 모델: 속성 (계속)

## ■ 속성의 분류

- 기준 1: 속성값의 개수
  - 단일 값 속성, 다중 값 속성
- 기준 2: 의미의 분해 가능성
  - 단순 속성, 복합 속성
- 기준 3: 다른 속성에 의해 값이 유도되어 결정
  - 유도 속성



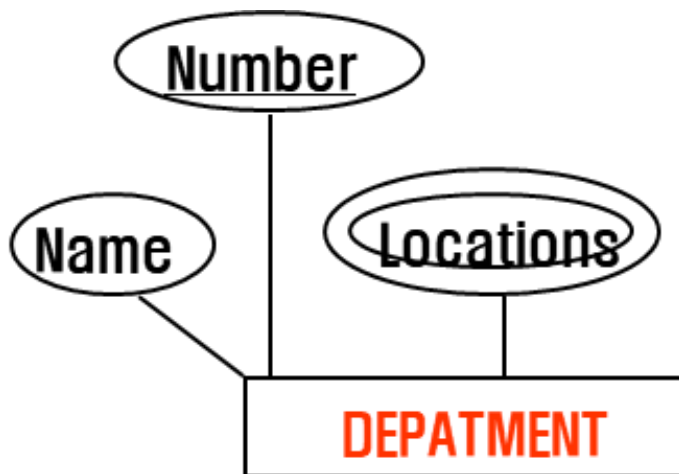
# 개체-관계 모델: 속성 (계속)

## ■ 단일 값 속성(Single-Valued Attribute)

- 값을 하나만 가질 수 있는 속성
- 예: 사원 개체의 이름, 급여

## ■ 다중 값 속성(Multi-Valued Attribute)

- 값을 여러 개 가질 수 있는 속성
- ER 다이어그램에서 이중 타원으로 표현
- 예: 사원 개체의 연락처 속성(집전화번호, 휴대폰 등)



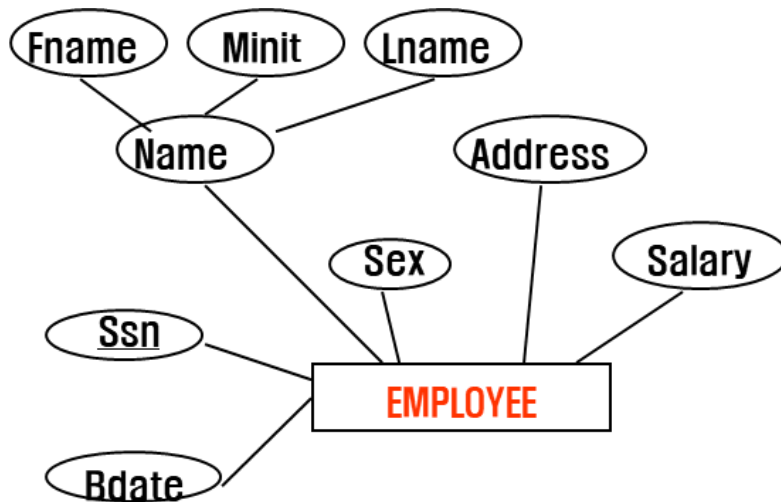
# 개체-관계 모델: 속성 (계속)

## ■ 단순 속성(Simple Attribute)

- 의미를 더는 분해할 수 없는 속성
- 예: 사원 개체의 급여, 성별 등

## ■ 복합 속성(Composite Attribute)

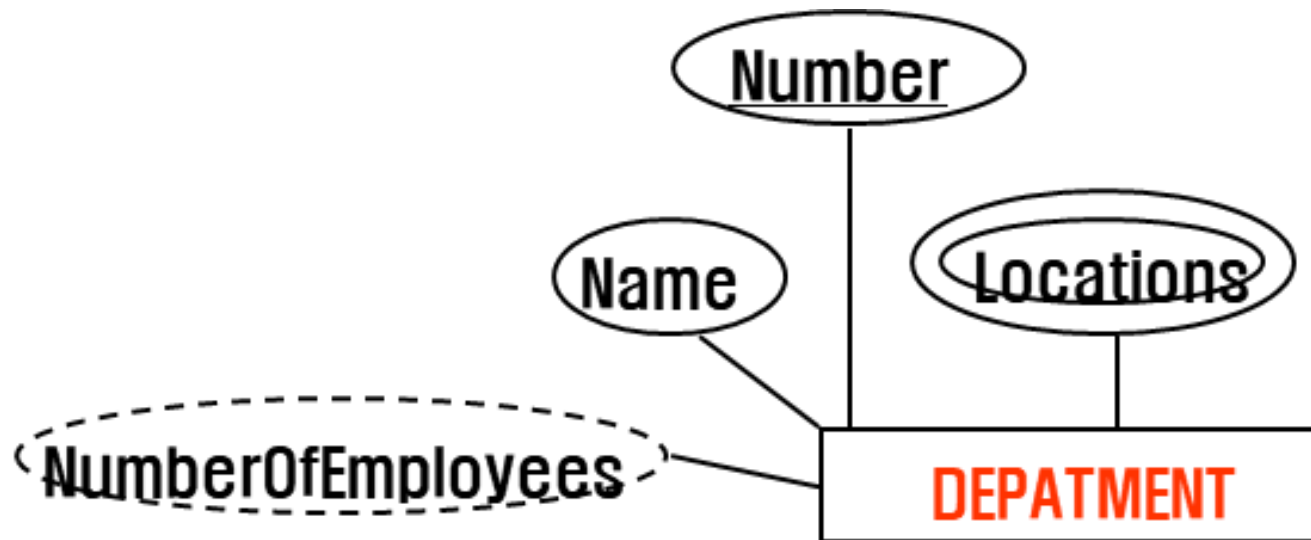
- 의미를 분해할 수 있는 속성
- 예: 사원 개체의 이름 속성
  - 성, 이름 등으로 의미를 세분화 할 수 있음



