

제 5 장



데이터베이스 설계와 ER 모델

어플리케이션 스키마 설계 : 프로젝트

- 5.1 데이터베이스 설계의 개요
- 5.2 ER 모델
- 5.3 데이터베이스 설계 사례
- 5.4 논리적 설계: ER 스키마를 관계 모델의 릴레이션으로 사상
 - 연습문제

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

물리적인 것뿐만
아니라 개념적인 것
도 포함

5장. 데이터베이스 설계와 ER 모델(계속)

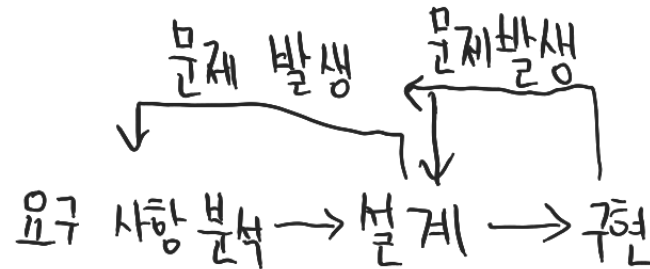
□ 개념적 수준의 모델

- ✓ 특정 데이터 모델과 독립적으로 응용 세계를 모델링할 수 있도록 함
- ✓ 데이터베이스 구조나 스키마를 하향식으로 개발할 수 있기 위한 틀(framework)을 제공함
- ✓ 인기 있는 개념적 수준의 모델은 엔티티-관계(ER: Entity-Relationship) 모델
- ✓ ER 모델과 같은 개념적인 데이터 모델이 사상될 수 있는 다수의 구현 데이터 모델(implementation data model)이 존재함 / 계층 모델
- ✓ 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	비용	어떤 DBMS가 적절한가?
논리적 설계	개념적 설계를 데이터베이스 스키마로 사상	데이터베이스가 무엇을 모델링하는가?
스키마 정제	중복을 제거함 - 데이터베이스 스키마의 정규화 → 문제 확인	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		

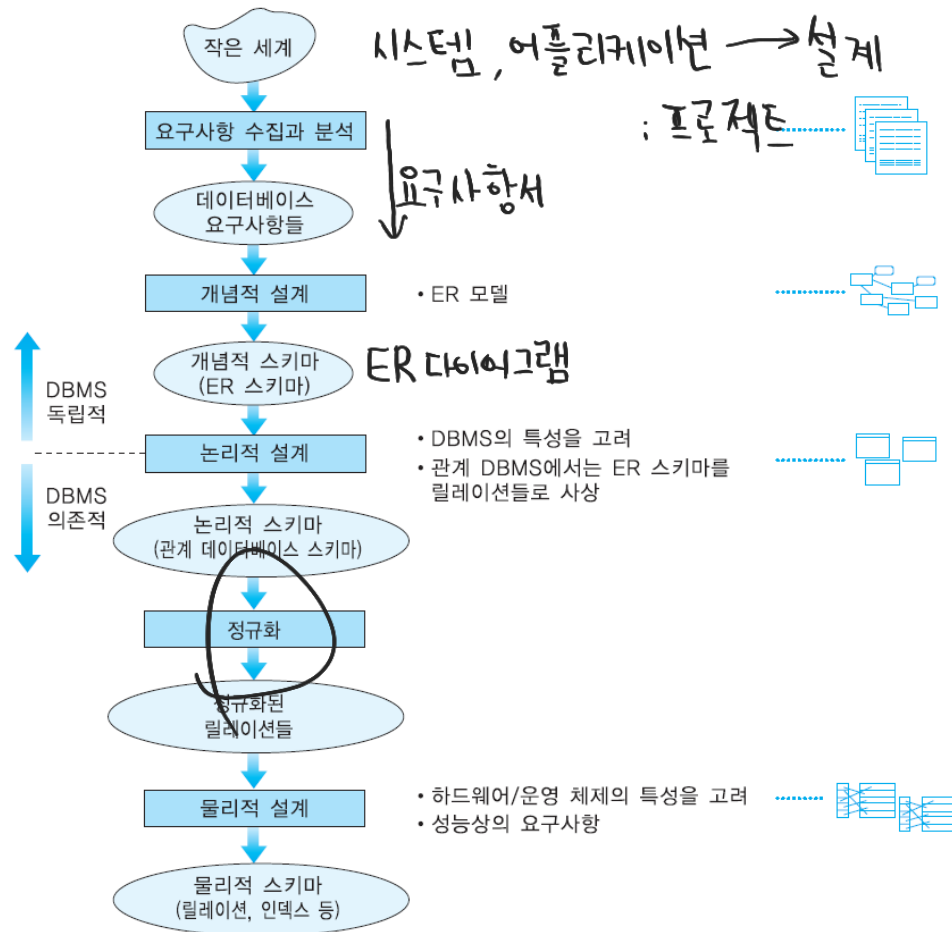
↑ 성능상의 문제 고려
↓ 성능상의 문제 고려

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

왜 쓰는 거지? 무슨 일 함? 상세하게 질문, 여러 사용자를 만나 질문

5.1 데이터베이스 설계의 개요(계속)

□ 개념적 설계

- ✓ 모든 물리적인 사항과 독립적으로, ^{구현을 재사용할지 봐야함} 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

- ✓ 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 과정을 적용함 중복 제거 등
- ✓ 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
- ✓ 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

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

5.2 ER 모델

□ ER 모델

- ✓ 데이터베이스 설계를 용이하게 하기 위해서 P.P. Chen이 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

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** 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

5.2 ER 모델(계속)

□ 엔티티

- ✓ 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



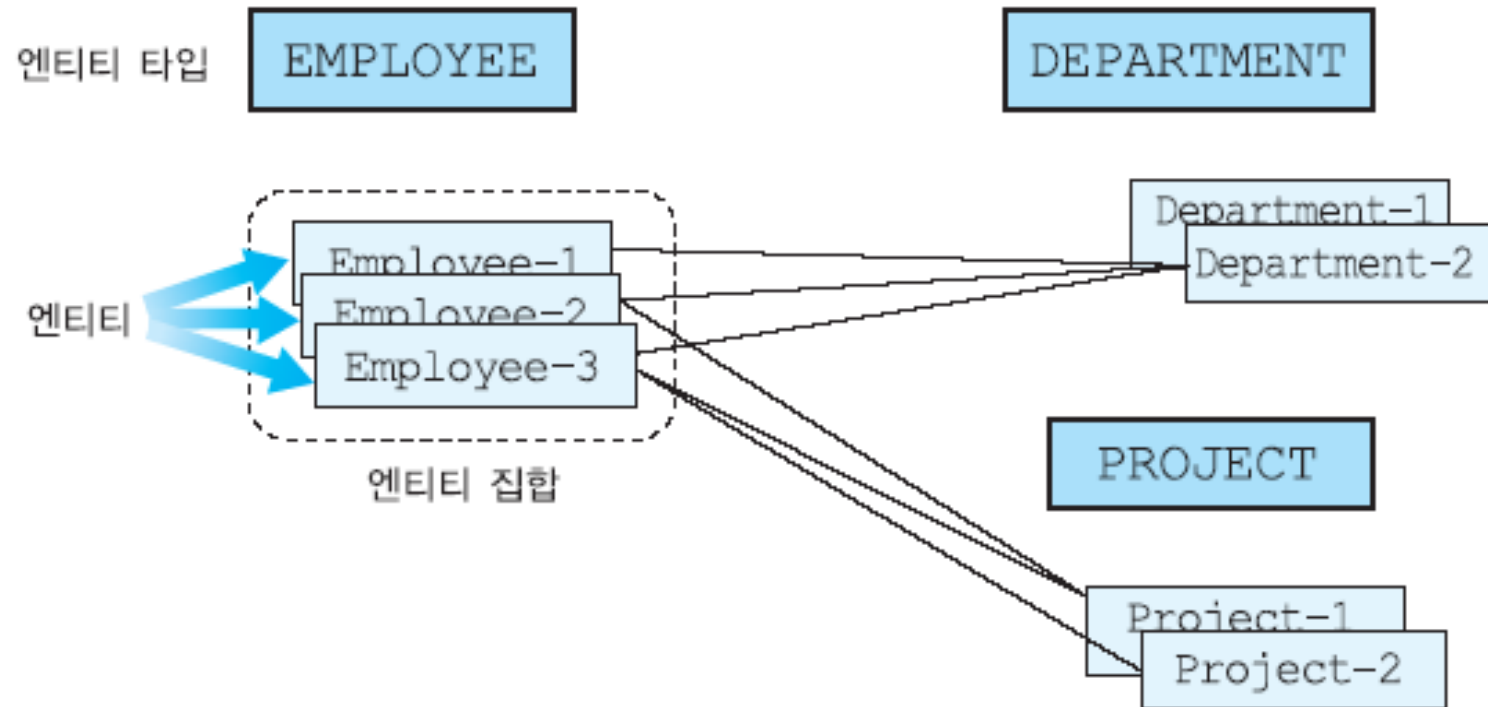
[그림 5.2] 엔티티의 예

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 내포에 해당하고, 엔티티 집합은 관계 모델의 릴레이션의 외연에 해당함
- ✓ 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
- ✓ ER 다이어그램에서 기본 키에 속하는 애트리뷰트는 밑줄을 그어 표시함

