

# 데이터베이스

- 순서 : Ch5( 데이터베이스 설계와 ER모델)
- 학기 : 2018학년도 2학기
- 학과 : 가천대학교 컴퓨터공학과 2학년
- 교수 : 박양재

# 데이터베이스

- 목차

5.1 데이터베이스 설계의 개요

5.2 ER 모델

5.3 데이터베이스 설계 사례

5.4 논리적 설계: ER 스키마를 관계모델의 릴레이션으로 사상<sup>mapping</sup>

## 5장. 데이터베이스 설계와 ER 모델

### □ 데이터베이스 모델링

- ✓ 데이터 모델링은 관계 DBMS 세계에서 가장 어렵고, 가장 중요한 업무
- ✓ 모델링이 잘못되면 조직체의 응용은 사용자의 요구를 만족시키지 못할 수 있고, 신뢰할 수 없으며, 데이터베이스에는 쓸모없는 데이터를 채우게 된다.
- ✓ 한 조직체에 가용 가능한 모든 상세한 데이터를 모델에 기록하는 것은 불가능하며, 바람직하지도 않다.
- ✓ 데이터베이스 설계는 미리 정의된 응용들의 모임을 위해서 조직체의 사용자들의 정보 요구를 수용하여 하나 이상의 데이터베이스의 논리적인 구조와 물리적인 구조를 설계하는 것으로 서로 충돌되는 요구들을 조정하고, 좋은 구조를 갖는 데이터베이스를 구축하고, 응답 시간과 저장공간 등의 처리 요구 사항을 만족시키도록 한다.
- ✓ 하나의 데이터베이스는 실세계의 어떤 측면(작은 세계)을 나타낸다. 작은 세계에서 일어나는 변화는 데이터베이스에 반영된다.

## 5장. 데이터베이스 설계와 ER 모델

### □ 데이터베이스 설계

- ✓ 개념적 데이터베이스 설계와 물리적 데이터베이스 설계로 구분
- ✓ 개념적 데이터베이스 설계는 실제로 데이터베이스를 어떻게 구현할 것인가와는 독립적으로 정보 사용의 모델을 개발하는 과정
- ✓ 물리적 데이터베이스 설계에서는 물리적인 저장 장치와 접근 방식을 다룸
- ✓ 개념적 데이터베이스 설계 과정에서 조직체(실세계)의 엔티티, 관계, 프로세스, 무결성 제약조건 등을 나타내는 추상화 모델을 구축
- ✓ 엔티티는 서로 구분이 되면서 조직체에서 데이터베이스에 나타내려는 객체(사람, 장소, 사물 등)를 의미
- ✓ 관계는 두 개 이상의 엔티티들 간의 연관을 나타냄
- ✓ 프로세스는 관련된 활동을 나타냄
- ✓ 무결성 제약조건은 데이터의 정확성과 비즈니스 규칙을 의미
- ✓ 추상화 모델이 구축되면 물리적 구현을 고려하지 않은 한 조직체의 개념적 스키마

## 5장. 데이터베이스 설계와 ER 모델

### □ 데이터베이스 설계

- ✓ 개념적 데이터베이스 설계를 수행하는 이유
  - 비즈니스를 좀 더 이해하기 위해서
  - 최종 사용자와 의사 소통을 가능하게 하기 위해서
  - 설계 상의 실수를 초기에 발견하기 위해서
  - 튼튼한 기초를 구축하기 위해서
  - 데이터베이스 품질을 보정하기 위해서
  - DBMS와 독립적인 데이터베이스를 설계하기 위해서

## 5장. 데이터베이스 설계와 ER 모델(계속)

### □ 개념적 수준의 모델

- ✓ 특정 데이터 모델과 독립적으로 응용 세계를 모델링할 수 있도록 함
- ✓ 모델링 되고 있는 환경, 사용자 필요성, 정보 요구사항 등의 변경시 데이터베이스의 구조가 진화할 수 있도록 허용하는 편리한 메커니즘
- ✓ 데이터베이스 구조나 스키마를 하향식으로 개발할 수 있기 위한 틀(framework)을 제공함
- ✓ 인기 있는 개념적 수준의 모델은 **엔티티-관계(ER: Entity-Relationship)** 모델  
ERD(ER - Diagram) DFD(Data Flow Diagram - )
- ✓ ER 모델과 같은 개념적인 데이터 모델이 사상될 수 있는 다수의 **구현 데이터 모델**(implementation data model)이 존재함
- ✓ 구현 단계에서 사용되는 세 가지 데이터 모델: 관계 데이터 모델, 계층 데이터 모델, 네트워크 데이터 모델

## 5.1 데이터베이스 설계의 개요

### □ 데이터베이스 설계의 개요

- ✓ 한 조직체의 운영과 목적을 지원하기 위해 데이터베이스를 생성하는 과정
- ✓ 목적은 모든 주요 응용과 사용자들이 요구하는 데이터, 데이터 간의 관계를 표현하는 것
- ✓ 데이터베이스 개발은 일반적인 프로젝트 라이프 사이클 과정을 따름
- ✓ 훌륭한 데이터베이스 설계는 시간의 흐름에 따른 데이터의 모든 측면을 나타내고, 데이터 항목의 중복을 최소화하고, 데이터베이스에 대한 효율적인 접근을 제공하고, 데이터베이스의 무결성을 제공하고, 이해하기 쉬워야 함

## 5.1 데이터베이스 설계의 개요(계속)

〈표 5.1〉 데이터베이스 개발의 라이프 사이클

데이터베이스 설계 과정		
단계	기능	질문
요구사항 분석 단계		
	요구사항 수집과 분석	
설계 단계		
개념적 설계	ER 모델링 또는 객체 지향 모델	어떤 엔티티와 관계들이 요구되는가?
DBMS의 선정		어떤 DBMS가 적절한가?
논리적 설계	개념적 설계를 데이터베이스 스키마로 사상	데이터베이스가 무엇을 모델링하는가?
스키마 정제	중복을 제거함 - 데이터베이스 스키마의 정규화	가장 단순한 스키마?
물리적 설계	성능상의 문제를 고려하여 인덱스 등을 정의	어떤 성능을 원하는가?
보안 설계	사용자들의 그룹과 접근 제한	어떤 수준의 보안을 원하는가?
구현 단계		
	데이터베이스의 구축과 튜닝	
데이터를 적재하거나 변환		
기존의 응용 변환		

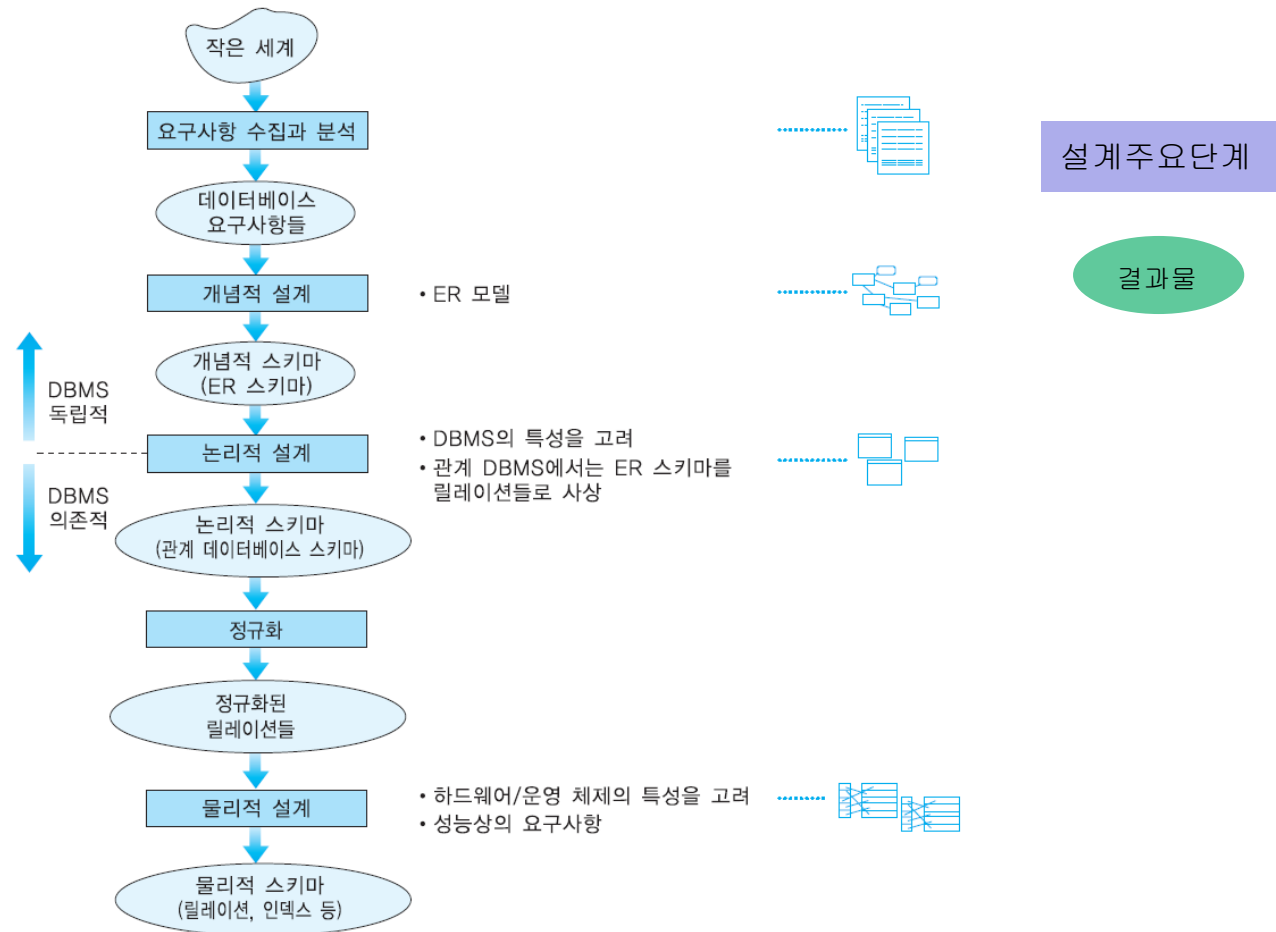


## 5.1 데이터베이스 설계의 개요(계속)

### □ 데이터베이스 설계의 주요 단계

- ✓ 요구사항 분석, 개념적 설계, DBMS의 선정, 논리적 설계, 스키마 정제, 물리적 설계와 튜닝 등 여러 작업들로 이루어짐
- ✓ 일반적으로, 데이터베이스 설계의 완성도를 높이기 위해서 이런 작업들을 앞뒤로 왔다갔다할 필요가 있음

## 5.1 데이터베이스 설계의 개요(계속)



[그림 5.1] 데이터베이스 설계의 주요 단계

## 5.1 데이터베이스 설계의 개요(계속)

### □ 요구사항 수집과 분석

- ✓ 흔히 기존의 문서를 조사하고, 인터뷰나 설문 조사 등이 시행됨
- ✓ 인터뷰는 요구사항 수집을 위해 가장 흔히 사용됨
- ✓ 설문 조사는 자유롭게 의견을 적어내도록 하는 방식과 주어진 질문에 대해서만 답을 하는 방식으로 구분
- ✓ 요구사항에 관한 지식을 기반으로 관련 있는 엔티티들과 이들의 애트리뷰트들이 무엇인가, 엔티티들 간의 관계가 무엇인가 등을 파악함
- ✓ 또한 데이터 처리에 관한 요구사항에 대하여 전형적인 연산들은 무엇인가, 연산들의 의미, 접근하는 데이터의 양 등을 분석함
- ✓ 설계자와 사용간의 원활한 의사소통이 목적

## 5.1 데이터베이스 설계의 개요(계속)

### □ 개념적 설계

- ✓ 모든 물리적인 사항과 독립적으로, 한 조직체에서 사용되는 정보의 모델을 구축하는 과정
- ✓ 사용자들의 요구사항 명세로부터 개념적 스키마가 만들어짐
- ✓ 높은 추상화 수준의 데이터 모델을 기반으로 정형적인 언어로 데이터 구조를 명시함
- ✓ 대표적인 데이터 모델이 ER 모델
- ✓ 개념적 설계의 단계에서는 엔티티 타입, 관계 타입, 애트리뷰트들을 식별하고, 애트리뷰트들의 도메인을 결정하고, 후보 키와 기본 키 애트리뷰트들을 결정함
- ✓ 완성된 개념적 스키마(ER 스키마)는 ER 다이어그램으로 표현됨