# 개체-관계 모델: 속성 (계속)

## ■ 유도 속성 (Derived Attribute)

- 기존의 다른 속성의 값에서 유도되어 결정되는 속성
- 값이 별도로 저장되지 않음
- ER 다이어그램에서 점선 타원으로 표현
- 예: 사원 개체의 전체 수



# 개체-관계 모델: 속성 (계속)

- **널 속성 (Null Attribute)**

- 널 값이 허용되는 속성

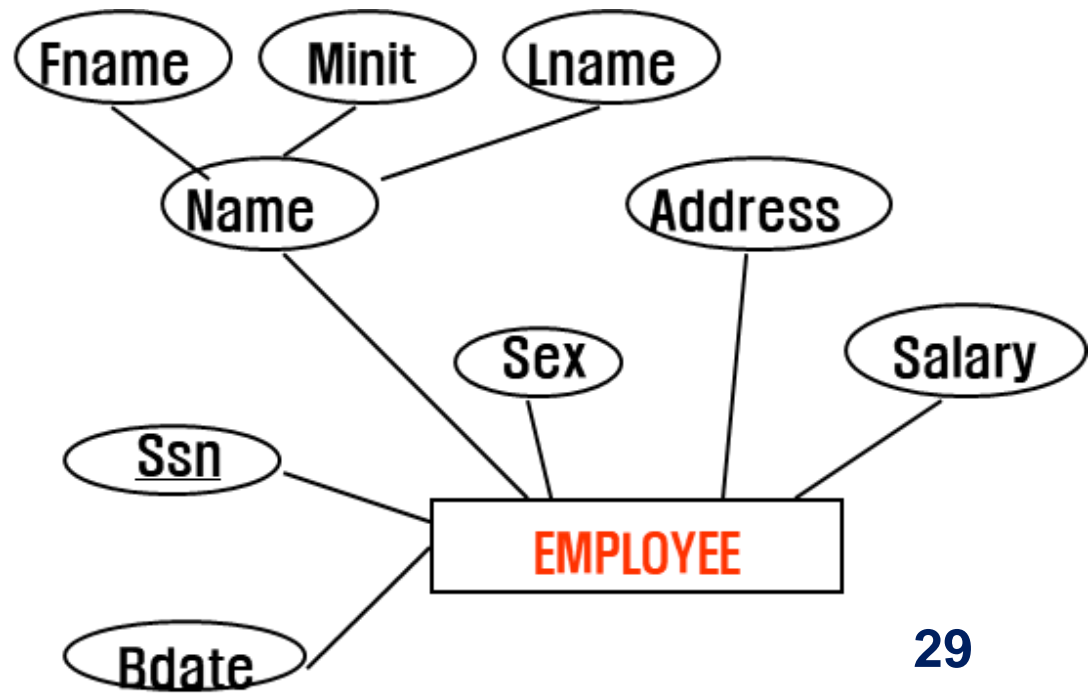
- **널 (Null) 값**

- 아직 결정되지 않거나 존재하지 않는 값
- 공백과 0과는 의미가 다름

# 개체-관계 모델: 속성 (계속)

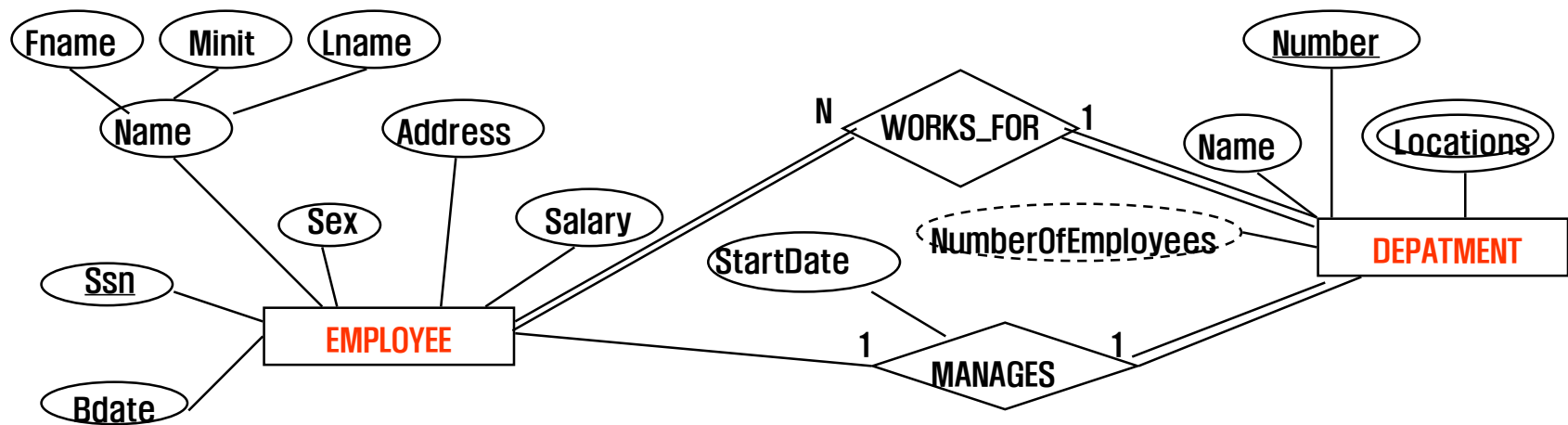
## ■ 키 속성 (Key Attribute)