5.2 ER 모델(계속)

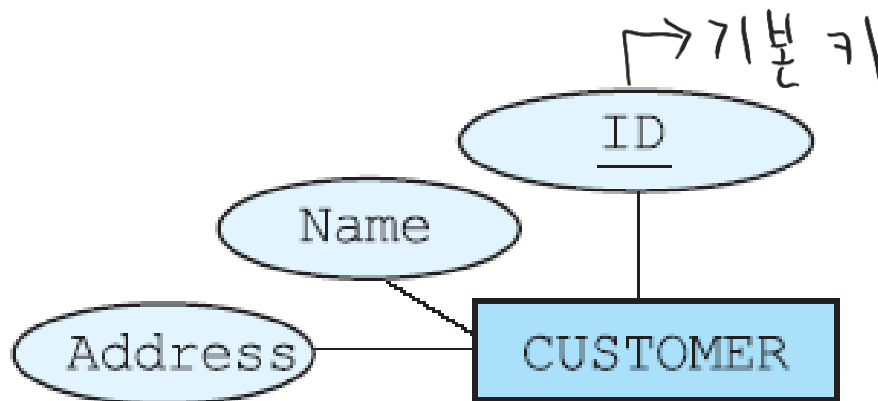
□ 애트리뷰트(계속)

- ✓ 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

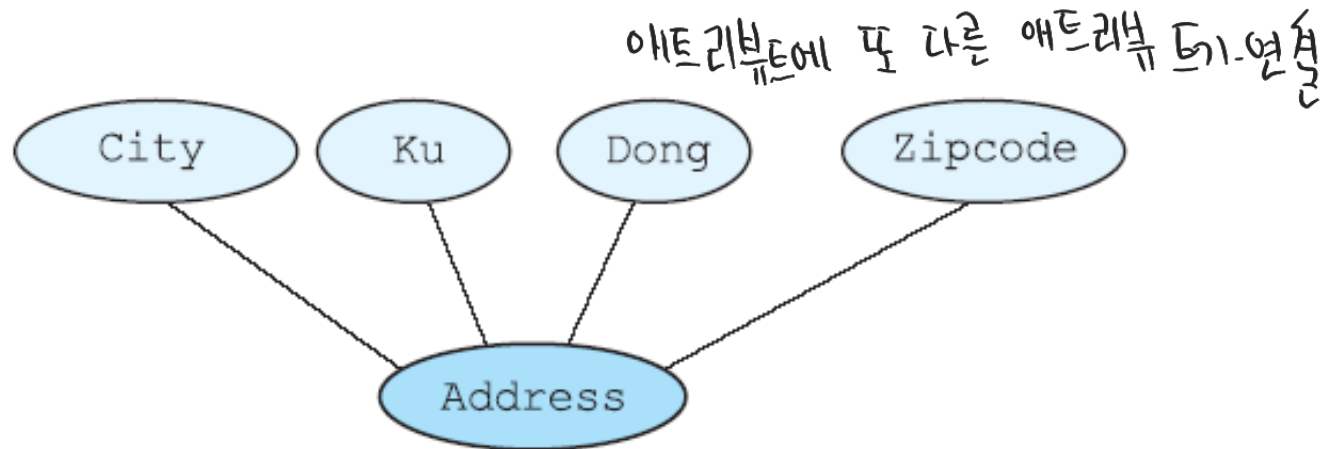


[그림 5.4] 단순 애트리뷰트

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



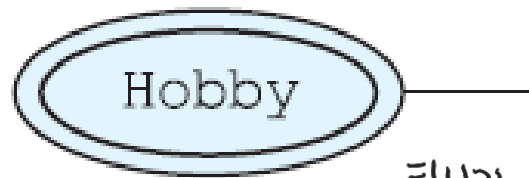
[그림 5.5] 복합 애트리뷰트

5.2 ER 모델(계속)

- ❑ 단일 값 애트리뷰트(single-valued attribute)
 - ✓ 각 엔티티마다 정확하게 하나의 값을 갖는 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



하obby가 아니라 여러 개의 값을 가질 수 있음.

[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
- ✓ 관계 데이터베이스에서 릴레이션의 애트리뷰트로 포함시키지 않는 것이 좋음
- ✓ ER 다이어그램에서 점선 타원으로 표현함

다른 애트리뷰트를 통해서 계산을 한다던가

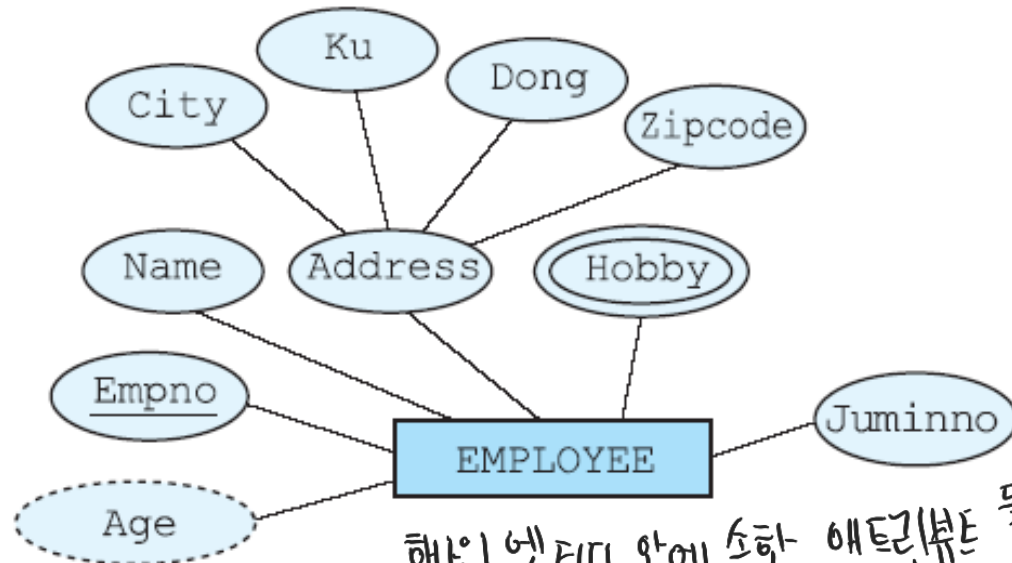


[그림 5.7] 유도된 애트리뷰트

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]

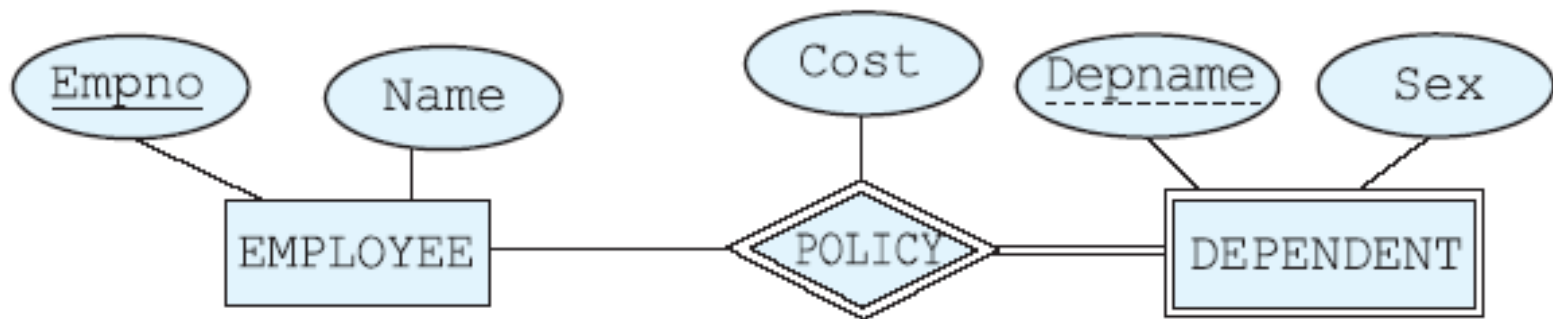
여러 가지 애트리뷰트의 예
하나의 엔티티 안에 속한 애트리뷰트들이 저장되어 있는지

5.2 ER 모델(계속)

□ 약한 엔티티 타입

- ✓ 키를 형성하기에 충분한 애트리뷰트들을 갖지 못한 엔티티 타입
- ✓ 약한 엔티티 타입에게 키 애트리뷰트를 제공하는 엔티티 타입을 **소유 엔티티 타입**(owner entity type) 또는 **식별 엔티티 타입**(identifying entity type)라고 부름
- ✓ 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 직원들의 부양가족들 전체에서는 같은 경우가 생길 수 있는 애트리뷰트