## 5.1 데이터베이스 설계의 개요(계속)

### □ DBMS 선정<sup>DBA</sup>

- ✓ 여러 가지 요인들을 검토한 후 DBMS를 선정함
- ✓ 기술적인 요인은 DBMS가 제공하는 데이터 모델, 저장 구조, 인터페이스, 질의어, 도구, 제공되는 서비스 등
- ✓ 정치적인 요인은 고수준의 전략적인 결정 등
- ✓ 경제적인 요인은 DBMS 구입 비용, 하드웨어 구입 비용, 유지 보수(서비스) 비용, 기존의 시스템을 새로운 DBMS에 맞게 변환하는데 소요되는 비용, 인건비, 교육비 등

## 5.1 데이터베이스 설계의 개요(계속)

### □ 논리적 설계

- ✓ 데이터베이스 관리를 위해 선택한 DBMS의 데이터 모델을 사용하여 논리적 스키마(외부 스키마도 포함)를 생성함
- ✓ 개념적 스키마에 알고리즘을 적용하여 논리적 스키마를 생성함
- ✓ 논리적 스키마를 나타내기 위해 관계 데이터 모델을 사용하는 경우에는, ER 모델로 표현된 개념적 스키마를 관계 데이터베이스 스키마로 사상함
- ✓ 관계 데이터베이스 스키마를 더 좋은 관계 데이터베이스 스키마로 변환하기 위해서 정규화 과정(관계 스키마에 중복과 갱신 이상이 발생하는지 검사)을 적용함
- ✓ 데이터베이스 설계자가 요구사항 수집과 분석 후에 바로 논리적 설계 단계로 가는 경우가 있는데, 이런 경우에는 흔히 좋은 관계 데이터베이스 스키마가 생성되지 않음

## 5.1 데이터베이스 설계의 개요(계속)

### □ 물리적 설계

- ✓ 처리 요구사항들을 만족시키기 위해 **저장 구조와 접근 경로 등을 결정함**
- ✓ 물리적 설계에 영향을 미치는 요소: 트랜잭션들의 예상 수행 빈도, 트랜잭션들의 시간 제약조건
- ✓ **성능상의 주요 기준은 몇 가지로 구분할 수 있음**
  - **응답 시간**: 질의와 갱신이 평균적으로 또는 피크 시간 때 얼마나 오래 걸릴 것인가?
  - **트랜잭션 처리율**: 1초당 얼마나 많은 트랜잭션들이 평균적으로 또는 피크 시간 때 처리될 수 있는가?
  - **전체 데이터베이스에 대한 보고서를 생성하는데** 얼마나 오래 걸릴 것인가?

## 5.1 데이터베이스 설계의 개요(계속)

### □ 트랜잭션 설계

- ✓ 트랜잭션은 완성될 데이터베이스에서 동작할 응용 프로그램
- ✓ 요구사항 수집과 분석 후에 데이터베이스 설계 과정과 별도로 트랜잭션 설계를 진행할 수 있음
- ✓ 데이터베이스 스키마는 트랜잭션에서 요구하는 모든 정보를 포함해야 함
- ✓ 검색, 갱신, 혼합 등 세 가지 유형으로 구분하여 입력과 출력, 동작 등을 식별함
- ✓ 데이터베이스 설계의 최종결과는 중요한 설계 결정이 문서화된 완전히 동작하는 데이터베이스이다.



## 5.2 ER 모델

### □ ER (Entity Relationship) 모델

- ✓ 데이터베이스 설계를 용이하게 하기 위해서 **P.P. Chen**이 1976년에 제안하였음
- ✓ 그 후에 많은 학자들이 이 모델을 강화시켰음
- ✓ 현재는 **EER**(Enhanced Entity Relationship) 모델이 데이터베이스 설계 과정에 널리 사용되고 있음
- ✓ 개념적 설계를 위한 인기 있는 모델로서, 많은 CASE 도구들에서 지원됨
- ✓ 실세계를 엔티티, 애트리뷰트, 엔티티들 간의 관계로 표현함
- ✓ 쉽게 관계 데이터 모델로 사상됨

Computer Aided SE

CASE  
1DDL

2

CASE

## 5.2 ER 모델(계속)

### □ ER 모델(계속)

- ✓ 기본적인 구문으로는 엔티티, 관계, 애트리뷰트가 있고, 기타 구문으로는 카디날리티 비율, 참여 제약조건 등이 있음
- ✓ 적은 노력으로 쉽게 배울 수 있고, 전문가가 아니어도 이해하기 쉬우며, 자연어보다는 좀더 정형적이고, 구현에 독립적이어서 데이터베이스 설계자들이 최종 사용자와 의사 소통을 하는데 적합함
- ✓ ER 모델을 기반으로 만들어진 다수의 CASE 도구(예, **ERWin**, **DataArchitect**, **PowerBuilder** 등)들이 존재함
- ✓ 이런 도구들은 ER 설계를 자동적으로 오라클, SQL Server, 사이베이스 등의 데이터 정의어로 변환하고, 어떤 도구는 XML로 변환함
- ✓ 현재는 데이터베이스 설계를 위한 다소 구형 그래픽 표기법→UML(Unified Modeling Language) :통합 모델링 언어, 객체지향분석/설계용 모델링 언어(대기업 데이터베이스 설계)

## 5.2 ER 모델(계속)

### □ 엔티티(Entity)

- ✓ 하나의 엔티티는 사람, 장소, 사물, 사건 등과 같이 **독립적으로 존재하면서 고유하게 식별이 가능한 실세계의 객체**
- ✓ 사원처럼 실체가 있는 것도 있지만 생각이나 개념과 같이 추상적인 것도 있음



[그림 5.2] 엔티티의 예

## 5.2 ER 모델(계속)

### □ 엔티티 타입

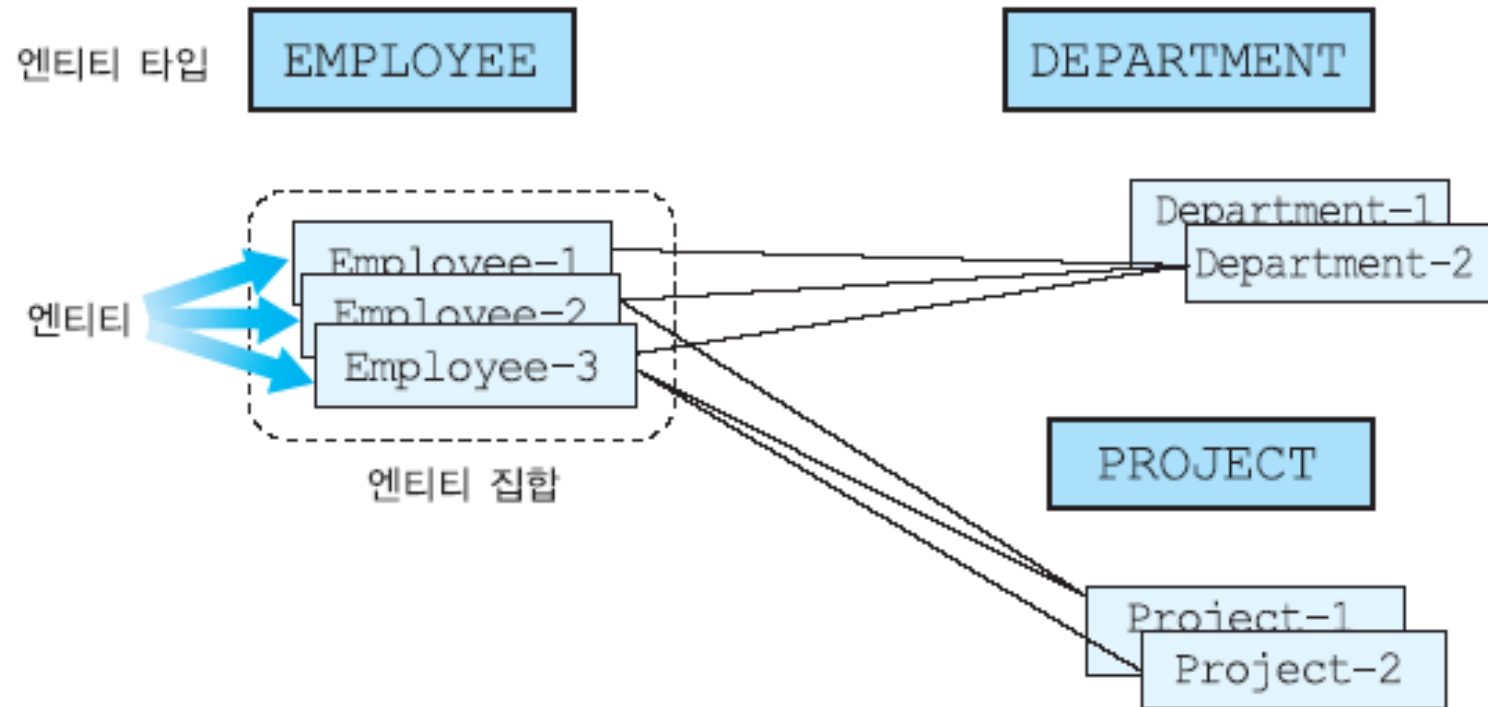
- ✓ 엔티티들은 엔티티 타입(또는 엔티티 집합)들로 분류됨
- ✓ 엔티티 타입은 동일한 애트리뷰트들을 가진 엔티티들의 틀
- ✓ 엔티티 집합은 동일한 애트리뷰트들을 가진 엔티티들의 모임
- ✓ 하나의 엔티티는 한 개 이상의 엔티티 집합에 속할 수 있음
- ✓ 엔티티 타입은 관계 모델의 릴레이션의 내포(intension)에 해당하고, 엔티티 집합은 관계 모델의 릴레이션의 외연(extension)에 해당함

EMPLOYEE	EMPNO	EMPNAME	TITLE	DNO	SALARY	내포
	2106	김창섭	대리	2	2000000	외연
	3426	박영권	과장	3	2500000	
	3011	이수민	부장	1	3000000	
	1003	조민희	대리	1	2000000	

[그림 2.3] 릴레이션의 내포와 외연

- ✓ 엔티티 집합과 엔티티 타입을 엄격하게 구분할 필요는 없음
- ✓ ER 다이어그램에서 엔티티 타입은 직사각형으로 나타냄

## 5.2 ER 모델(계속)



[그림 5.3] 엔티티, 엔티티 타입, 엔티티 집합

## 5.2 ER 모델(계속)

### □ 강한 엔티티 타입

- ✓ 강한 엔티티 타입(정규 엔티티 타입)은 독자적으로 존재하며 엔티티 타입 내에서 자신의 키 애트리뷰트를 사용하여 고유하게 엔티티들을 식별할 수 있는 엔티티 타입

### □ 약한 엔티티 타입

- ✓ 약한 엔티티 타입은 키를 형성하기에 충분한 애트리뷰트들을 갖지 못한 엔티티 타입
- ✓ 이 엔티티 타입이 존재하려면 소유 엔티티 타입이 있어야 함
- ✓ 소유 엔티티 타입의 키 애트리뷰트를 결합해야만 고유하게 약한 엔티티 타입의 엔티티들을 식별할 수 있음

## 5.2 ER 모델(계속)

### □ 애트리뷰트

- ✓ 하나의 엔티티는 연관된 애트리뷰트들의 집합으로 설명됨
  - 예: 사원 엔티티는 사원번호, 이름, 직책, 급여 등의 애트리뷰트를 가짐
- ✓ 한 애트리뷰트의 도메인은 그 애트리뷰트가 가질 수 있는 모든 가능한 값들의 집합을 의미
  - 예: 사원번호는 1000부터 9999까지의 값을 가짐
- ✓ 여러 애트리뷰트가 동일한 도메인을 공유할 수 있음
  - 예: 사원번호와 부서번호가 네 자리 정수를 가질 수 있음
- ✓ 키 애트리뷰트는 한 애트리뷰트 또는 애트리뷰트들의 모임으로서 한 엔티티 타입 내에서 각 엔티티를 고유하게 식별함
- ✓ 키로 사용하기 적합한 애트리뷰트가 없을 경우-인위적으로 기본키 엔티티타입을 추가한다(예: 순번, NO...)
- ✓ ER 다이어그램에서 기본 키에 속하는 애트리뷰트는 밑줄을 그어 표시함