- 각 개체 인스턴스를 식별하는데 사용되는 속성
- 모든 개체 인스턴스의 키 속성 값이 다름
- 둘 이상의 속성들로 구성되기도 함
  - 사원 개체 이름 + 주소로 구별
- ER 다이어그램에서 밑줄로 표현
- 예: 사원 개체의 ssn ID



# 개체-관계 모델: 관계 (Relationship)

- 개체와 개체가 맺고 있는 의미 있는 연관성
- 개체 집합들 간 사상(Mapping)을 의미
- ER 다이어그램에서 마름모로 표현
- 예: "사원 개체는 부서를 위하여 일한다 "



# 개체-관계 모델: 관계 (계속)

- **관계의 유형: 관계에 참여하는 개체 타입의 수 기준**
  - 이항 관계: 개체 타입 두 개가 맺는 관계
  - 삼항 관계: 개체 타입 세 개가 맺는 관계
  - 순환 관계: 개체 타입 하나가 자기 자신과 맺는 관계
- **관계의 유형: 매핑 카디널리티 기준 (ER 다이어그램에서 레이블로 표기)**
  - 일 대 일 (1:1) 관계
  - 일 대 다 (1:n) 관계
  - 다 대 다 (n:m) 관계
- **매핑 카디널리티(Mapping Cardinality)**
  - 관계를 맺는 두 개체 집합에서, 각 개체 인스턴스가 연관성을 맺고 있는 상대 개체 집합의 인스턴스 개수

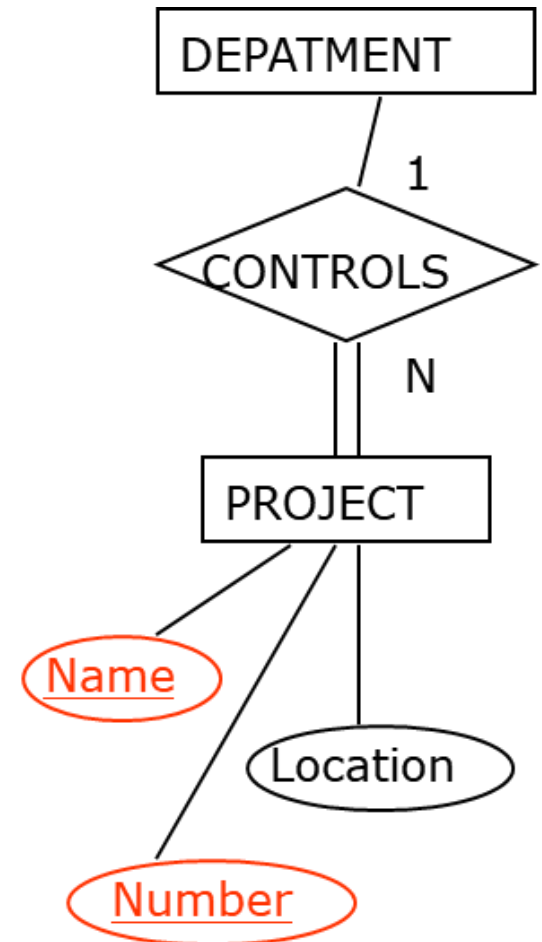
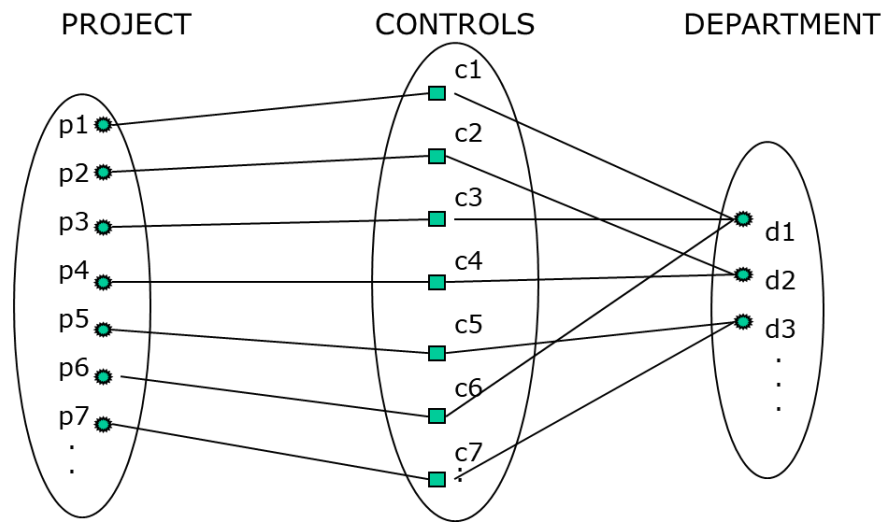
# 개체-관계 모델: 관계 (계속)

## ■ 일 대 일(1:1) 관계

- 개체 A의 각 개체 인스턴스가 개체 B의 개체 인스턴스 하나와 관계를 맺을 수 있음
- 역으로도 마찬가지
- 예: 남편 개체와 아내 개체 관계

## ■ 일 대 다(1:n) 관계

- 개체 A의 각 개체 인스턴스가 개체 B의 개체 인스턴스 여러 개와 관계를 맺을 수 있음
- 개체 B의 각 개체 인스턴스는 개체 A의 개체 인스턴스 하나만 관계를 맺을 수 있음
- 예: 부서 개체와 프로젝트 개체 관계