5.2 ER 모델(계속)



[그림 5.9] 약한 엔티티 타입

5.2 ER 모델(계속)

□ 관계와 관계 타입

- ✓ 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

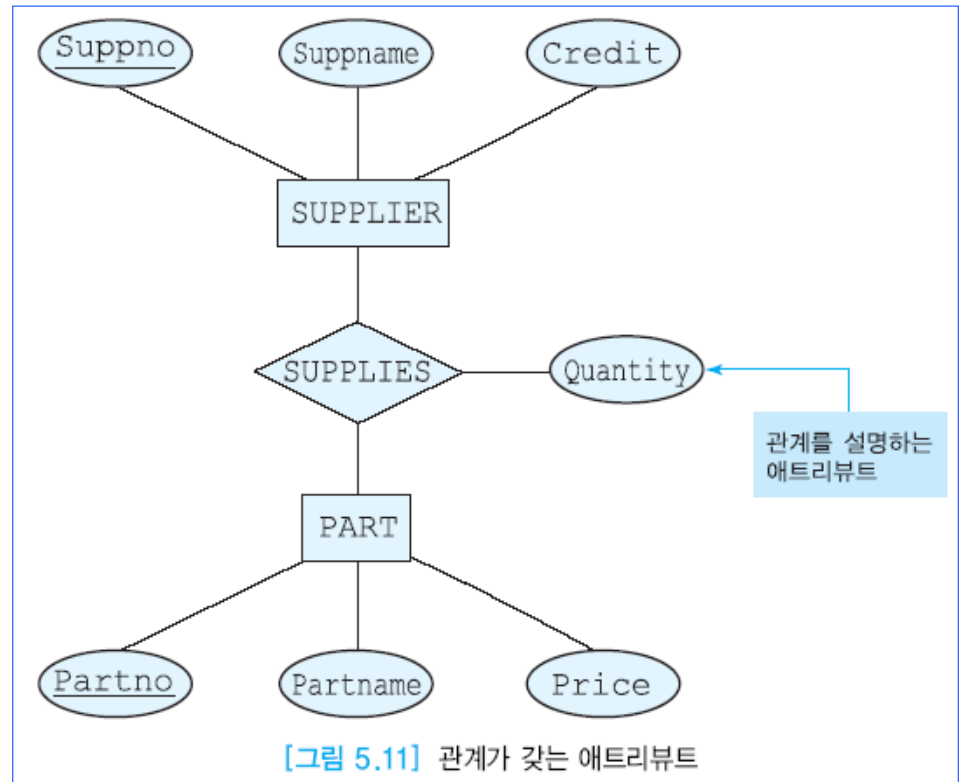
〈표 5.2〉 엔티티와 엔티티 간의 관계의 예

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

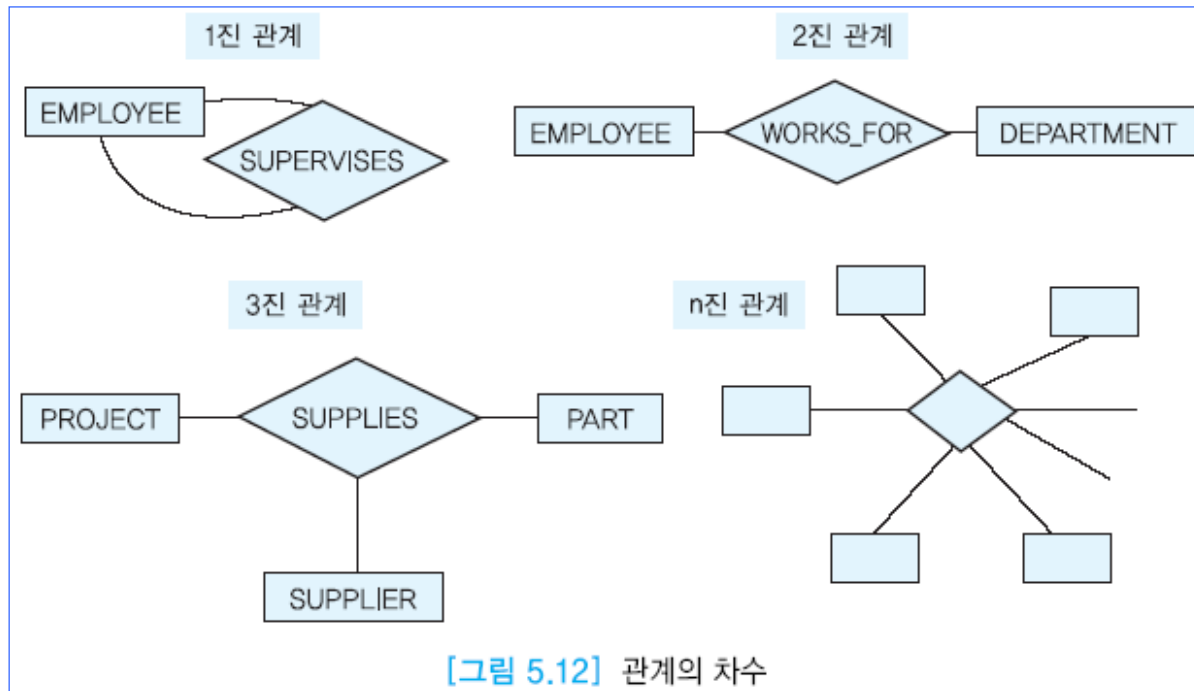
- ✓ 관계 타입은 관계의 특징을 기술하는 애트리뷰트들을 가질 수 있음
- ✓ 관계 타입은 키 애트리뷰트를 갖지 않음



5.2 ER 모델(계속)

❑ 차수(degree)

- ✓ 관계로 연결된 엔티티 타입들의 개수를 의미
- ✓ 실세계에서 가장 흔한 관계는 두 개의 엔티티 타입을 연결하는 2진 관계

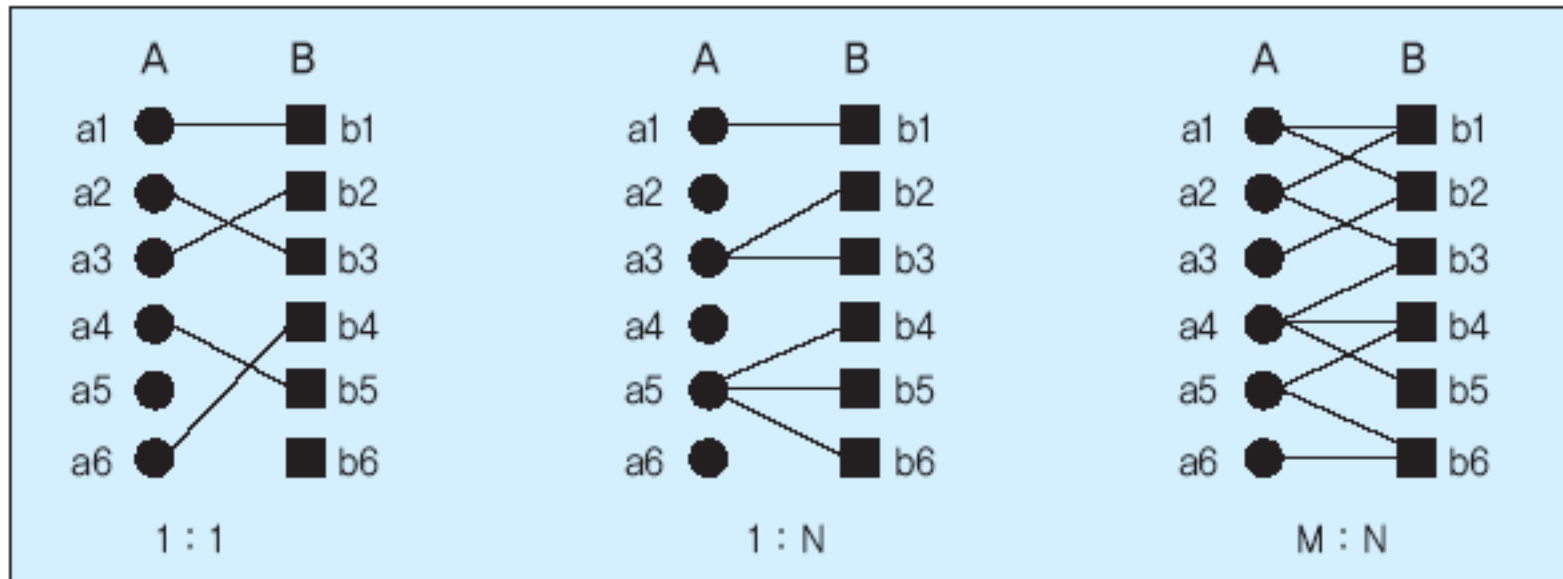


5.2 ER 모델(계속)

□ 카디날리티

- ✓ 카디날리티 비율은 한 엔티티가 참여할 수 있는 관계의 수를 나타냄
- ✓ 관계 타입에 참여하는 엔티티들의 가능한 조합을 제한함
- ✓ 관계를 흔히 1:1, 1:N, M:N으로 구분
- ✓ 카디날리티에 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

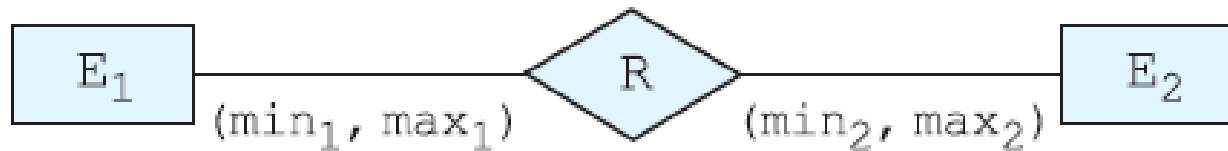
- ✓ 한 엔티티 타입에 속하는 임의의 개수의 엔티티가 다른 엔티티 타입에 속하는 임의의 개수의 엔티티와 연관됨
- ✓ 예: 각 사원에 대해 여러 대의 PC가 있고, 각 PC에 대해 여러 명의 사원들이 있으면 사원과 PC 간의 관계는 M:N 관계

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 적어도 min 번 관계에 참여함을 의미
- ✓ max는 이 엔티티 타입 내의 각 엔티티는 최대한 max 번 관계에 참여함을 의미
- ✓ min=0은 어떤 엔티티가 반드시 관계에 참여해야 할 필요는 없음을 의미
- ✓ max=*는 어떤 엔티티가 관계에 임의의 수만큼 참여할 수 있음을 의미