## 5.2 ER 모델(계속)

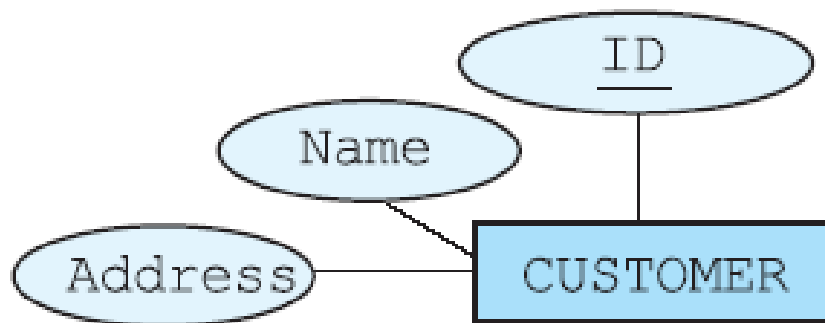
### □ 애트리뷰트(계속)

- ✓ 요구사항 명세에서 명사나 형용사로 표현됨
- ✓ 엔티티는 독립적인 의미를 갖는데 반해서 애트리뷰트는 독립적인 의미를 갖지 않음
- ✓ ER 다이어그램에서 타원형으로 나타냄
- ✓ 애트리뷰트와 엔티티 타입은 실선으로 연결



## 5.2 ER 모델(계속)

- ❑ **단순 애트리뷰트**(simple attribute)
  - ✓ 더 이상 다른 애트리뷰트로 나눌 수 없는 애트리뷰트
  - ✓ ER 다이어그램에서 실선 타원으로 표현함
  - ✓ ER 다이어그램에서 대부분의 애트리뷰트는 단순 애트리뷰트

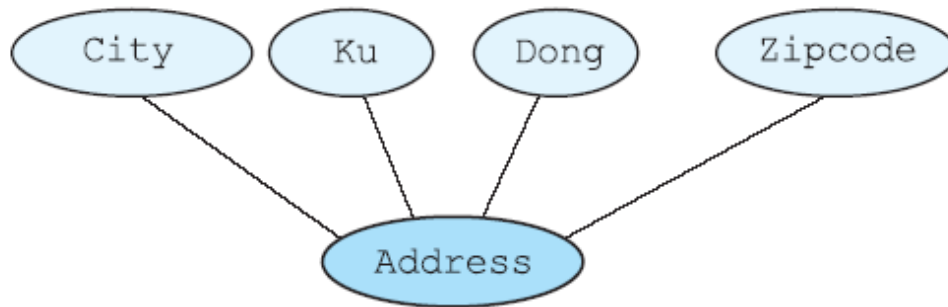


[그림 5.4] 단순 애트리뷰트

## 5.2 ER 모델(계속)

### ❑ 복합 애트리뷰트(composite attribute)

- ✓ 두 개 이상의 애트리뷰트로 이루어진 애트리뷰트
- ✓ 동일한 엔티티 타입이나 관계 타입에 속하는 애트리뷰트들 중에서 밀접하게 연관된 것을 모아놓은 것



[그림 5.5] 복합 애트리뷰트

- ✓ Why? : 주소 전체를 하나의 단순 애트리뷰트로 지정하면 하나의 문자열로 저장되므로 거주지역별 소비패턴분석과 , 광고효과 등이 어렵다.
- ✓ 단순한 주소 사용시는 단순 애트리뷰트 적용

## 5.2 ER 모델(계속)

- ❑ 단일 값 애트리뷰트(single-valued attribute)
  - ✓ 각 엔티티마다 정확하게 하나의 값을 갖는 애트리뷰트  
즉, 값들의 집합이나 리스트를 갖지 않는 애트리뷰트
  - ✓ ER 다이어그램에서 단순 애트리뷰트와 동일하게 표현됨
  - ✓ 예: 사원의 사원번호 애트리뷰트는 어떤 사원도 두 개 이상의 사원번호를 갖지 않으므로 단일 값 애트리뷰트
  - ✓ ER 다이어그램에서 대부분의 애트리뷰트는 단일 값 애트리뷰트

## 5.2 ER 모델(계속)

- ❑ 다치 애트리뷰트(multi-valued attribute)
  - ✓ 각 엔티티마다 여러 개의 값을 가질 수 있는 애트리뷰트
  - ✓ ER 다이어그램에서 이중선 타원으로 표현함
  - ✓ 사원은 여러 개의 취미를 가질 수 있으므로



[그림 5.6] 다치 애트리뷰트

## 5.2 ER 모델(계속)

- ❑ **저장된 애트리뷰트(stored attribute)**
  - ✓ 다른 애트리뷰트와 독립적으로 존재하는 애트리뷰트
  - ✓ ER 다이어그램에서 **단순 애트리뷰트와 동일하게 표현됨**
  - ✓ ER 다이어그램에서 **대부분의 애트리뷰트는 저장된 애트리뷰트**
  - ✓ 예: 사원 엔티티 타입에서 사원이름, 급여는 다른 애트리뷰트와 독립적으로 존재함

## 5.2 ER 모델(계속)

- ❑ 유도된 애트리뷰트(derived attribute)
  - ✓ 다른 애트리뷰트의 값으로부터 얻어진 애트리뷰트
  - ✓ 유도된 애트리뷰트는 반드시 필요한 애트리뷰트가 아니고, 데이터의 불일치를 유발할 수 있으므로, 관계 데이터베이스에서 릴레이션의 애트리뷰트로 포함시키지 않는 것이 좋음
  - ✓ ER 다이어그램에서 점선 타원으로 표현함
  - ✓ Age 애트리뷰트는 주민등록번호로 부터 유도될 수 있는 애트리뷰트

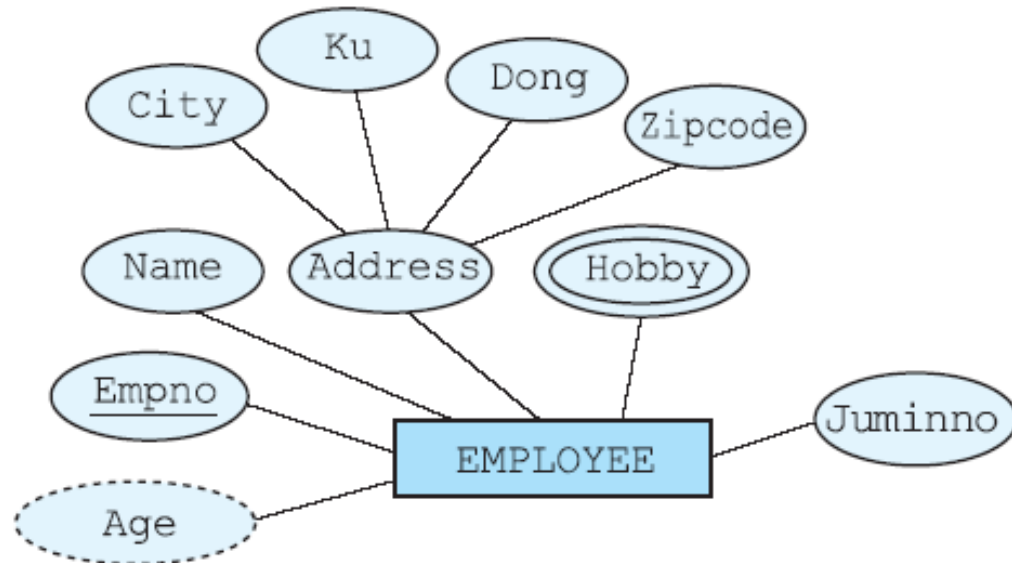


[그림 5.7] 유도된 애트리뷰트

## 5.2 ER 모델(계속)

### 예 : 애트리뷰트들의 유형

아래 그림 5.8에서 단순 애트리뷰트, 복합 애트리뷰트, 단일 값 애트리뷰트, 다치 애트리뷰트, 키 애트리뷰트, 저장된 애트리뷰트, 유도된 애트리뷰트들을 구분하라.



[그림 5.8] 여러 가지 애트리뷰트의 예

## 5.2 ER 모델(계속)

### □ 약한 엔티티 타입

- ✓ 엔티티 타입 내의 엔티티들이 자체적으로 갖고 있는 애트리뷰트들의 값에 의해서 고유하게 식별이 안되는 경우
- ✓ 키를 형성하기에 충분한 애트리뷰트들을 갖지 못한 엔티티 타입
- ✓ 예:부양가족의 이름이 다른 사원 부양가족이름과 같을 수 있다.
- ✓ 해결책 1: 모든 부양가족에게 고유번호 부여?
- ✓ 해결책2: **부양가족이 속한 사원번호와 부양가족의 이름을 결합하여 모든 사원들의 부양가족을 고유하게 식별 가능**
- ✓ 약한 엔티티 타입에게 키 애트리뷰트를 제공하는 엔티티 타입을 **소유 엔티티 타입(owner entity type)** 또는 **식별 엔티티 타입(identifying entity type)**라고 부름(예:사원번호)

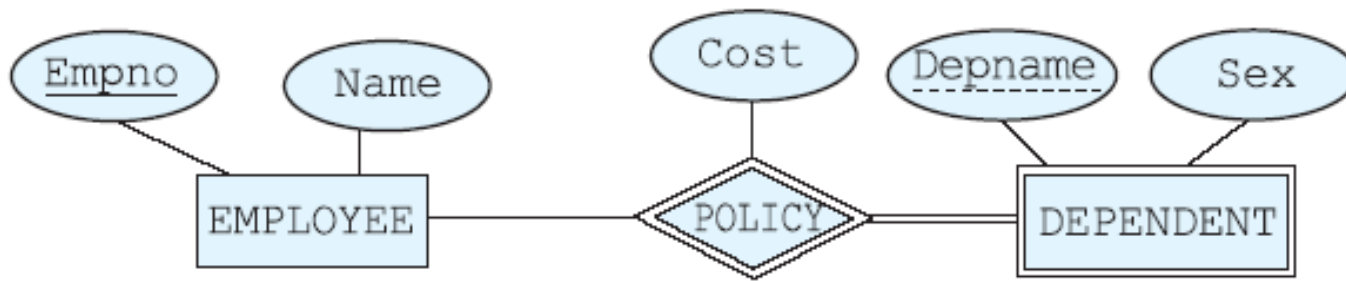


## 5.2 ER 모델(계속)

### □ 약한 엔티티 타입

- ✓ ER 다이어그램에서 이중선 직사각형으로 표기
- ✓ 약한 엔티티 타입의 부분 키는 점선 밑줄을 그어 표시
- ✓ 부분 키(partial key): 부양가족의 이름처럼 한 사원에 속한 부양가족 내에서는 서로 다르지만 회사 전체 사원들의 부양가족들 전체에서는 같은 경우가 생길 수 있는 애트리뷰트
- ✓ 기본 키를 가진 엔티티 강한 엔티티 타입 또는 정규 엔티티 타입
- ✓ 강한 관계 : 강한 엔티티 사이의 관계, 강한 엔티티 타입과 약한 엔티티 타입을 연결하는 관계는 약한 관계
- ✓ 약한 엔티티는 다른 엔티티의 존재에 의존하므로 종속되는 엔티티라 한다.
- ✓ 이를 의존 종속성(existence dependence)라 한다.