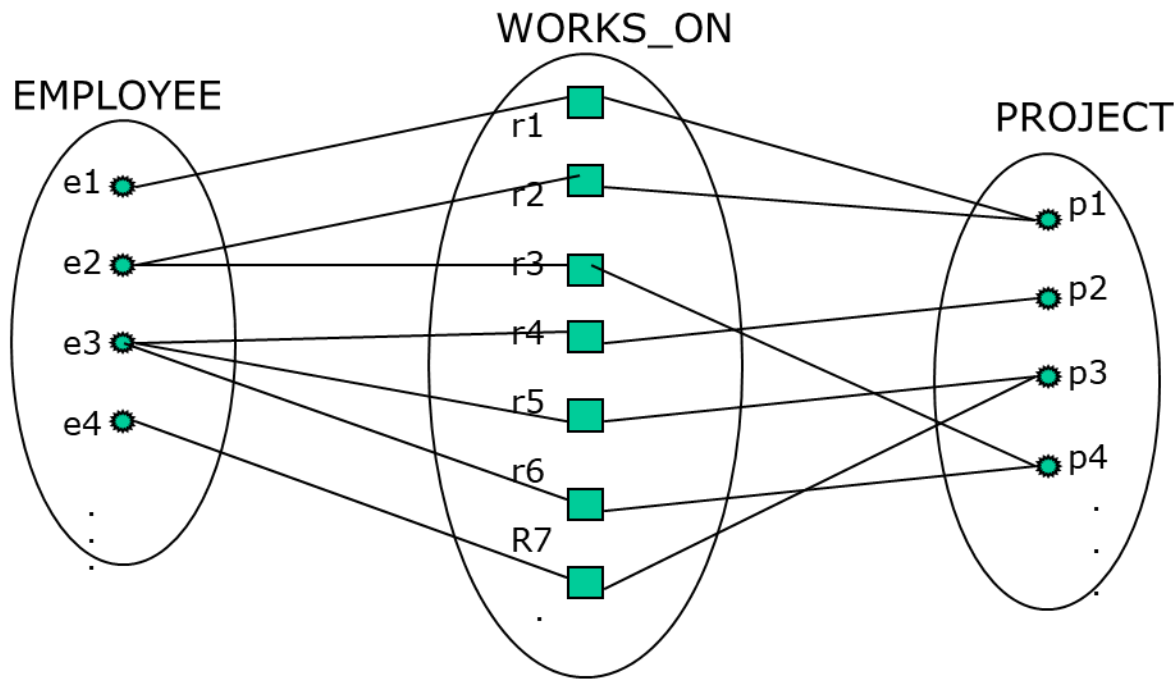




# 개체-관계 모델: 관계 (계속)

## ■ 다 대 다(n:m) 관계

- 개체 A의 각 객체 인스턴스가 개체 B의 개체 인스턴스 여러 개와 관계를 맺을 수 있음(그 역도 가능)
- 예: 사원 개체와 프로젝트 개체와의 관계
  - 사원은 여러 개의 프로젝트를 수행할 수 있음
  - 프로젝트 역시 여러 명의 사원에 의해서 수행할 수 있음



# 개체-관계 모델: 관계 (계속)

## ■ 관계의 참여 특성

### □ 필수적 참여(전체 참여)

- 모든 개체 인스턴스가 관계에 반드시 참여해야 함
- ER 다이어그램에서 이중선으로 표현
- 예: 사원 개체가 부서에 반드시 포함되어야 함

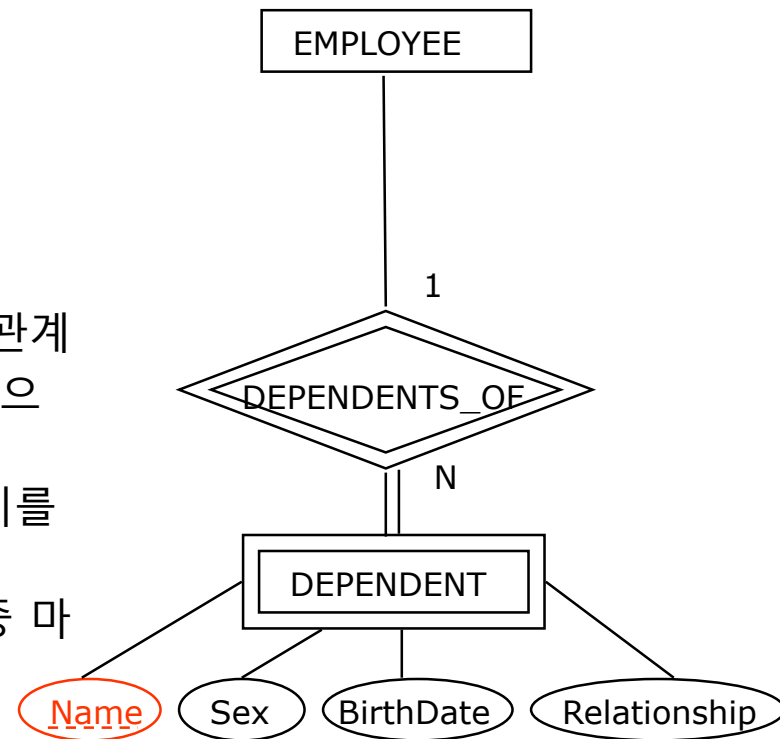
### □ 선택적 참여(부분 참여)