5.2 ER 모델(계속)

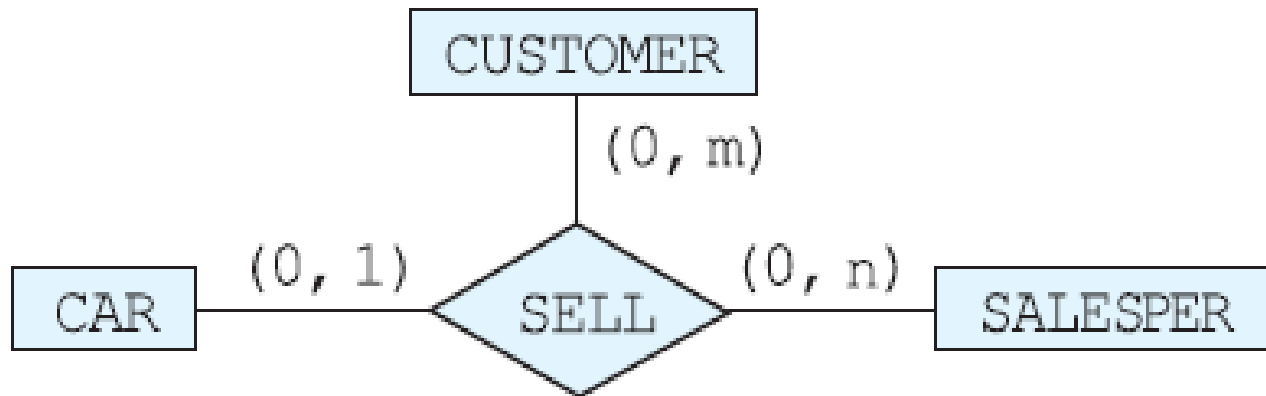


[그림 5.16] 카디날리티의 최소값과 최대값

〈표 5.3〉 카디날리티들의 몇 가지 유형

관계	(\min_1, \max_1)	(\min_2, \max_2)	그래픽 표기	화살표 표기
1 : 1	$(0, 1)$	$(0, 1)$	$1 \text{ --- } \diamond \text{ --- } 1$	\longleftrightarrow
1 : N	$(0, *)$	$(0, 1)$	$1 \text{ --- } \diamond \text{ --- } N$	$\longleftarrow \diamond \longrightarrow$
M : N	$(0, *)$	$(0, *)$	$M \text{ --- } \diamond \text{ --- } N$	$\text{---} \diamond \text{---}$

5.2 ER 모델(계속)

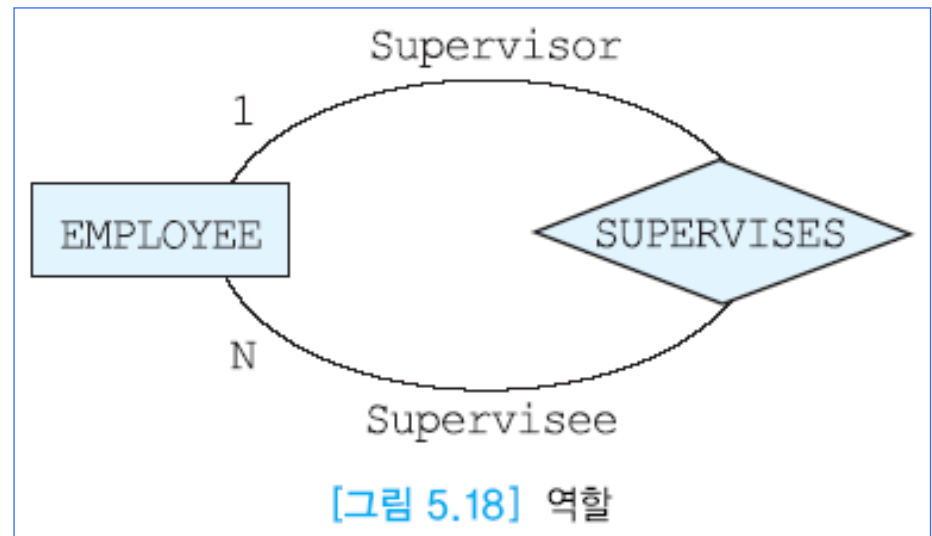


[그림 5.17] 카디널리티가 명시된 3진 관계 타입

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

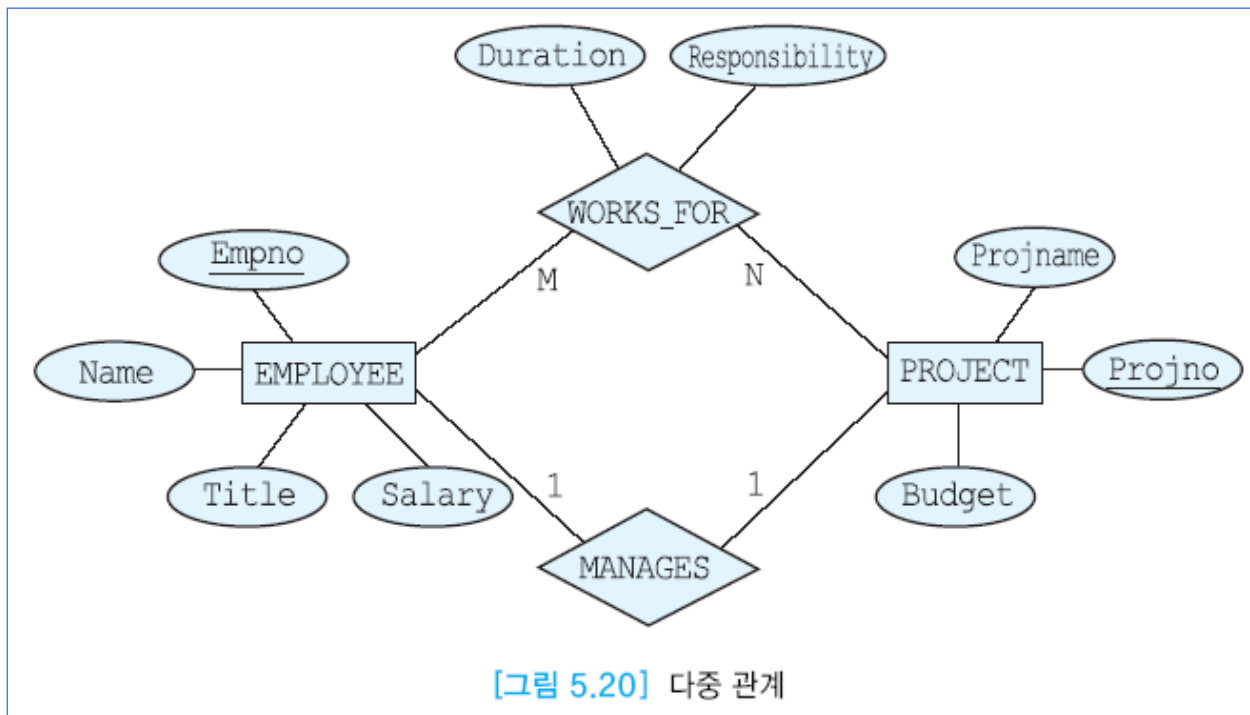
- ✓ 전체 참여는 어떤 관계에 엔티티 타입 E1의 모든 엔티티들이 관계 타입 R에 의해서 어떤 엔티티 타입 E2의 어떤 엔티티와 연관되는 것을 의미
- ✓ 부분 참여는 어떤 관계에 엔티티 타입 E1의 일부 엔티티만 참여하는 것을 의미
- ✓ 약한 엔티티 타입은 항상 관계에 전체 참여
- ✓ 전체 참여는 ER 다이어그램에서 이중 실선으로 표시
- ✓ 카디널리티 비율과 함께 참여 제약조건은 관계에 대한 중요한 제약조건



5.2 ER 모델(계속)

□ 다중 관계

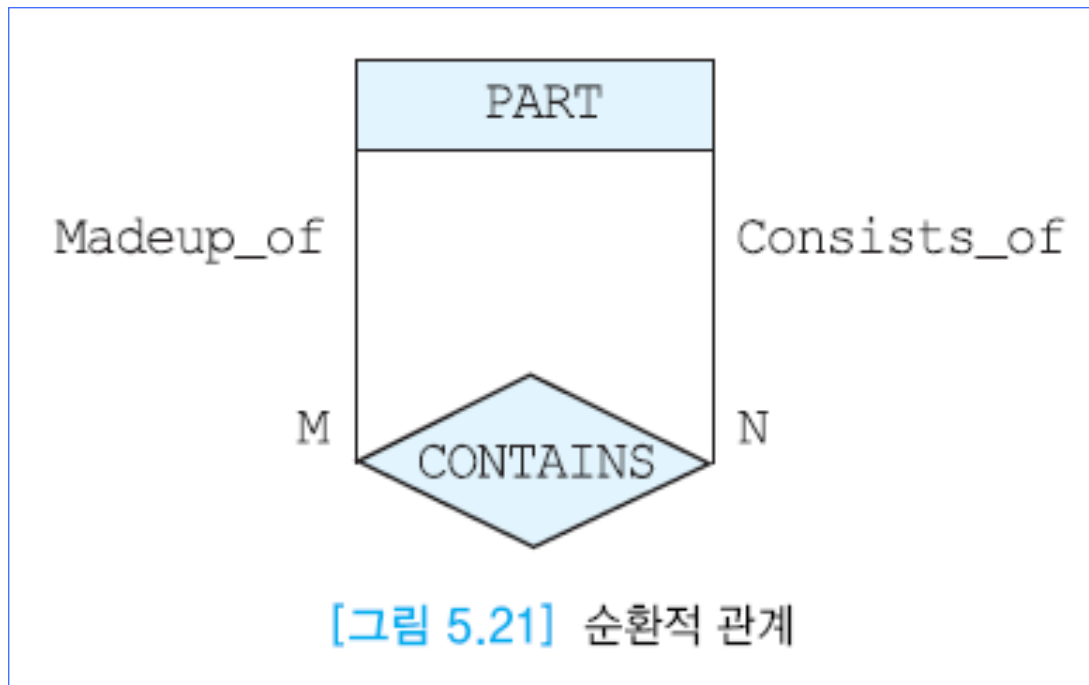
- ✓ 두 엔티티 타입 사이에 두 개 이상의 관계 타입이 존재할 수 있음