## 5.2 ER 모델(계속)



[그림 5.9] 약한 엔티티 타입

- 강한 엔티티 타입 EMPLOYEE와 약한 엔티티 타입 DEPENDENT를 POLICY(보험) 관계타입으로 연결
- 약한 엔티티 부분키(depname)
- 소유 엔티티(EMPLOYEE)
- 약한 관계 타입

## 5.2 ER 모델(계속)

### □ 관계와 관계 타입

- ✓ 하나의 엔티티 자체는 다른 어떤 엔티티와 연관을 가질 수 있다.
- ✓ 예: **사원 엔티티는 부서 엔티티에서 일한다.** 이를 엔티티들간의 관계라한다.
- ✓ 관계는 엔티티들 사이에 존재하는 **연관이나 연결로서** 두 개 이상의 엔티티 타입들 사이의 사상으로 생각할 수 있음
- ✓ 관계 집합은 동질의 관계들의 집합
- ✓ 관계 타입은 동질의 관계들의 틀
- ✓ 관계 집합과 관계 타입을 엄격하게 구분할 필요는 없음
- ✓ 요구사항 명세에서 흔히 **동사는 ER 다이어그램에서 관계로 표현됨**
- ✓ ER 다이어그램에서 **다이어몬드로** 표기
- ✓ 관계 타입이 서로 연관시키는 엔티티 타입들을 관계 타입에 실선으로 연결함

## 5.2 ER 모델(계속)



[그림 5.10] 관계 타입 WORKS\_FOR

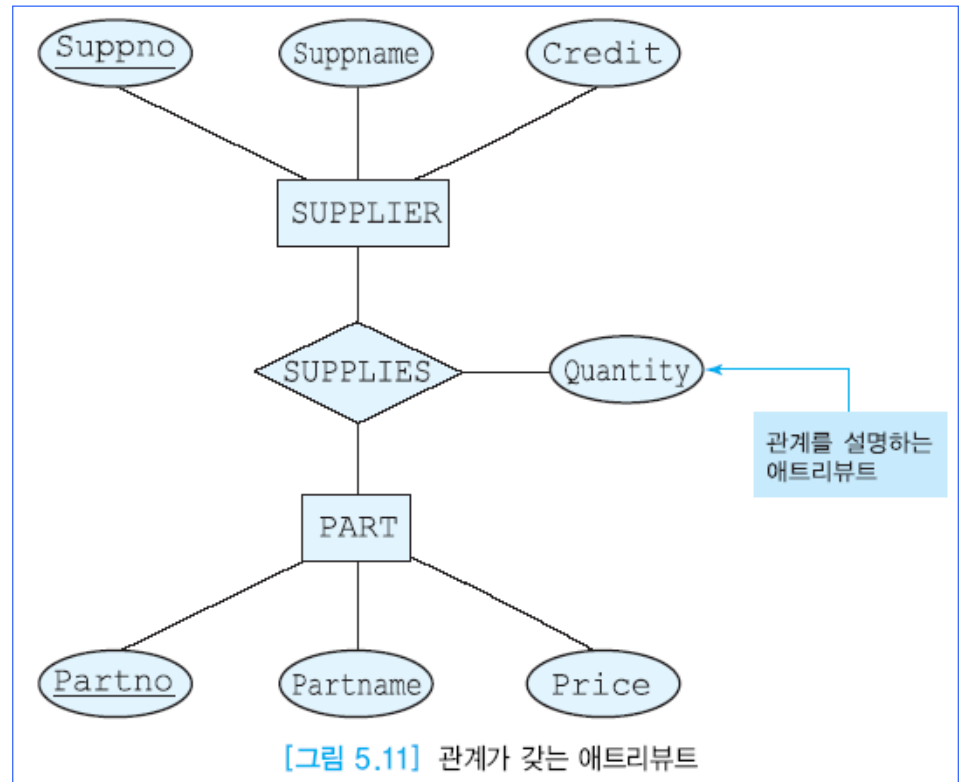
〈표 5.2〉 엔티티와 엔티티 간의 관계의 예

엔티티	관계	엔티티
사원(employee)	근무한다(works for)	부서(department)
공급자(supplier)	공급한다(supplies)	부품(part)
학생(student)	수강한다(enrolls)	과목(course)

## 5.2 ER 모델(계속)

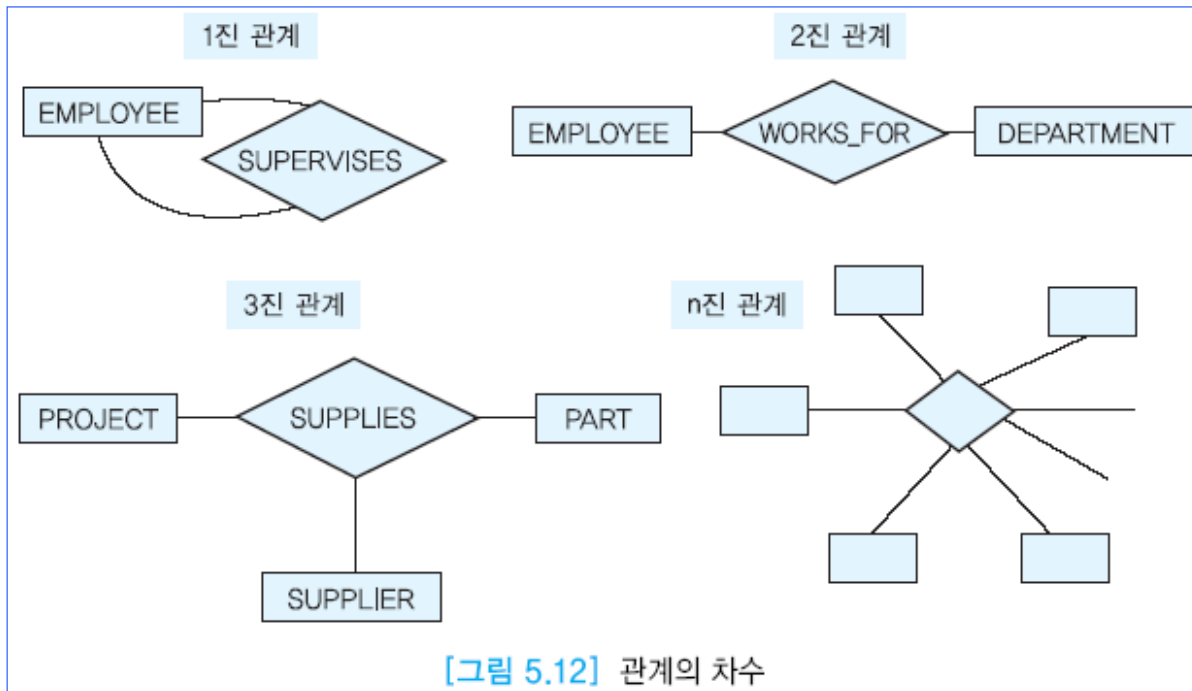
### □ 관계의 애트리뷰트

- ✓ 관계 타입은 **관계의 특징을 기술하는 애트리뷰트들을** 가질 수 있음
- ✓ 관계 타입은 **키 애트리뷰트를 갖지 않음**
- ✓ 수량 애트리뷰트는 어떤 공급자가 어떤 부품을 얼마나 공급하는가 설명
- ✓ 수량 애트리뷰트를 공급자와 부품 엔티티에는 붙일 수 없다.



## 5.2 ER 모델(계속)

- ❑ 차수(degree)=열(컬럼)의 개수, 애트리뷰트의 갯수
  - ✓ 관계로 연결된 엔티티 타입들의 개수를 의미
  - ✓ 실세계에서 가장 흔한 관계는 두 개의 엔티티 타입을 연결하는 2진 관계

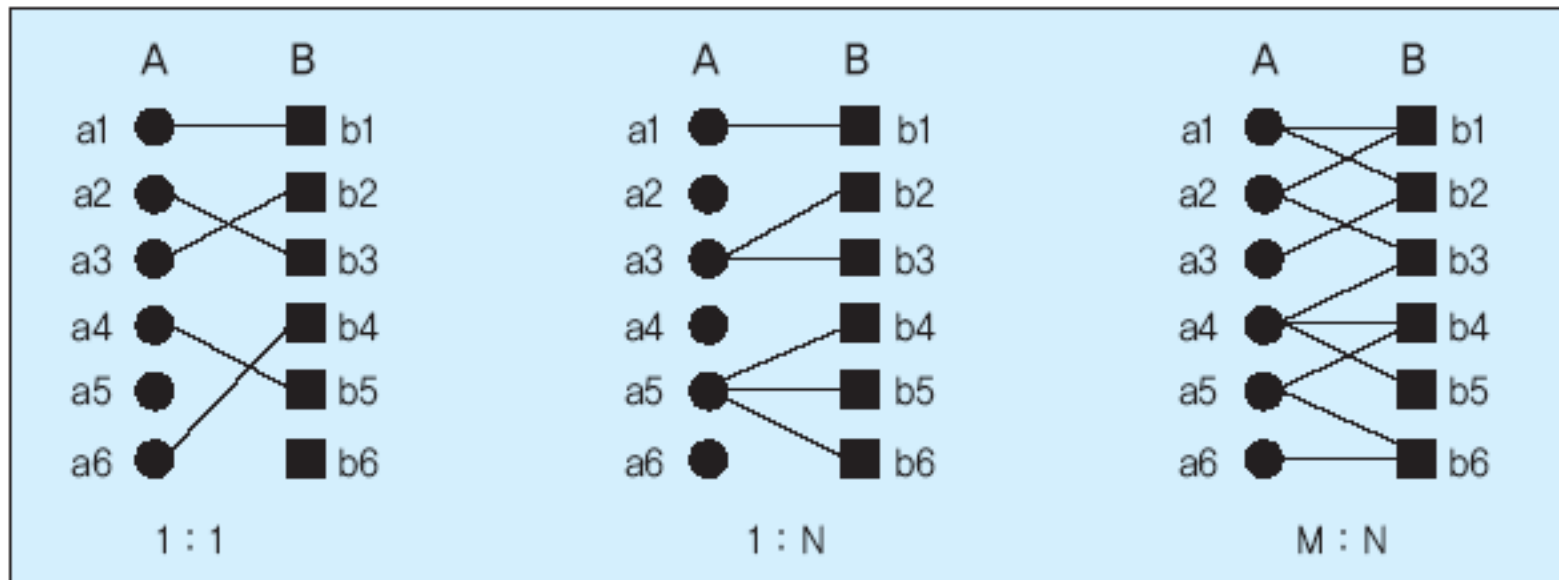


## 5.2 ER 모델(계속)

### □ 카디널리티(행들의 수, 대응수)

- ✓ 데이터 모델링시, 두개의 엔티티 타입이 서로 연관된다는 사실만 알아서는 충분하지 않다.
- ✓ 두 엔티티 타입 간의 카디널리티 비율도 정확히 모델링 되어야 한다.
- ✓ 카디널리티 비율은 **한 엔티티 타입의 몇 개의 엔티티가 다른 엔티티 타입의 몇 개의 엔티티와 연관 되는가?**(한 엔티티가 참여할 수 있는 관계의 수를 나타냄)
- ✓ 관계 타입에 참여하는 엔티티들의 가능한 조합을 제한함
- ✓ 관계를 흔히 1:1, 1:N, M:N으로 구분
- ✓ 카디널리티에 관한 정보는 간선 위에 나타냄

## 5.2 ER 모델(계속)



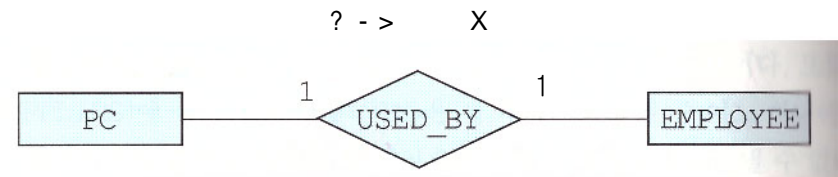
[그림 5.14] 카디날리티 비율



## 5.2 ER 모델(계속)

### □ 1:1 관계

- ✓ E1의 각 엔티티가 정확하게 E2의 한 엔티티와 연관되고, E2의 각 엔티티가 정확하게 E1의 한 엔티티와 연관되면 이 관계를 1:1 관계라고 함
- ✓ 예: 각 사원에 대해 최대한 한 개의 PC가 있고, 각 PC에 대해 **최대한 한 명의 사원이** 있으면 사원과 PC 간의 관계는 1:1 관계



### □ 1:N 관계

- ✓ E1의 각 엔티티가 E2의 임의의 개수의 엔티티와 연관되고, E2의 각 엔티티는 정확하게 E1의 한 엔티티와 연관되면 이 관계를 1:N 관계라고 함
- ✓ 예: 각 사원에 대해 최대한 한 대의 PC가 있고, 각 PC에 대해 **여러 명의 사원들이** 있으면 PC와 사원 간의 관계는 1:N 관계
- ✓ **실세계에서 가장 흔히 나타나는 관계**



## 5.2 ER 모델(계속)

### □ M:N 관계

- ✓ 한 엔티티 타입에 속하는 임의의 개수의 엔티티가 다른 엔티티 타입에 속하는 임의의 개수의 엔티티와 연관됨
- ✓ 예: 각 사원에 대해 여러 대의 PC가 있고, 각 PC에 대해 여러 명의 사원들이 있으면 사원과 PC 간의 관계는 M:N 관계

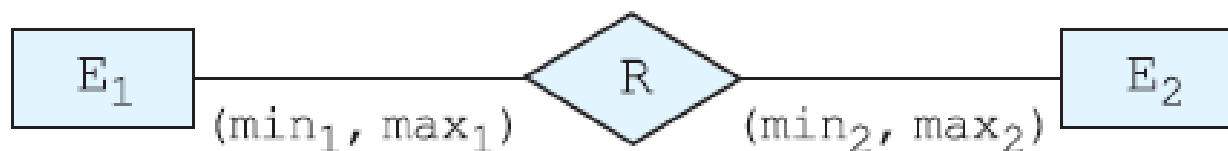


## 5.2 ER 모델(계속)

### □ 카디널리티 비율의 최소값과 최대값

- ✓ ER 다이어그램에서 관계 타입과 엔티티 타입을 연결하는 실선 위에 (min, max) 형태로 표기
- ✓ 어떤 관계 타입에 참여하는 각 엔티티 타입에 대하여 min은 이 엔티티 타입 내의 각 엔티티는 관계에 참여하는 개체의 수가 최소한 min 값 이상이어야 함을 의미
- ✓ max는 이 엔티티 타입 내의 각 엔티티는 관계에 참여하는 개체의 수가 max 값을 넘을 수 없음을 의미
- ✓ min=0은 어떤 엔티티가 반드시 관계에 참여해야 할 필요는 없음을 의미(선택적 참여) ; 예) 0 이상 이므로 반드시 참여할 필요가 없다.
- ✓ max=\*는 어떤 엔티티가 관계에 임의의 수만큼 참여할 수 있음을 의미  
예) max=10 일 경우 10개의 개체까지 참여할 수 있다.