- 개체 인스턴스 중 일부만 관계에 참여해도 되는 것을 의미

# 개체-관계 모델: 관계 (계속)

## ■ 관계의 종속성

- 약한 개체(Weak Entity)
  - 다른 개체의 존재 여부에 의존적인 개체
  - ER 다이어그램에서 이중 사각형으로 표현
- 오너 개체(Owner Entity)
  - 다른 개체의 존재 여부를 결정하는 개체
- 약한 개체와 오너 개체 관계
  - 오너 개체와 약한 개체는 일반적으로 1:n 관계
  - 약한 개체는 오너 개체와의 관계에 필수적으로 참여
  - 약한 개체는 오너 개체의 키를 포함하여 키를 구성
  - 약한 개체와 오너 개체와 맺는 관계는 이중 마름모로 표현
  - 예: 직원 개체와 부양가족 개체
    - 직원이 없으면 부양가족 없음
    - 강한 개체: 직원 개체
    - 약한 개체: 부양가족 개체


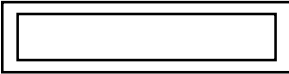
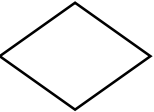
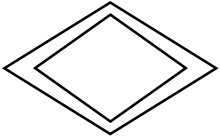





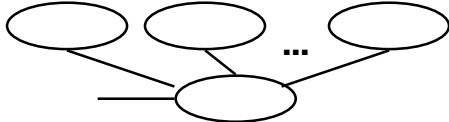



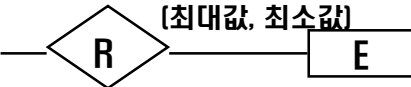
# 개체-관계 모델: 관계 (계속)

- ER 다이어그램, 명명에 대한 규칙, 설계 관한 사항
  - 스키마에 사용된 각 구조물에 대해 가능한 한 그 의미를 전달할 수 있는 이름 선택
  - 복수보다 단수 이름 선택
  - 일반적으로 자연어로 기술된 요구 사항에서 **명사는 엔티티 타입 이름, 동사는 관계 타입 이름으로** 해석하는 것이 도움이 됨
  - ER 다이어그램은 **왼쪽에서 오른쪽, 위에서 아래로** 읽기 쉽게 작성함
  - 스키마 설계는 반복해서 개선하는 작업이 필요함 – **한번에 완성하기는 쉽지 않음**

# 개체-관계 모델: 관계 (계속)

## ■ ER 다이어그램 표기법 요약

Symbol	Meaning
	엔티티 타입
	약한 엔티티 타입
	관계 타입
	식별 관계 타입
	애트리뷰트
	키 애트리뷰트
	다치 애트리뷰트

Symbol	Meaning
	복합 애트리뷰트
	유도된 애트리뷰트
	E2가 R에 전체참여 E1이 R에 부분참여
	R에서 E1:E2의 카디널리티 비율이 1:N
	R에서 E의 참여에 대한 구조적 제약조건 [최대값, 최소값] E가 R의 관계를 몇 개나 가지는가?

# 논리적 데이터 모델

## ■ 논리적 데이터 모델이란?

- 개념적 데이터 모델을 DB로 구현하기 위한 중간 단계로서, 구성 요소가 '**레코드**'와 '**관계**'라는 **논리적인 개념**으로 구성된 데이터 모델

## ■ 특징

- DBMS는 하나의 논리적 데이터 모델을 기반으로 개발되므로, DBMS는 논리적 데이터 모델을 이해할 수 있다

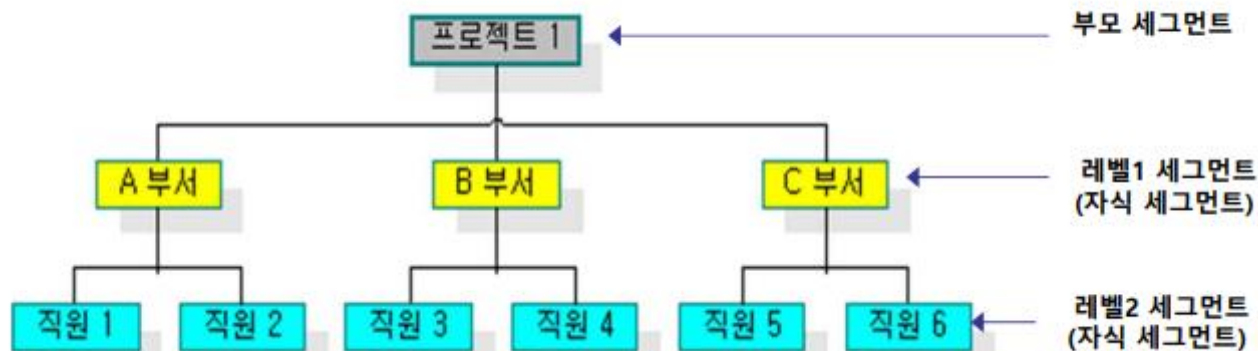
## ■ 예: 관계 데이터 모델, 계층 데이터 모델, 네트워크 데이터 모델 등

학생	<table><tr><td><u>학번</u></td><td>이름</td><td>학년</td><td>학과</td></tr></table>	<u>학번</u>	이름	학년	학과	or 학생( <u>학번</u> , 이름, 학년, 학과)
<u>학번</u>	이름	학년	학과			
교수	<table><tr><td><u>교수번호</u></td><td>이름</td><td>학과</td></tr></table>	<u>교수번호</u>	이름	학과	or 교수( <u>교수번호</u> , 이름, 학과)	
<u>교수번호</u>	이름	학과				
지도하다	<table><tr><td><u>교수번호</u></td><td><u>학번</u></td></tr></table>	<u>교수번호</u>	<u>학번</u>	or 지도하다( <u>교수번호</u> , <u>학번</u> )		
<u>교수번호</u>	<u>학번</u>					