5.2 ER 모델(계속)

□ 순환적 관계

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



5.2 ER 모델(계속)

- ER 스키마를 작성하기 위한 지침
 - ✓ 엔티티는 키 애트리뷰트 이외에 설명 정보를 추가로 가짐
 - ✓ 다치 애트리뷰트는 엔티티로 분류해야 함
 - ✓ 애트리뷰트들이 직접적으로 설명하는 엔티티에 애트리뷰트들을 붙임
 - ✓ 가능한 한 복합 식별자를 피함
 - ✓ 관계는 일반적으로 독자적으로 존재할 수 없지만 엔티티 타입과 관계 타입을 절대적으로 구분하는 것은 어려움

5.2 ER 모델(계속)

□ 애트리뷰트 vs. 엔티티

- ✓ 엔티티 타입과 애트리뷰트를 구분하는 절대적인 기준 없음
- ✓ 예: 공급자에 대한 정보가 아래와 같다

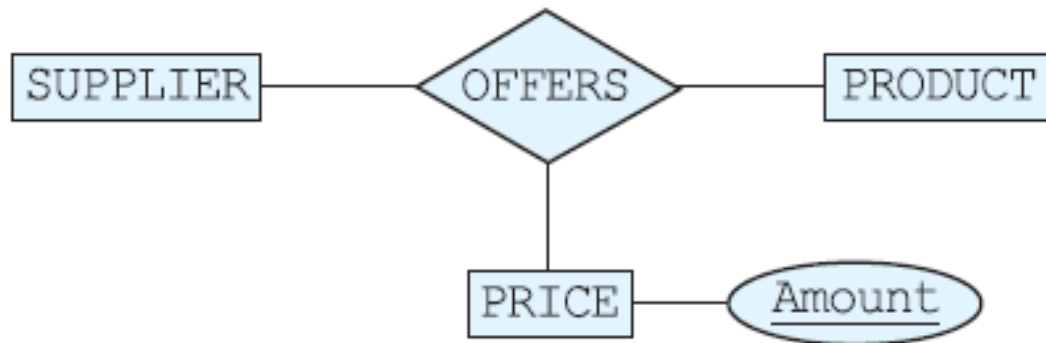
공급자 번호, 공급자 이름, 신용, 공급자 도시

- 공급자가 엔티티인 것은 명확
- 공급자 도시가 엔티티인가 또는 애트리뷰트인가?
- 고려사항:
 - 도시가 조직체에 관심이 있는 객체인가?
 - 도시에 관한 애트리뷰트들을 유지할 필요가 있는가?
 - 도시를 여러 엔티티 타입들이 공유하는가?
- 설계 방안
 - 위 고려사항 중에 하나라도 대답이 '예' 라면 도시에 대한 추가 정보를 모아서 엔트리로 나타냄. 그렇지 않으면 애트리뷰트로 표현.

5.2 ER 모델(계속)



[그림 5.22] Price가 관계에 애트리뷰트로 사용됨



[그림 5.23] Price가 엔티티 타입으로 모델링됨

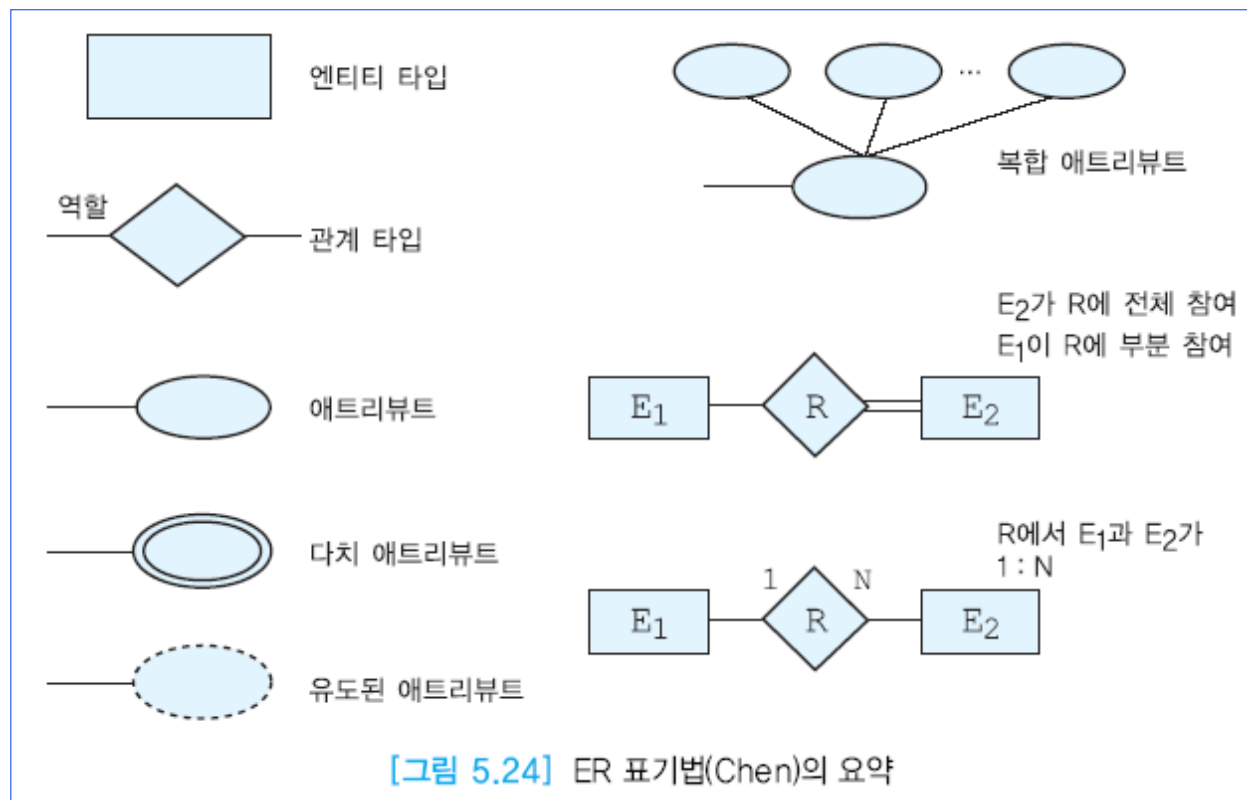
5.2 ER 모델(계속)