## 5.2 ER 모델(계속)



[그림 5.16] 카디날리티의 최소값과 최대값

최소한 min 값 이상이어야 함, 최대한 max 값을 넘을 수 없다

〈표 5.3〉 카디날리티들의 몇 가지 유형

관계	$(\min_1, \max_1)$	$(\min_2, \max_2)$	그래픽 표기
1 : 1	$(0, 1)$	$(0, 1)$	$\overset{1}{\text{---}} \text{---} \text{---} \overset{1}{\text{---}}$
1 : N	$(0, *)$	$(0, 1)$	$\overset{1}{\text{---}} \text{---} \text{---} \overset{N}{\text{---}}$
M : N	$(0, *)$	$(0, *)$	$\overset{M}{\text{---}} \text{---} \text{---} \overset{N}{\text{---}}$

## 5.2 ER 모델(계속)

### □ 카디널리티 예제

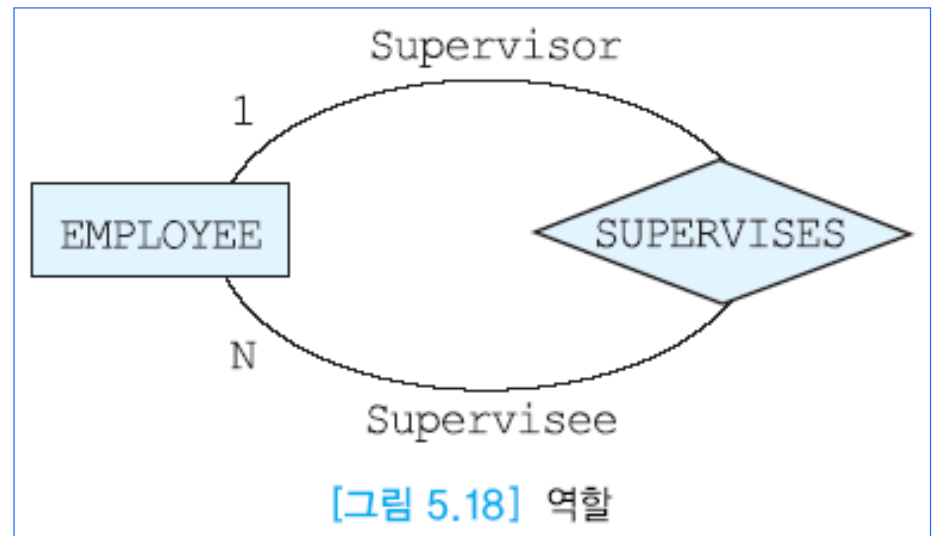


- 관계 대응수(카디널리티)의 (최소값, 최대값) 표기 방법
- 학과와 학생은 1:N관계를 맺고 있다. 즉, 하나의 학과에 여러 명의 학생이 소속되어 있다. 관계 대응수 1은 학과에 N은 학생에 표기
- (최소값, 최대값)으로 관계 대응수를 표기할 때는 각 객체의 관점에서 참여하는 횟수를 적는다.
- 먼저 **학과**의 관점에서 소속관계는 최소 0번(학과에 학생이 없는 경우)부터 최대 \*번(학과에 학생이 여러명인 경우)으로 학과 객체에 표기
- **학생**의 관점에서 소속관계는 최소 1번(학생은 반드시 **학과에 1번 이상 소속되어야**) 최대 1번(학생은 학과에 **1번 이상 소속될 수 없다**)으로 (1,1)로 학생 객체에 표기

## 5.2 ER 모델(계속)

### ❑ 역할(role)

- ✓ 관계 타입의 의미를 명확하게 하기 위해 사용됨
- ✓ 특히 하나의 관계 타입에 하나의 엔티티 타입이 여러 번 나타나는 경우에는 반드시 역할을 표기해야 함
- ✓ 관계 타입의 간선 위에 표시



## 5.2 ER 모델(계속)

### □ 전체 참여와 부분 참여

- ✓ 전체 참여는 어떤 관계에 엔티티 타입 E1의 모든 엔티티들이 관계 타입 R에 의해서 어떤 엔티티 타입 E2의 어떤 엔티티와 연관되는 것을 의미
- ✓ 부분 참여는 어떤 관계에 엔티티 타입 E1의 일부 엔티티만 참여하는 것을 의미
- ✓ 약한 엔티티 타입은 항상 관계에 전체 참여
- ✓ 전체 참여는 ER 다이어그램에서 이중 실선으로 표시

## 5.2 ER 모델(계속)

### □ 전체 참여와 부분 참여



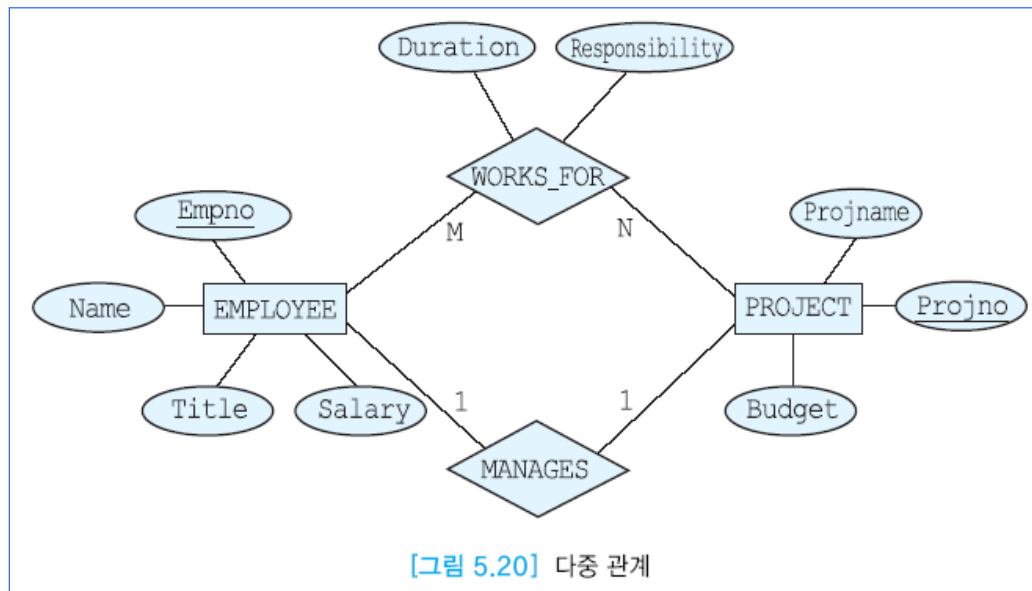
- ✓ DEPARTMENT 엔티티 타입이 MANAGES 관계에 **전체 참여**(모든 부서 엔티티들은 MANAGES 관계를 통해 EMPLOYEE 엔티티 타입에 속하는 **어떤 엔티티와 반드시 연결되어야 한다**), 즉 **부서 엔티티에 대하여 부서마다 반드시 1명의 관리자가 있어야 한다**.
- ✓ EMPLOYEE 엔티티 타입은 MANAGES 관계에 **부분참여**(EMPLOYEE 엔티티 타입에 속하는 **하나 이상의 엔티티가 MANAGES 관계에 참여하지 않음을 의미한다**).
- ✓ 즉, 일부 직원만 부서의 관리자가 될 수 있으므로 EMPLOYEE 엔티티 타입이 MANAGES 관계에 부분 참여한다.
- ✓ 카디널리티 비율과 함께 **참여 제약조건은 관계에 대한 중요한 제약조건**



## 5.2 ER 모델(계속)

### ❑ 다중 관계

- ✓ 두 엔티티 타입 사이에 두 개 이상의 관계 타입이 존재할 수 있음

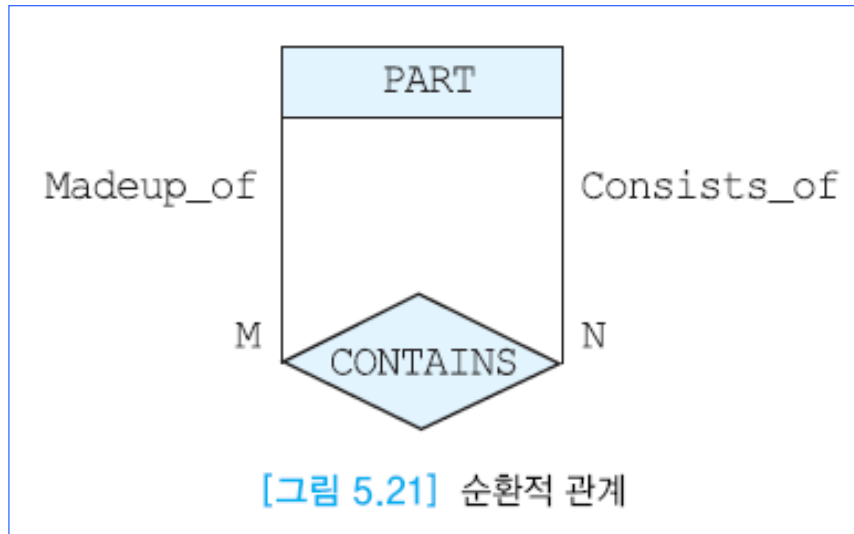


- ✓ 각 사원은 프로젝트에서 일하고(work\_for), 일부 사원은 어떤 프로젝트의 관리자(manages)이기도 하다. 프로젝트 사원과 관리자는 의미가 다르므로 하나의 관계로 두가지 의미를 나타낼수 없다.

## 5.2 ER 모델(계속)

### □ 순환적 관계

- ✓ 하나의 엔티티 타입이 동일한 관계 타입에 두 번 이상 참여하는 것



- ✓ 관계타입 CONTAINS(포함)은 부품 엔티티 타입을 부품 엔티티 타입과 연결하는 관계 타입
- ✓ 어떤 부품은 다른 부품들이 한 개 이상 모여서 구성된다.

## 5.2 ER 모델(계속)

### □ ER 스키마를 작성하기 위한 지침

- ✓ 개념적 설계에서는 요구사항 명세로부터 실세계 개념을
  - 엔티티 타입, 애트리뷰트, 관계 타입 중 어떤 것으로 모델링할 것인가?
  - 무엇이 엔티티와 관계인가?
  - 어떤 엔티티와 관계에 관하여 어떤 정보를 데이터베이스에 저장하는가?
  - 엔티티와 관계가 만족해야하는 제약조건(또는 비즈니스 규칙)은 무엇인가?