# 논리적 데이터 모델 (계속)

## ■ 계층 데이터 모델 (Hierarchical Data Model)

- 데이터베이스의 **논리적 구조가 트리(Tree) 형태로 표현**되며, ER 모델의 개체를 레코드 타입으로 표현
- 1960년대 후반에 최초의 계층 DBMS가 등장 (IBM사의 IMS)
- ER 모델의 1:n 관계를 **두 레코드 타입 간의 부모-자식 관계로 표현**
  - 레이블은 기술할 필요가 없음
- 필드의 집합을 **세그먼트(Segment)라 칭하며, 부모 세그먼트부터 시작 하는 계층구조**를 구성
- 부모 세그먼트에서 다음 레벨로 내려오면서 가장 왼쪽 세그먼트를 우선 방문
  - 가장 빈번하게 접근하는 세그먼트를 트리 왼쪽에 배치하는 것이 효율적



# 논리적 데이터 모델 (계속)

## ■ 장점

- 1:n의 관계를 가지는 대용량의 데이터베이스 처리에 매우 강력한 방법
- 주어진 기능 내에서 찾고자 하는 데이터 항목을 빠르게 찾는 것이 가능

## ■ 단점

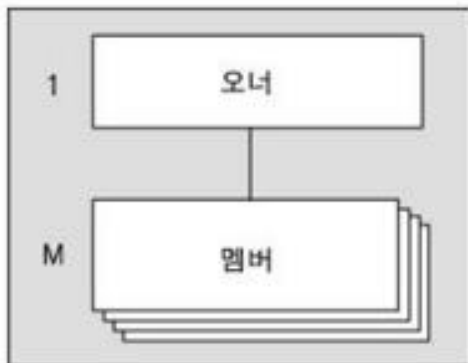
- 데이터 접근에 유연성이 부족함
  - 사용자의 측면에서 볼 때 데이터가 저장된 물리적 구조에 대한 지식 필요
  - 어떻게 데이터에 접근하는가를 미리 응용 프로그램에 정의하는 것이 필요
- 데이터들 간의 모든 관계가 데이터베이스가 처음 설계될 때 정해져야 함
- 레코드들이 링크로 연결되어 있으므로 레코드 구조를 변경하기 어려움
- 1:n의 기준에 맞지 않는 일반적인 관계는 구현하기가 매우 어려움



# 논리적 데이터 모델 (계속)

## ■ 네트워크 데이터 모델(Network Data Model)

- 데이터를 표현하는데 있어 제한적인 트리 구조가 아닌, **그래프 형태를 기반으로 현실 세계의 데이터 구조를 표현**하기 위한 필요성에 의해 만들어짐
- 1960년대 초에 Charles Bachman이 Honeywell사에서 최초의 네트워크 DBMS인 IDS를 개발
- 데이터 구조는 관계(Relationship)를 나타내는 집합(Set)을 기반으로 하며, **집합을 구성하는 요소들은 명시적 링크를 사용하여 오너(Owner)와 멤버(Member) 레코드 형으로 관계 표현**



# 논리적 데이터 모델 (계속)

- 오너와 멤버 레코드 형은 계층형 데이터 모델의 부모와 자식 세그먼트 같은 역할을 하며, 다중 부모/자식 관계를 허용
- 물리적 포인터를 사용해서 복잡한 오너/멤버 관계를 표현하므로 데이터에 대한 접근이 매우 효율적

# 논리적 데이터 모델 (계속)

## ■ 장점

- 한 멤버가 여러 개의 집합에 속할 수 있기 때문에 **다 대 다(N:M)의 관계가 계층형에 비해 쉽게 구현 가능**하고, 데이터에 대한 접근이 용이함