□ 데이터베이스 설계 과정

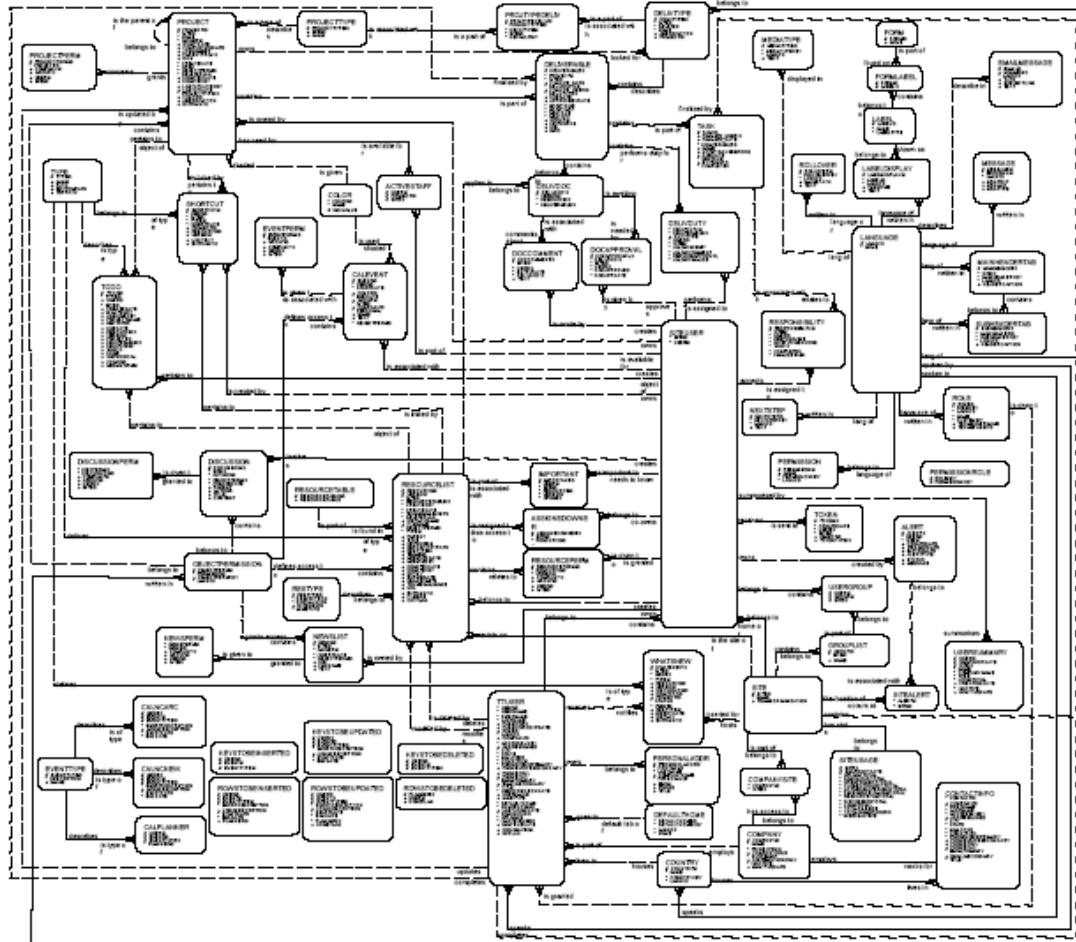
- ✓ 응용의 요구사항을 수집하여 기술
- ✓ 응용과 연관이 있는 엔티티 타입들을 식별
- ✓ 응용과 연관이 있는 관계 타입들을 식별
- ✓ 관계가 1:1, 1:N, M:N 중에서 어느 것에 해당하는지 결정
- ✓ 엔티티 타입과 관계 타입들에 필요한 애트리뷰트들을 식별하고, 각 애트리뷰트가 가질 수 있는 값들의 집합을 식별
- ✓ 엔티티 타입들을 위한 기본 키를 식별
- ✓ 응용을 위한 ER 스키마 다이어그램을 그림
- ✓ ER 스키마 다이어그램이 응용에 대한 요구사항과 부합되는지 검사
- ✓ ER 스키마 다이어그램을 DBMS에서 사용되는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

5.2 ER 모델(계속)

□ 본 책의 ER 표기법의 요약



A Large ER Diagram



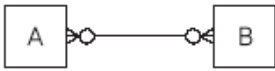
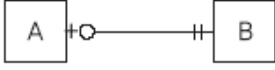

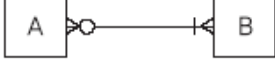


5.2 ER 모델(계속)

□ 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 표기법에도 여러 가지 변형들이 존재함

5.2 ER 모델(계속)

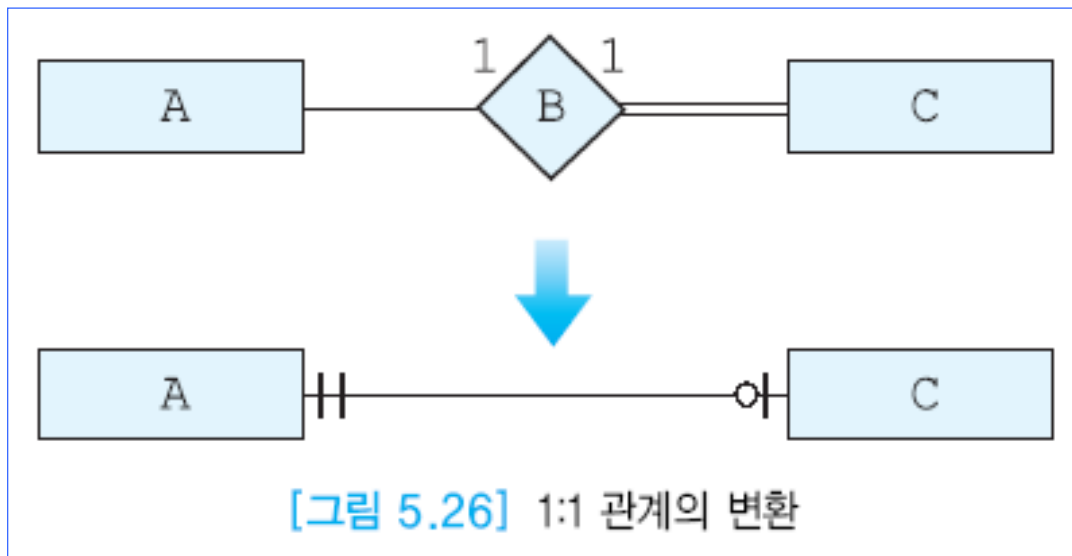
	1:1 관계. 엔티티 A의 각 인스턴스는 엔티티 B의 0 또는 1개의 인스턴스와 연관됨. 엔티티 B의 각 인스턴스는 엔티티 A의 0 또는 1개의 인스턴스와 연관됨.
	1:N 관계. 엔티티 A의 각 인스턴스는 엔티티 B의 0개 이상의 인스턴스와 연관됨. 엔티티 B의 각 인스턴스는 엔티티 A의 0 또는 1개의 인스턴스와 연관됨.
	M:N 관계. 엔티티 A의 각 인스턴스는 엔티티 B의 0개 이상의 인스턴스와 연관됨. 엔티티 B의 각 인스턴스는 엔티티 A의 0 개 이상의 인스턴스와 연관됨.
	1:1 관계. 엔티티 A의 각 인스턴스는 엔티티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

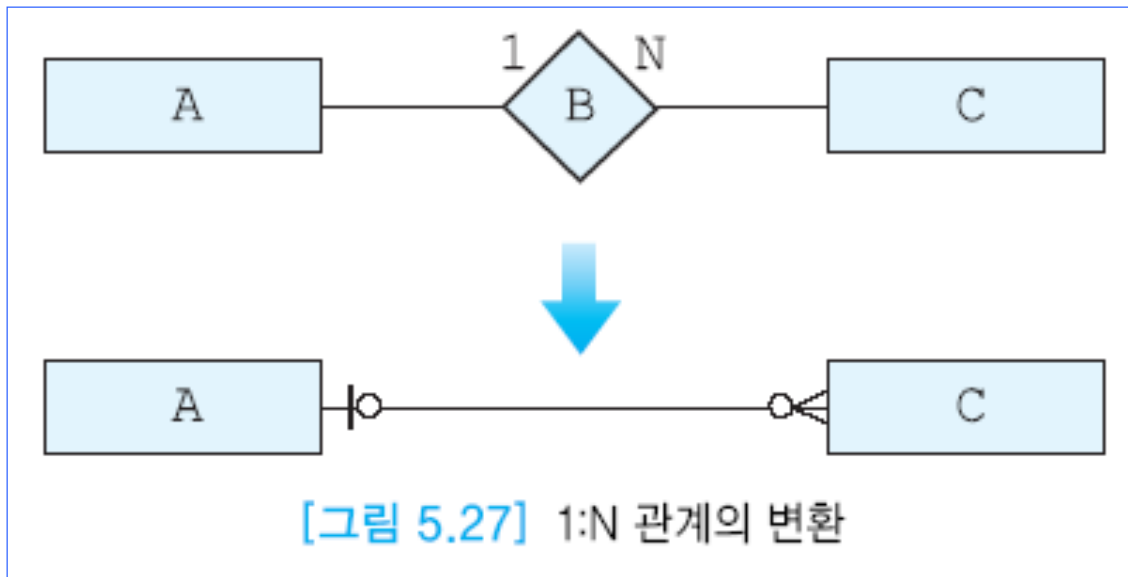
5.2 ER 모델(계속)

- 본 책의 표기법을 새발 표기법으로 표현하는 방법
 - ✓ 1:1 관계



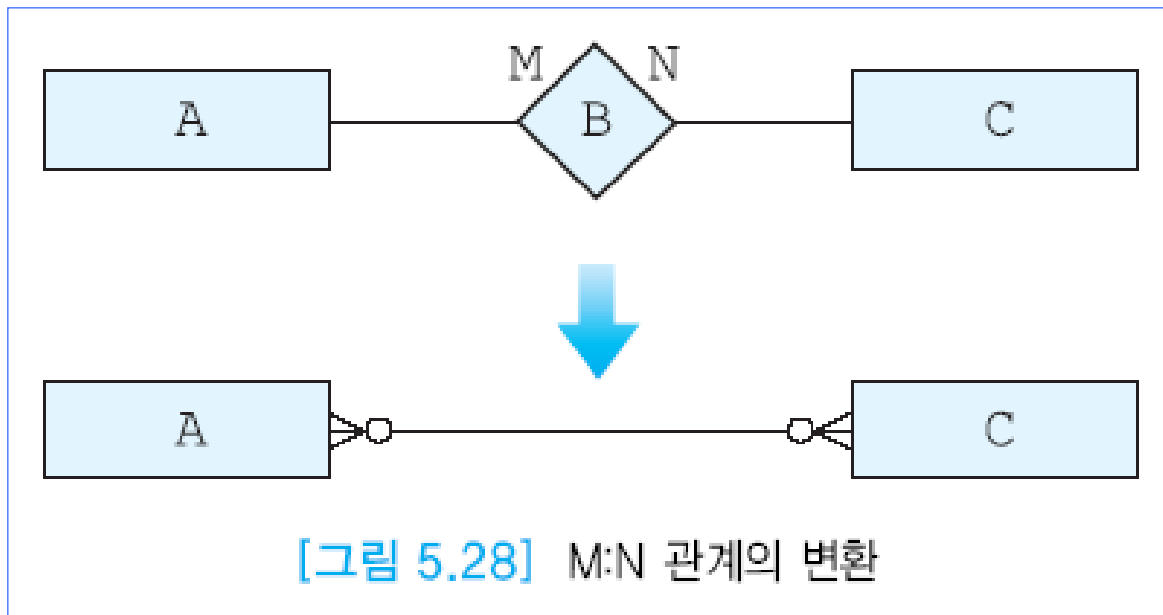
5.2 ER 모델(계속)