## 5.2 ER 모델(계속)

### □ 애트리뷰트 vs 엔티티

- ✓ 공급자-부품 데이터 베이스에서 각 공급자에 대해 다음과 같은 정보가 있을 때, 공급자가 엔티티는 명확하지만 **공급자도시는 엔티티인지, 공급자엔티티의 애트리뷰트인지 결정해야 한다.**

공급자번호, 공급자이름, 신용, 공급자도시

- 공급자 도시가 조직체에게 관심이 있는 객체인가?
- 공급자 도시에 관한 애트리뷰트들을 유지할 필요가 있는가?
- 공급자도시를 여러 엔티티 타입이 공유하는가?
- ✓ 질문 중 하나라도 “예”이면 새로운 엔티티, 그렇지 않으면 애트리뷰트
- ✓ 예:color가 자동차 회사에서는 자동차 엔티티의 애트리뷰트로 사용되지만, 페인트 제조회사에서는 각 color마다 필요한 염료의 유형과 염료의 양등의 정보를 유지해야하므로 엔티티로 사용

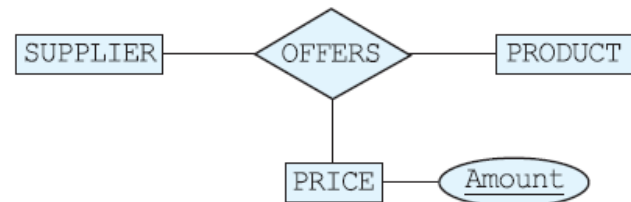
## 5.2 ER 모델(계속)

### □ 관계 vs 엔티티

- ✓ 어떤 개념을 엔티티 타입으로 모델링 할 것인가 또는 관계 타입으로 모델링할 것인가?
- ✓ 공급자가 동일한 상품을 수량에 따라 서로 다른 가격으로 공급할 수 없다.
- ✓ 가격을 엔티티 타입으로 모델링하면 가능하다.



[그림 5.22] Price가 관계에 애트리뷰트로 사용됨



[그림 5.23] Price가 엔티티 타입으로 모델링됨

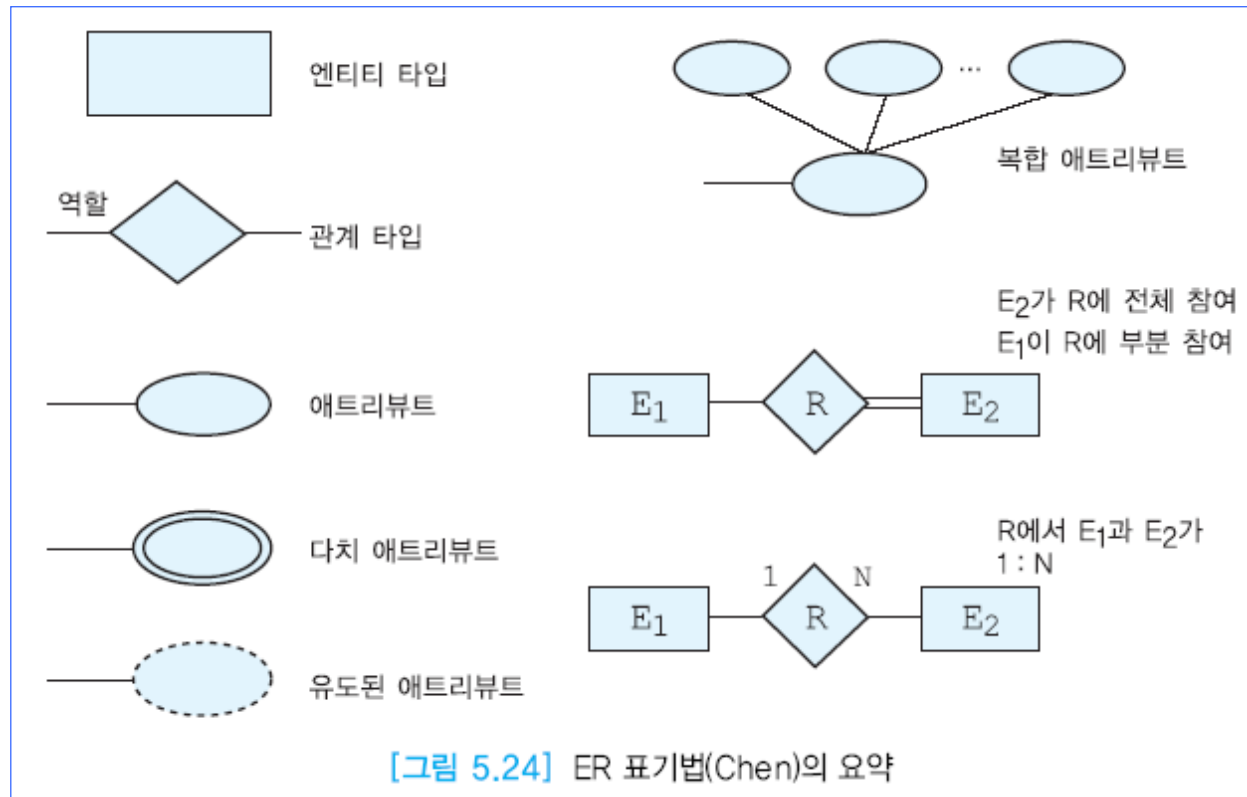
## 5.2 ER 모델(계속)

### □ 데이터베이스 설계 과정

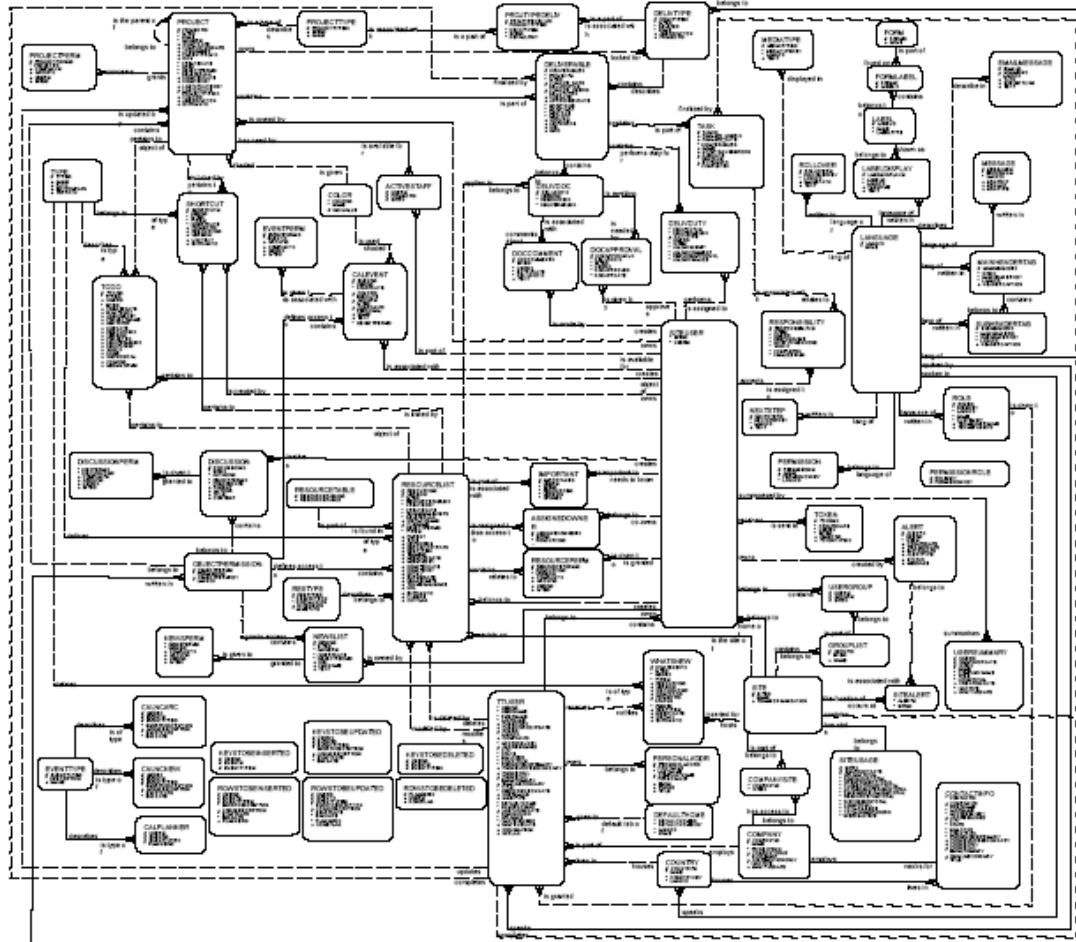
- ✓ 응용의 요구사항을 수집하여 기술
- ✓ 응용과 연관이 있는 엔티티 타입들을 식별
- ✓ 응용과 연관이 있는 관계 타입들을 식별
- ✓ 관계가 1:1, 1:N, M:N 중에서 어느 것에 해당하는지 결정
- ✓ 엔티티 타입과 관계 타입들에 필요한 애트리뷰트들을 식별하고, 각 애트리뷰트가 가질 수 있는 값들의 집합을 식별
- ✓ 엔티티 타입들을 위한 기본 키를 식별
- ✓ 응용을 위한 ER 스키마 다이어그램을 그림
- ✓ ER 스키마 다이어그램이 응용에 대한 요구사항과 부합되는지 검사
- ✓ ER 스키마 다이어그램을 DBMS에서 사용되는 데이터베이스 모델로 변환

## 5.2 ER 모델(계속)

### □ 본 책의 ER 표기법의 요약



## A Large ER Diagram





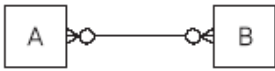
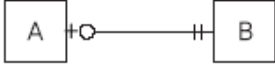

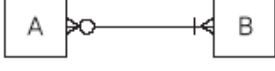


## 5.2 ER 모델(계속)

### □ ER 모델의 또 다른 표기법

- ✓ 본 장에서 사용한 표기법으로 수십 개 이상의 애트리뷰트가 엔티티 타입에 연결된 다이어그램을 나타내려면 매우 불편하고 공간을 많이 차지
  - ✓ ERWin 등의 CASE 도구들에서는 **새발**(crow-feet) 표기법이 흔히 사용됨
  - ✓ 새발 표기법(까마귀-발)에도 여러 가지 변형들이 존재함
- CASE(Computer aided Software Engineering)  
수많은 소프트웨어 요소들과 사람들이 관련된 크고 복잡한 프로젝트에서, 소프트웨어의 개발을 구조화하고 제어하는데 있어 컴퓨터의 지원을 받는 방법을 사용하는 것이다.
  - CASE의 사용은 각 개발 단계별 프로젝트 상황에 대해 설계자, 프로그래머, 테스터, 계획 수립자나 관리자들이 공통의 시각을 공유할 수 있게 해준다.
  - CASE는 부문별, 검사점 작업 진행에 도움을 준다. CASE 도구는 작업의 진도나 미진한 점 등을 그래픽으로 나타낼 수도 있다.
  - CASE는 또한 프로젝트의 사업계획, 설계요건, 설계 규격, 상세 코드 규격, 코드 단위, 테스트 문제 및 결과, 그리고 마케팅 및 서비스 계획 등을 담고 있는 문서들 및 프로그램 라이브러리 등을 위한 저장소로서의 역할을 하거나, 또는 그것들과 연결될 수도 있다.

## 5.2 ER 모델(계속)

	1:1 관계. 엔티티 A의 각 인스턴스는 엔티티 B의 0 또는 1개의 인스턴스와 연관됨. 엔티티 B의 각 인스턴스는 엔티티 A의 0 또는 1개의 인스턴스와 연관됨.
	1:N 관계. 엔티티 A의 각 인스턴스는 엔티티 B의 0개 이상의 인스턴스와 연관됨. 엔티티 B의 각 인스턴스는 엔티티 A의 0 또는 1개의 인스턴스와 연관됨.
	M:N 관계. 엔티티 A의 각 인스턴스는 엔티티 B의 0개 이상의 인스턴스와 연관됨. 엔티티 B의 각 인스턴스는 엔티티 A의 0 개 이상의 인스턴스와 연관됨.
	1:1 관계. 엔티티 A의 각 인스턴스는 엔티티 B의 1개의 인스턴스와 연관됨. 엔티티 B의 각 인스턴스는 엔티티 A의 0 또는 1개의 인스턴스와 연관됨.
	1:N 관계. 엔티티 A의 각 인스턴스는 엔티티 B의 1개 이상의 인스턴스와 연관됨. 엔티티 B의 각 인스턴스는 엔티티 A의 0 또는 1개의 인스턴스와 연관됨.
	M:N 관계. 엔티티 A의 각 인스턴스는 엔티티 B의 1개 이상의 인스턴스와 연관됨. 엔티티 B의 각 인스턴스는 엔티티 A의 0 개 이상의 인스턴스와 연관됨.

- ○ : 0을 의미
- | : 1을 의미
- ≧ : 이상을 의미

[그림 5.25] 새발 표기법의 예

## 5.2 ER 모델(계속)

① 1:1 관계(일 대 일): 개체 집합 A의 각 원소가 개체 집합 B의 원소 1개와 대응



<그래픽 표기법>



<새발(crow-feet) 표기법>

(한 명의 교수는 한 과목만 강의해야 하고,  
한 개의 과목은 한 교수에 의해서만 강의되어진다.)