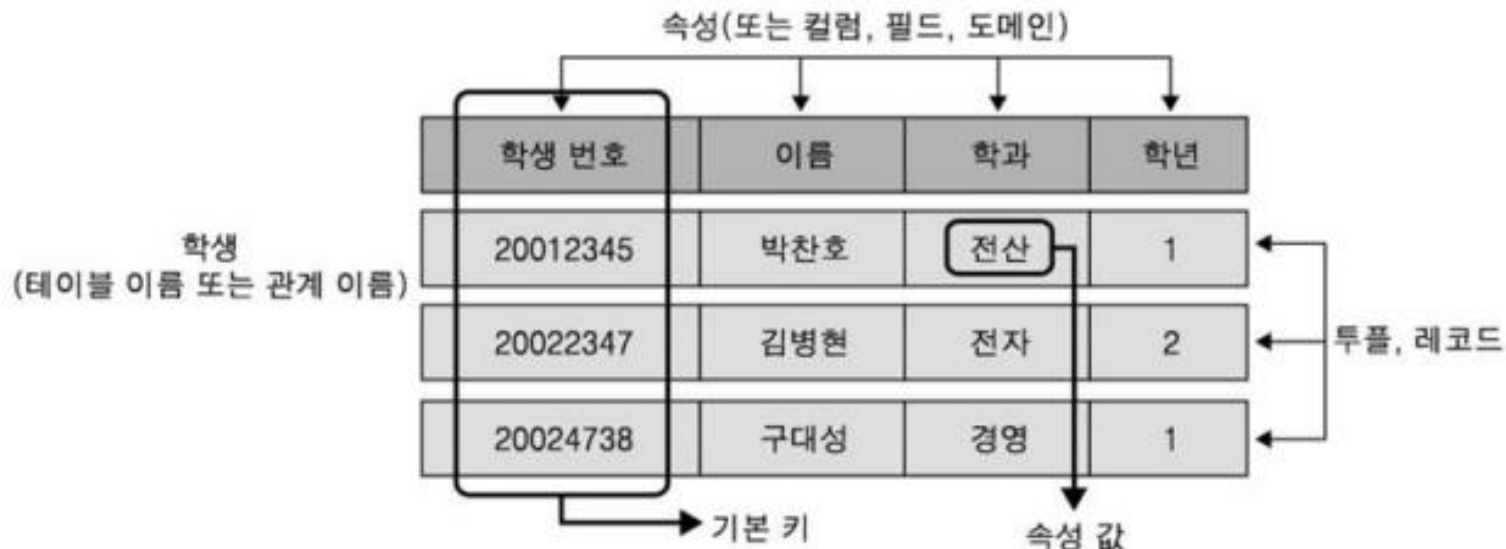
## ■ 단점

- **사용자가 데이터가 저장된 물리적인 구조에 대한 정보**를 가지고 있어야 함
- 원하는 데이터를 얻기 위해서는 **접근 경로를 항해**해야 함
- **데이터베이스의 구조를 변경**하면 이 구조를 참조한 **모든 응용 프로그램**을 수정해야 함

# 논리적 데이터 모델 (계속)

## ■ 관계 데이터 모형(Relational Data Model)

- 1970년 E. F. Codd에 의해 소개된 모델로, 단순하고 균일한 데이터 구조를 가지고 있어 많이 사용되고 있는 모델
- 개체의 모든 데이터와 데이터 사이의 관계를 **2차원의 테이블 (Table) 형태로 기술**
  - 테이블의 각 행(Row)은 하나의 개체를 나타냄
  - 테이블의 각 열(Column)은 개체의 각 속성을 표시



# 논리적 데이터 모델 (계속)

## ■ 장점

- 데이터 모델 구조가 탄력적이므로 필요할 때 테이블 사이의 연결을 통해 데이터를 생성 및 처리할 수 있음
- 데이터 정의 언어와 데이터 조작 언어가 간단하므로 비전문요원도 쉽게 사용할 수 있음
- 데이터들 간의 복잡한 관계를 개념적으로 분명하고 간단하게 표현하며, 강력한 데이터 조작 능력을 제공
- 데이터의 첨가, 삭제, 수정이 쉽고, 미래의 정보 요구에 신축성 있게 대응 가능

## ■ 단점

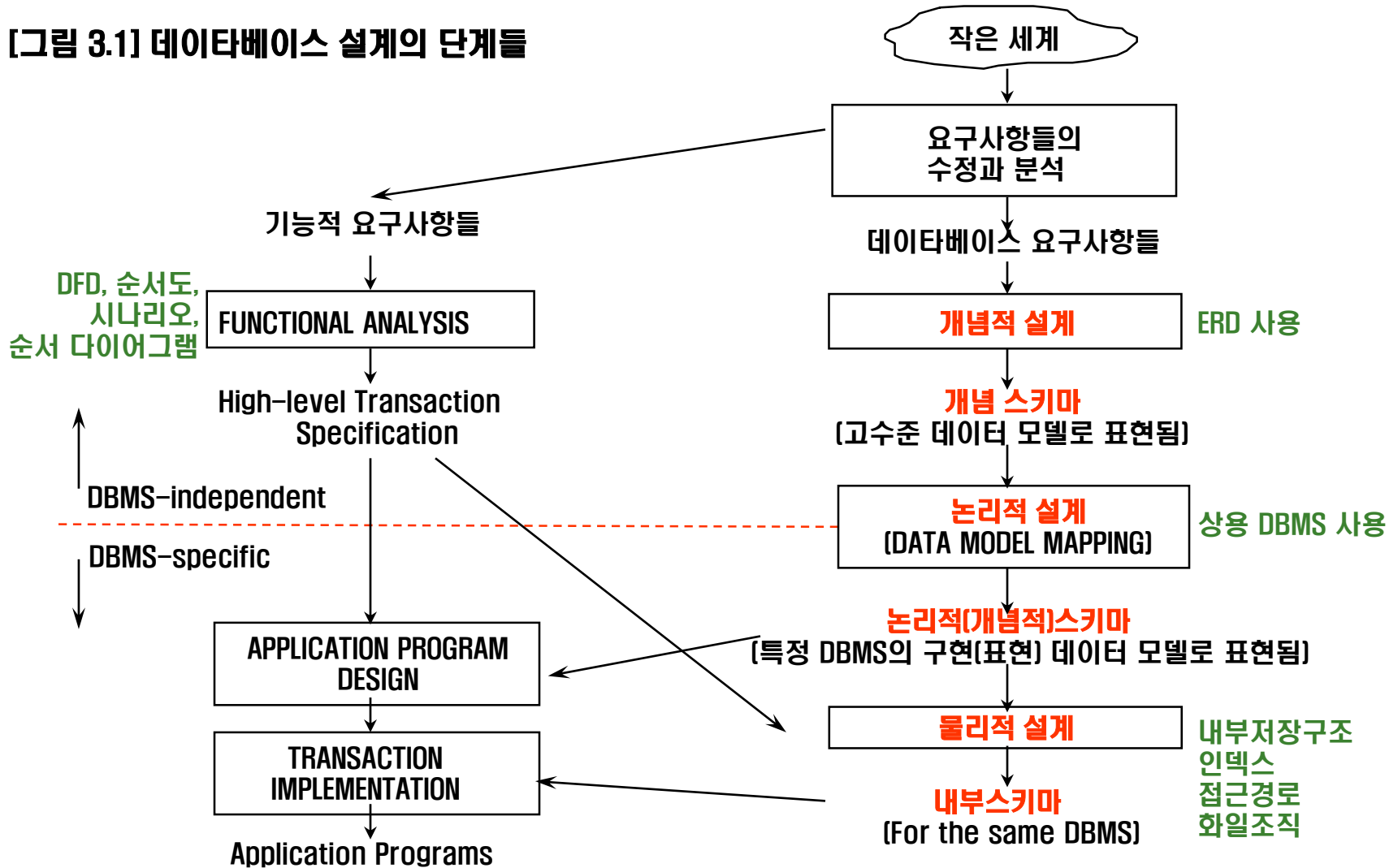
- 기억용량이 많이 필요
  - 각 테이블마다 인덱스가 수반
- 정보 추출에 시간이 많이 소요