- ❑ 본 책의 표기법을 새발 표기법으로 표현하는 방법(계속)
 - ✓ 1:N 관계



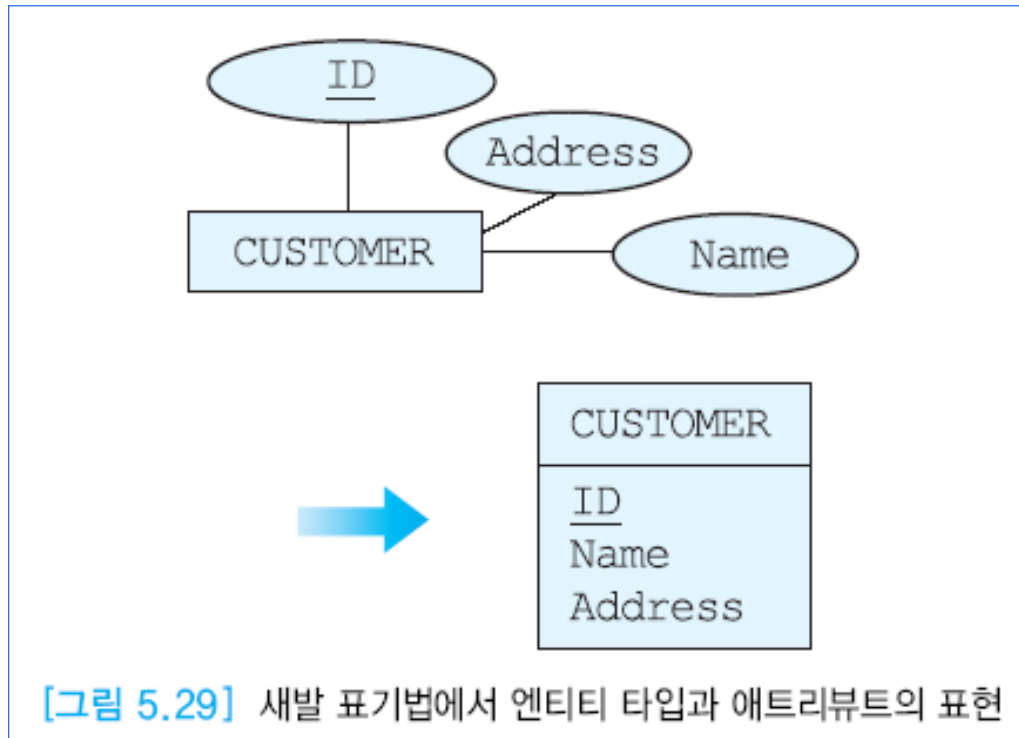
5.2 ER 모델(계속)

- ❑ 본 책의 표기법을 새발 표기법으로 표현하는 방법(계속)
 - ✓ M:N 관계

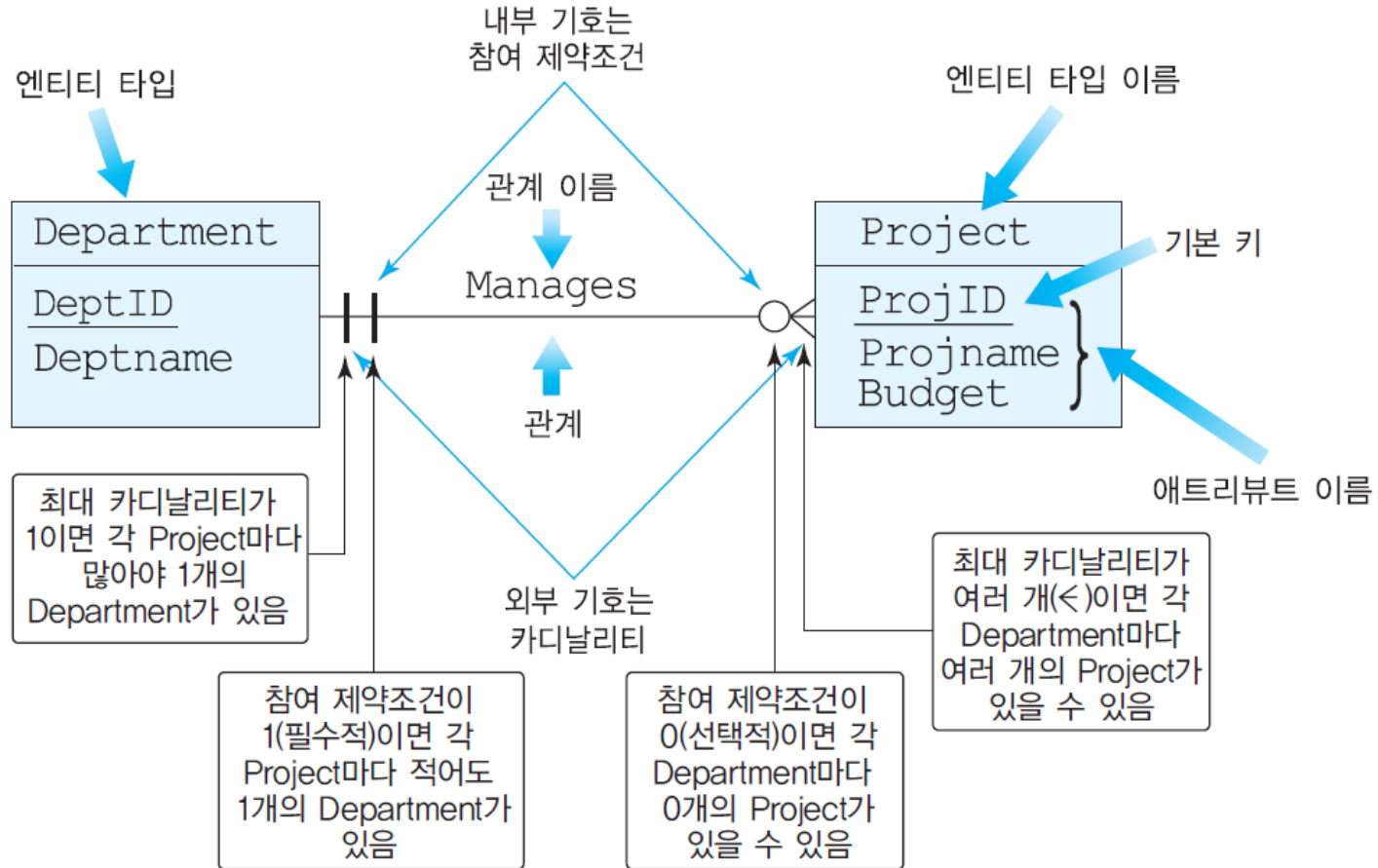


5.2 ER 모델(계속)

- ❑ 본 책의 표기법을 새발 표기법으로 표현하는 방법(계속)
 - ✓ 엔티티 타입과 애트리뷰트



5.2 ER 모델(계속)



[그림 5.30] 새발 표기법의 요약

5.3 데이터베이스 설계 사례

□ 기업에서 흔히 볼 수 있는 작은 세계에 관한 요구사항

- ✓ 회사에는 다수의 직원들이 재직
- ✓ 각 직원에 대해서 직원번호(고유함), 이름, 직책, 급여, 주소를 저장. 주소는 시, 구, 동으로 세분하여 나타냄
- ✓ 각 직원은 0명 이상의 부양가족을 가질 수 있음. 한 부양가족은 두 명 이상의 직원에게 속하지 않음. 각 부양가족에 대해서 부양가족의 이름과 성별을 저장
- ✓ 회사는 여러 개의 프로젝트들을 진행. 각 프로젝트에 대해서 프로젝트번호(고유함), 이름, 예산, 프로젝트가 진행되는 위치를 나타냄. 한 프로젝트는 여러 위치에서 진행될 수 있음. 각 프로젝트마다 여러 명의 직원들이 일함. 각 직원은 여러 프로젝트에서 근무할 수 있음. 각 직원이 해당 프로젝트에서 어떤 역할을 수행하고, 얼마 동안 근무해 왔는가를 나타냄. 각 프로젝트마다 한 명의 프로젝트 관리자가 있음. 한 직원은 두 개 이상의 프로젝트의 관리자가 될 수는 없음. 프로젝트 관리자 임무를 시작한 날짜를 기록