## 5.2 ER 모델(계속)

- ② 1:N 관계(일 대 다): 개체 집합 A의 각 원소는 개체 집합 B의 원소 여러 개와 대응할 수 있고, 개체 집합 B의 각 원소는 개체 집합 A의 원소 1개와 대응



〈그래픽 표기법〉



〈새발(crow-feet) 표기법〉

(한 학과에는 여러 명의 학생이 소속될 수 있고,  
한 명의 학생은 한 학과에만 소속되어야 한다.)

## 5.2 ER 모델(계속)

- ③ N : M 관계(다 대 다): 개체 집합 A의 각 원소는 개체 집합 B의 원소 여러 개와 대응할 수 있고, 개체 집합 B의 각 원소는 개체 집합 A의 원소 여러 개와 대응할 수 있음



<그래픽 표기법>



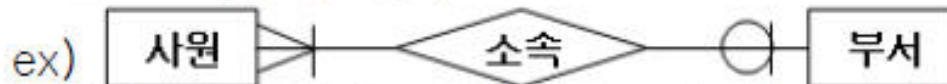
<새발(crow-feet) 표기법>

(한 명의 학생은 여러 과목을 수강할 수 있고,  
한 개의 과목은 여러 학생에 의해 수강되어질 수 있다.)

## 5.2 ER 모델(계속)

- 새발(crow-feet) 표기법

- O : 0을 의미
- | : 1을 의미
- $\rightarrow$  : 이상을 의미



(한 명의 사원은 0개 또는 한 개의 부서에 소속될 수 있고  
한 개의 부서는 1명 이상의 사원을 소속할 수 있다.)



(한 명의 사원은 한 개의 부서에 소속되어야 하며,  
한 개의 부서는 0명 이상의 사원을 소속할 수 있다.)

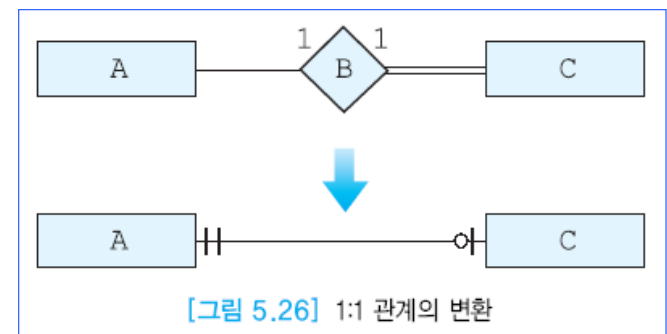
## 5.2 ER 모델(계속)

❑ 본 책의 표기법을 새발 표기법으로 표현하는 방법

✓ 1:1 관계

✓ 엔티티 타입 A의 각 엔티티는 엔티티 타입의 C 엔티티 0개 또는 1개와 연관된다. 엔티티 타입 C의 각 엔티티는 엔티티 타입 A의 엔티티 1개와 연관된다. 따라서 엔티티 타입 A는 관계 타입 B에 부분 참여하고, 엔티티 타입 C는 관계 타입 B에 전체 참여한다. 엔티티 타입 A와 엔티티 타입 C는 1:1관계를 갖는다.

✓ A = EMPLOYEE , 관계 타입 B는 MANAGES, C=DEPARTMENT

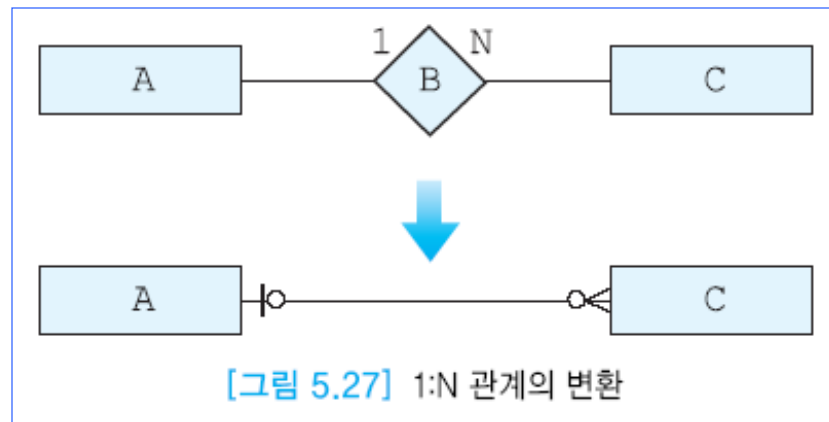


## 5.2 ER 모델(계속)

□ 본 책의 표기법을 새발 표기법으로 표현하는 방법(계속)

✓ 1:N 관계

✓ 엔티티 타입 A의 각 엔티티는 엔티티 타입 C의 엔티티 0개 이상과 연관, 엔티티 타입 C의 각 엔티티는 엔티티 타입 A의 엔티티에 0개 또는 1개와 연관, 따라서 A와 C는 1:N관계를 갖고 엔티티 타입 A와 엔티티 타입 C는 관계 타입 B에 부분 참여한다.



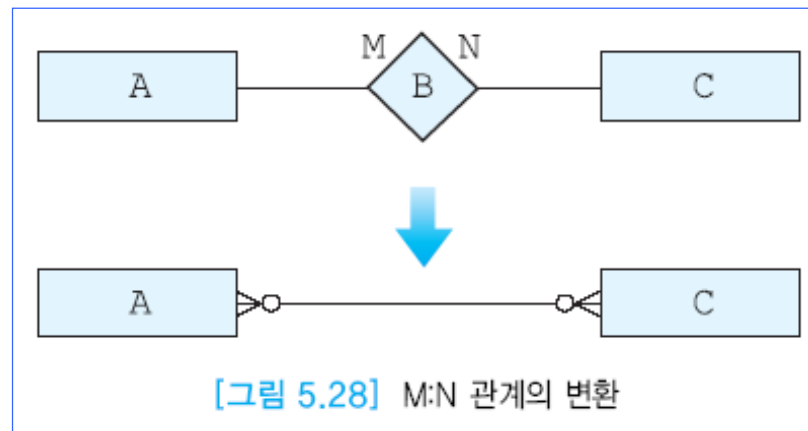


## 5.2 ER 모델(계속)

### ❑ 본 책의 표기법을 새발 표기법으로 표현하는 방법(계속)

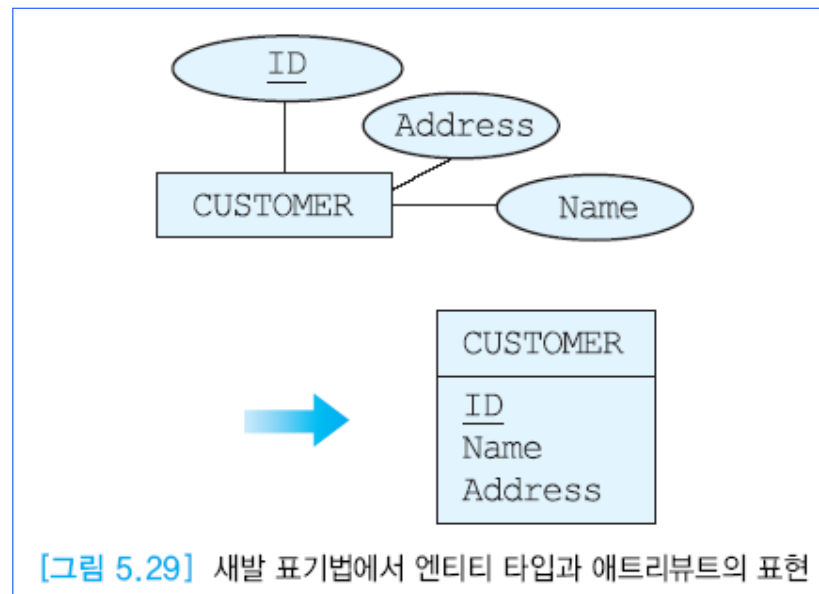
#### ✓ M:N 관계

- ✓ 엔티티 타입 A의 각 엔티티는 엔티티 타입 C의 엔티티에 0개 이상과 연관, 엔티티 타입 C의 각 엔티티는 엔티티 타입 A의 엔티티 0개 이상과 연관, 따라서 A와 C는 M:N관계를 갖고 엔티티 타입 A와 엔티티 타입 C는 관계 타입 B에 부분 참여한다.



## 5.2 ER 모델(계속)

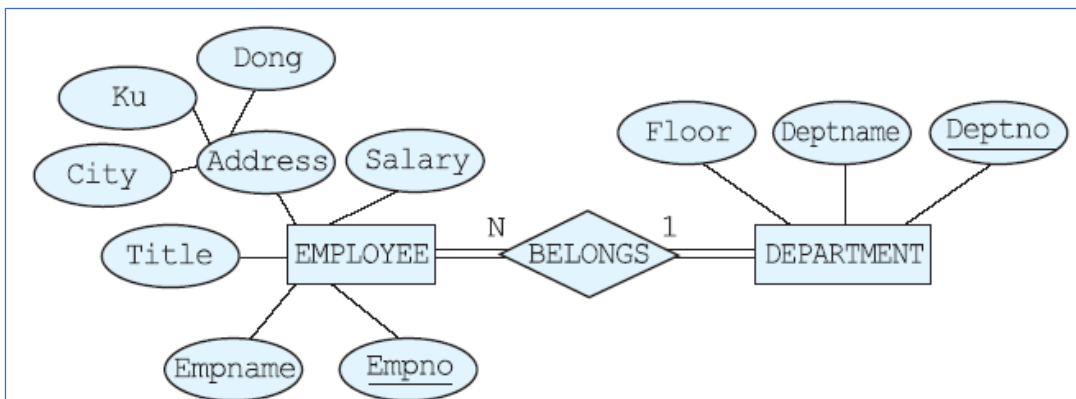
- ❑ 본 책의 표기법을 새발 표기법으로 표현하는 방법(계속)
  - ✓ 엔티티 타입과 애트리뷰트
  - ✓ 새발 표기법 문제점 : 다치 애트리뷰트, 유도된 애트리뷰트, 복합 애트리뷰트를 구분하기 어렵다.



## 5.3 데이터베이스 설계 사례(계속)

### □ 관계와 애트리뷰트들을 식별

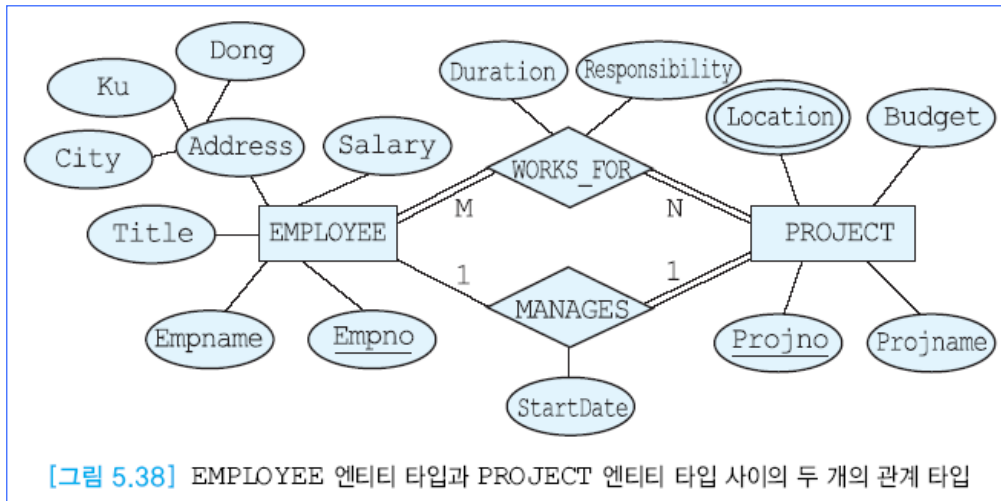
- 요구사항 명세로부터 동사를 주목하여 엔티티 타입들을 연관 시키는 관계 타입들을 식별, 사원 엔티티 타입과 부서 엔티티 타입은 소속(BELONGS) 관계로 연결, **각 부서에는 여러 명의 사원들이 속하고 한 사원은 한 부서에만 속할 수 있으므로 부서와 사원 엔티티 타입은 1:N**
- 사원 엔티티 타입과 부서엔티티 타입은 관계 타입에 **전체 참여**