# 감사합니다!

담당교수: 전강욱(컴퓨터공학부)

[kw.chon@koreatech.ac.kr](mailto:kw.chon@koreatech.ac.kr)

[그림 3.1] 데이터베이스 설계의 단계들



# 논리적 데이터 모델 (계속)

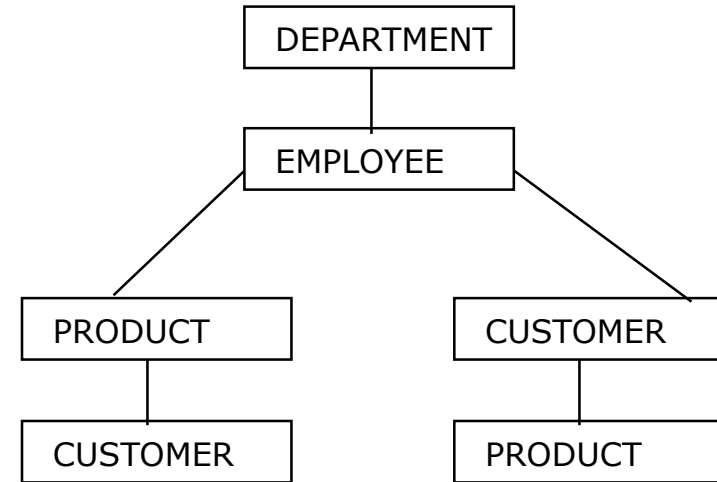
- 관계 데이터 모델 (다음 시간에 상세히 다룰 예정)
  - 일반적으로 많이 사용되는 논리적 데이터 모델
  - 데이터베이스의 논리적 구조가 2차원 테이블 형태



# 논리적 데이터 모델 (계속)

## ■ 계층 데이터 모델

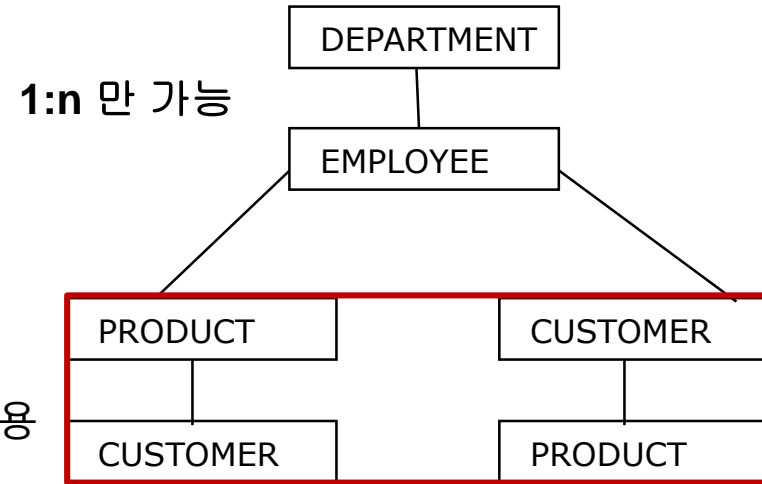
- 논리적 구조가 트리(tree) 형태
  - root 역할을 하는 개체 존재
  - 사이클이 없음
- 개체 간에 상하 관계가 성립
  - 부모 개체 / 자식 개체
  - 부모와 자식 개체는 일대다(1:n) 관계만 허용
- 두 개체 사이에 하나의 관계만 정의 가능
- 다 대 다(n:m) 관계를 직접 표현하는 것이 불가능
- 개념적 구조를 모델링하기 어려우며, 구조가 복잡
- 데이터 조작이 어려움
  - 삽입, 삭제, 수정, 검색



# 논리적 데이터 모델 (계속)

## ■ 계층 데이터 모델

- 논리적 구조가 트리(tree) 형태
  - root 역할을 하는 개체 존재
  - 사이클이 없음
- 개체 간에 상하 관계가 성립
  - 부모 개체 / 자식 개체
  - 부모와 자식 개체는 일대다(1:n) 관계만 허용
- 두 개체 사이에 하나의 관계만 정의 가능
- 다대다(n:m) 관계를 직접 표현하는 것이 불가능
- 개념적 구조를 모델링하기 어려우며, 구조가 복잡
- 데이터 조작이 어려움
  - 삽입, 삭제, 수정, 검색



같은 개체이지만  
따로 관계 정의 필요

# 논리적 데이터 모델 (계속)

## ■ 네트워크 데이터 모델 (network data model)

- 데이터베이스의 논리적 구조가 **그래프** 형태
  - 순환이 존재 가능함
- 개체 간 일대다(1:n) 관계만 허용
  - 오너(owner) / 멤버(member)
- 두 개체 사이에 여러 관계를 정의할 수 있어 이름으로 구별
- 다대다(n:m) 관계를 직접 표현할 수 없음
- 개념적 구조를 모델링하기 어려우며, 구조가 복잡 데이터 조작이 어려움 삽입, 삭제, 수정, 검색