[그림 5.37] EMPLOYEE 엔티티 타입과 DEPARTMENT 엔티티 타입 사이의 BELONGS 관계 타입

## 5.3 데이터베이스 설계 사례(계속)

### □ 관계와 애트리뷰트들을 식별(계속)

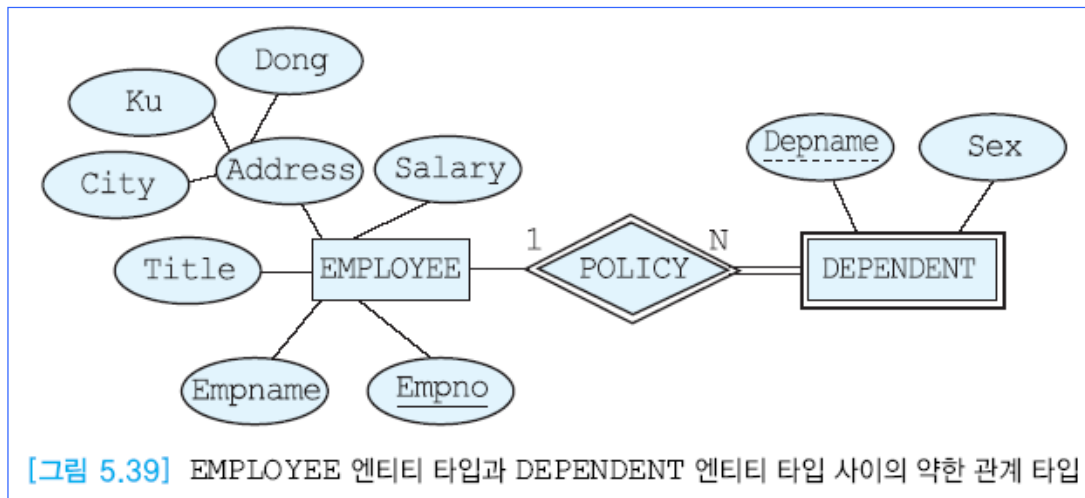


[그림 5.38] EMPLOYEE 엔티티 타입과 PROJECT 엔티티 타입 사이의 두 개의 관계 타입

- 사원과 프로젝트 엔티티 타입은 두가지 관계 타입으로 연결
- WORKS\_FOR 관계 타입은 **각 사원이 어떤 프로젝트에서 근무하는가?** M:N관계 전체참여, 관계 타입에 Responsibility(역할)와 Duration(근무기간)애트리뷰트
- MANAGES 관계 타입은 **어떤 사원이 어떤 프로젝트의 관리자 인가?** 각 프로젝트에는 **최대한 1명의 관리자만 있으므로 사원과 프로젝트 엔티티 타입은 1:1 관계**, 모든 사원이 관리자가 될 수 없으므로 사원 엔티티 타입은 부분참여, 모든 프로젝트에는 관리자가 필요하므로 프로젝트 엔티티 타입은 전체참여// 관계 타입의 애트리뷰트는 StartDate(시작날짜) 추가

## 5.3 데이터베이스 설계 사례(계속)

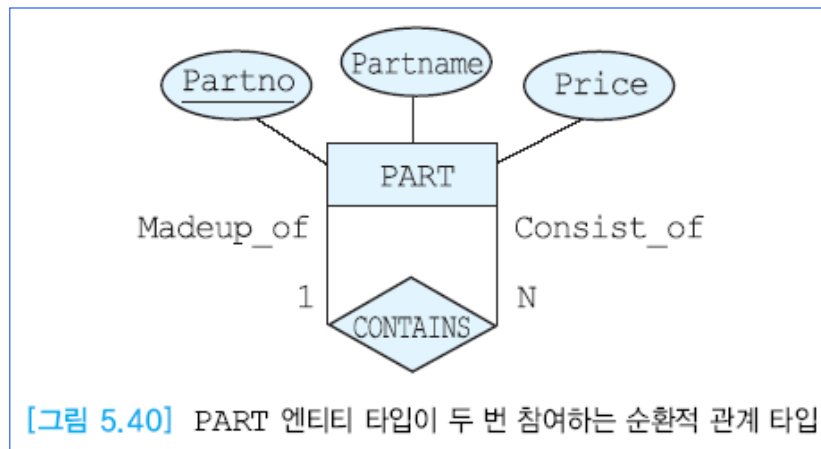
### □ 관계와 애트리뷰트들을 식별(계속)



- 사원의 부양가족은 회사의 의료보험 혜택을 받으므로 사원 엔티티 타입과 부양가족 엔티티 타입은 Policy(보험)관계 타입으로 연결
- 각 사원은 0명 이상의 부양가족을 가질 수 있고, 각 부양가족은 한 명의 사원과 연관 되므로 **사원과 부양가족 엔티티 타입은 1:N관계**, 부양가족 엔티티는 연관된 사원 엔티티가 존재하지 않으면 존재할 수 없는 약한 엔티티 타입으로 **전체참여**한다. 약한 관계 타입

## 5.3 데이터베이스 설계 사례(계속)

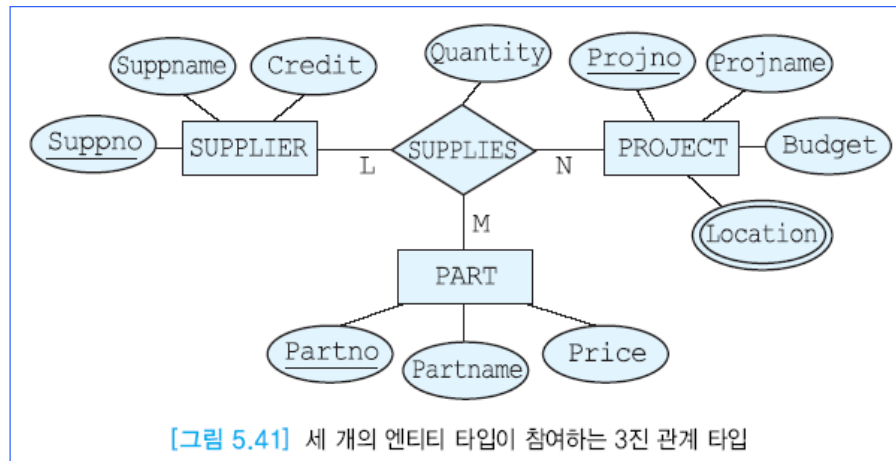
### □ 관계와 애트리뷰트들을 식별(계속)



- 한 부품은 다시 여러 부품들로 이루어 질 수 있으므로 부품 엔티티 타입은 CONTAINS(포함)관계 타입에 2번 참여한다. 즉, 순환적 관계 타입. 한 엔티티 타입이 어떤 관계 타입에 두번 이상 나타나는 경우에는 역할을 표시한다. 부품 엔티티 타입과 부품 엔티티 타입 사이에는 1:N관계

## 5.3 데이터베이스 설계 사례(계속)

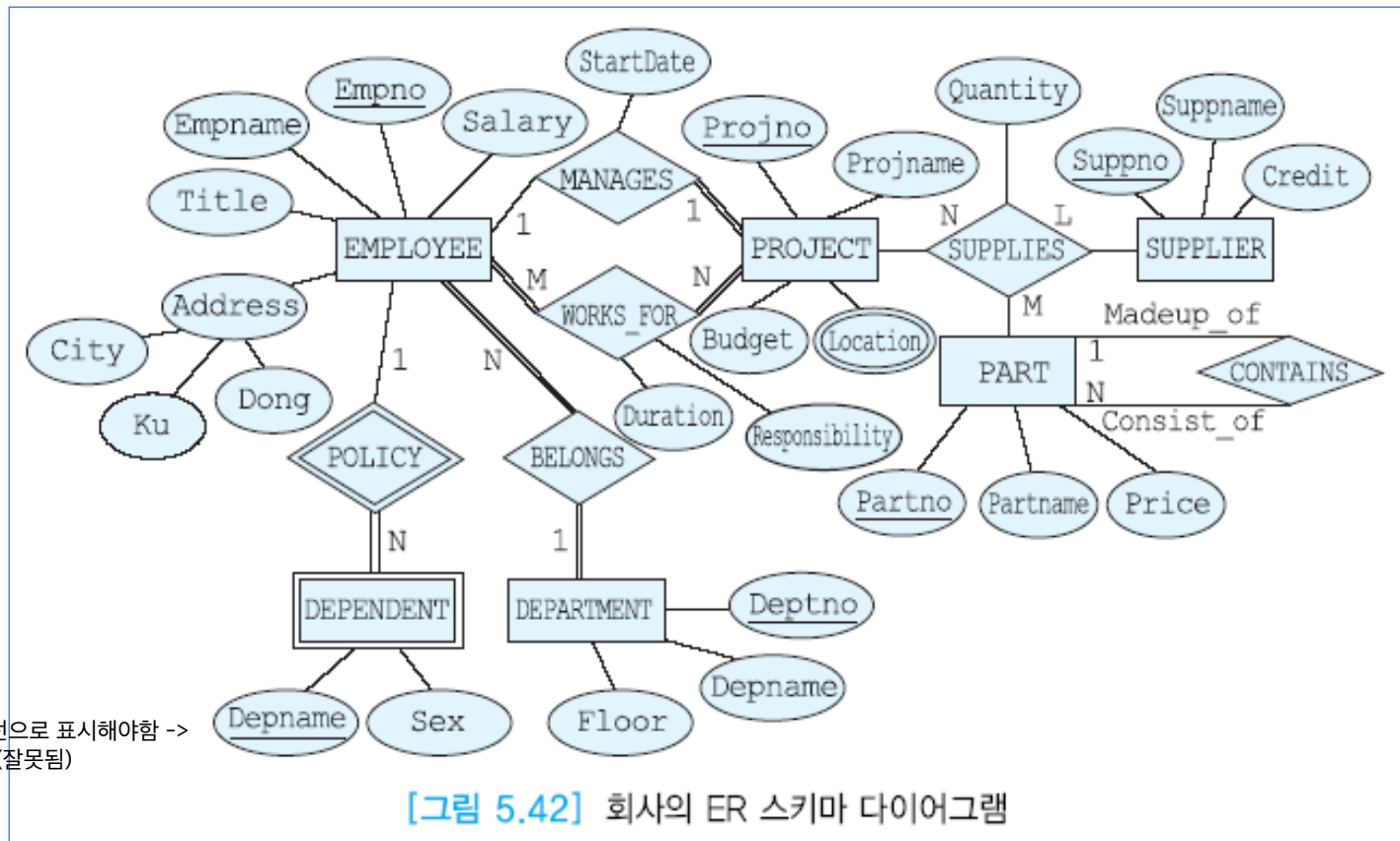
### □ 관계와 애트리뷰트들을 식별(계속)



- 공급자, 프로젝트, 부품 엔티티 타입을 연결하는 3진 관계 타입 SUPPLIES(공급)에는 Quantity(수량)애트리뷰트가 필요, 이 관계 타입에 참여하는 카디널리티는 L: M : N이다.

## 5.3 데이터베이스 설계 사례(계속)

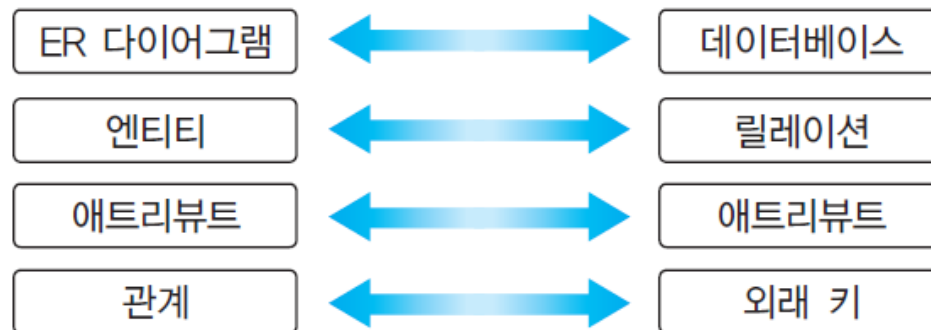
### □ 전체적인 회사의 ER 스키마 다이어그램





## 5.4 ER 스키마를 관계 모델의 릴레이션으로 사상

□ ER 스키마의 각 구성요소에 릴레이션으로 사상하는 알고리즘을 기계적으로 적용하여 생성한 릴레이션 스키마들이다. 이 스키마를 정규화 과정을 통하여 중복을 감소 시키고, 여러 가지 갱신 이상들이 발생하지 않도록 릴레이션 스키마를 정제한 후 관계 데이터베이스에 릴레이션을 정의하고, 생성된 데이터를 저장하고, 관리하는 것이 바람직하다.



[그림 5.51] ER 개념과 데이터베이스 개념들의 대응 관계