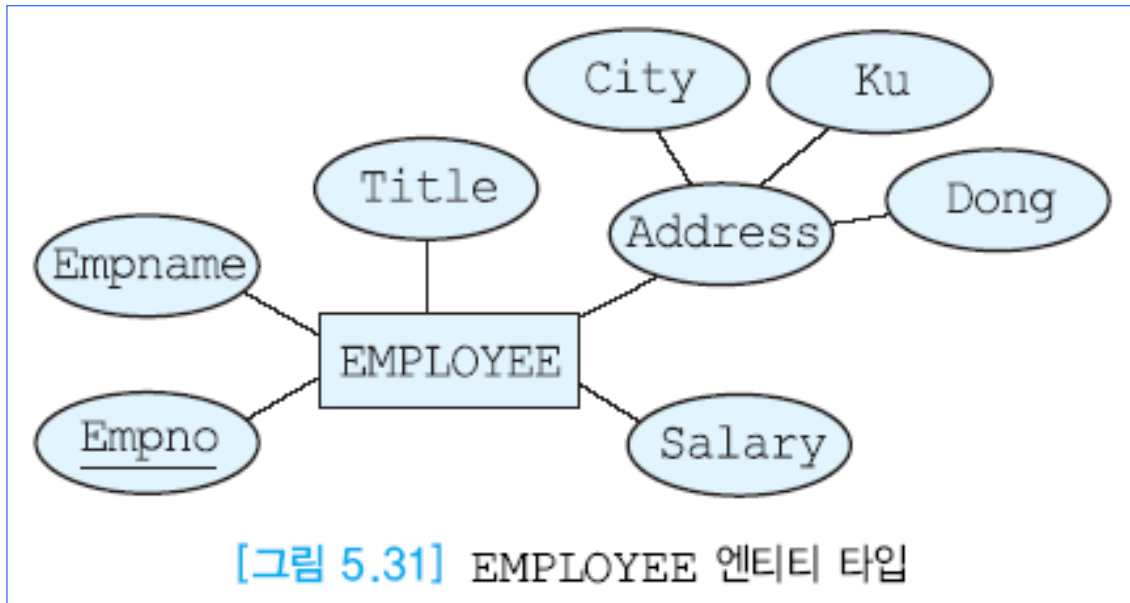
5.3 데이터베이스 설계 사례(계속)

□ 기업에서 흔히 볼 수 있는 작은 세계에 관한 요구사항(계속)

- ✓ 각 사원은 한 부서에만 속함. 각 부서에 대해서 부서번호(고유함), 이름, 부서가 위치한 층을 나타냄
- ✓ 각 프로젝트에는 부품들이 필요. 한 부품이 두 개 이상의 프로젝트에서 사용될 수 있음. 하나의 부품은 다른 여러 개의 부품들로 이루어질 수 있음. 각 부품에 대해서 부품번호(고유함), 이름, 가격, 그 부품이 다른 부품들을 포함하는 경우에는 그 부품들에 관한 정보도 나타냄
- ✓ 각 부품을 공급하는 공급자들이 있음. 한 명의 공급자는 여러 가지 부품들을 공급할 수 있고, 각 부품은 여러 공급자들로부터 공급될 수 있음. 각 공급자에 대해서 공급자번호(고유함), 이름, 신용도를 나타냄. 각 공급자에 대해서 그 공급자가 어떤 부품을 어떤 프로젝트에 얼마나 공급하는가를 나타냄

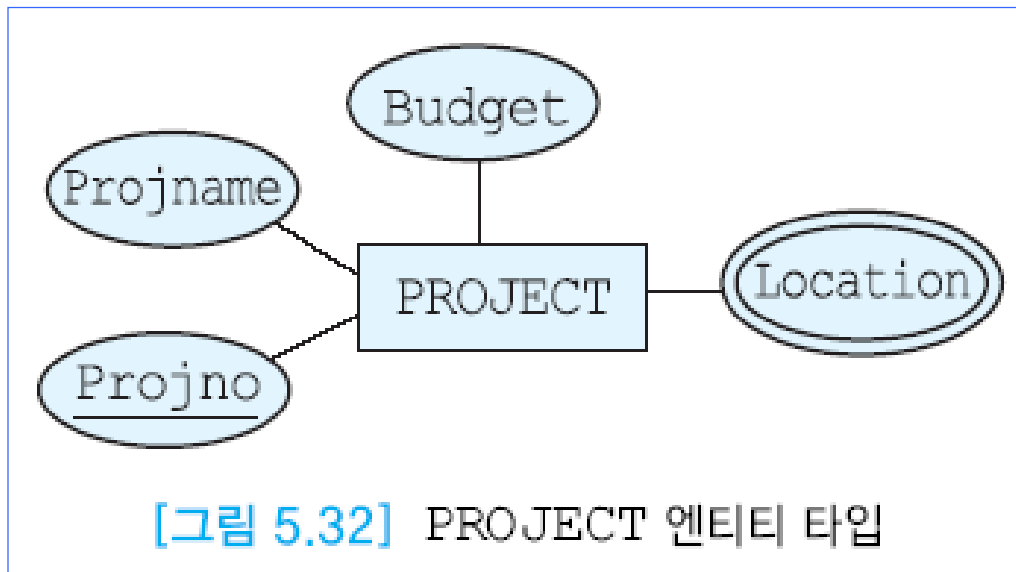
5.3 데이터베이스 설계 사례(계속)

□ 엔티티 타입 및 애트리뷰트들을 식별



5.3 데이터베이스 설계 사례(계속)

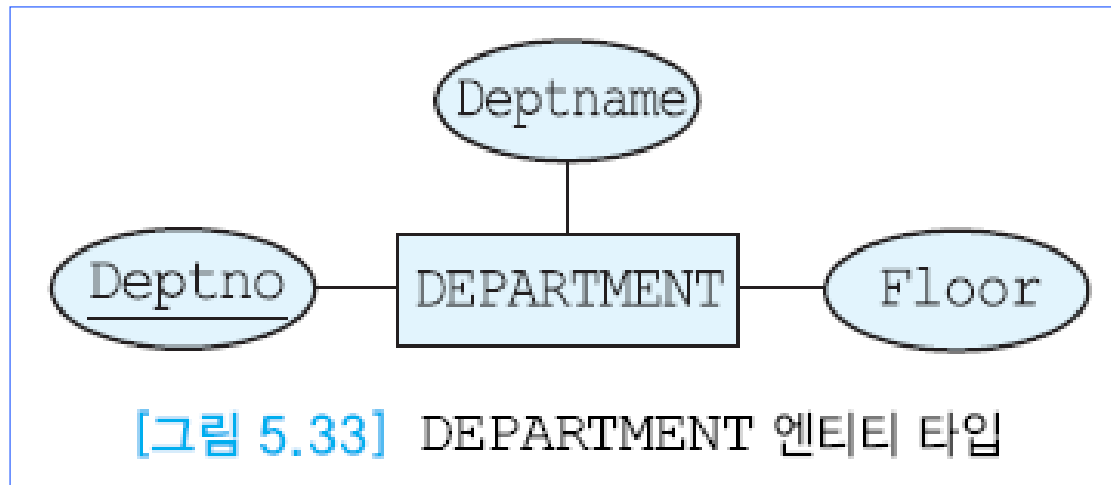
□ 엔티티 타입 및 애트리뷰트들을 식별(계속)



ER 모델 보고 풀거
ex) location은 무슨 애트리뷰트

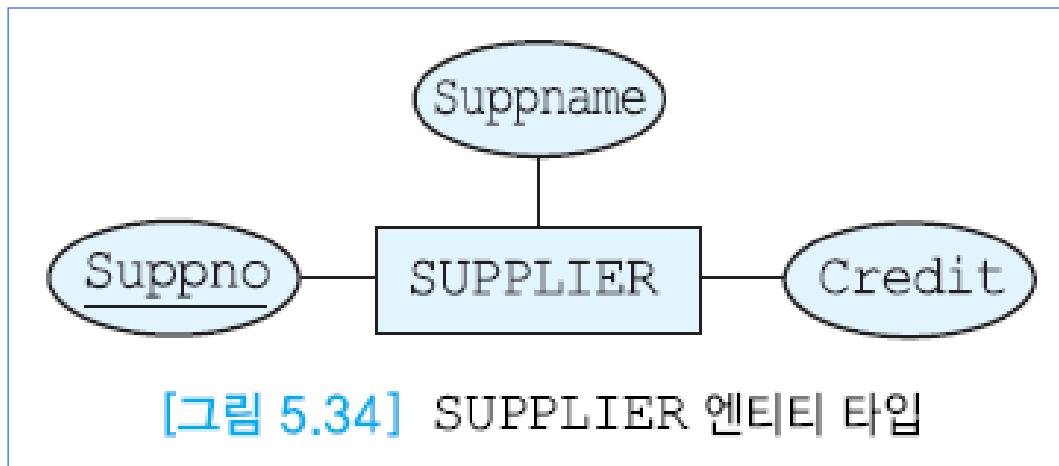
5.3 데이터베이스 설계 사례(계속)

□ 엔티티 타입 및 애트리뷰트들을 식별(계속)



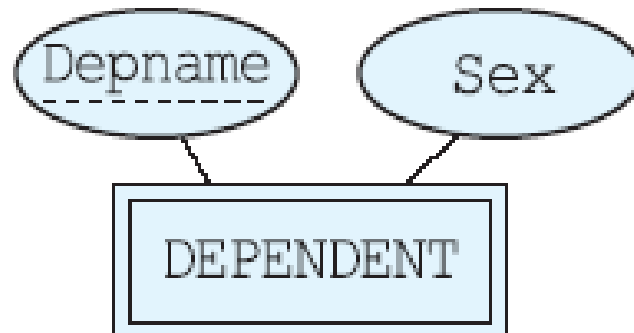
5.3 데이터베이스 설계 사례(계속)

□ 엔티티 타입 및 애트리뷰트들을 식별(계속)



5.3 데이터베이스 설계 사례(계속)

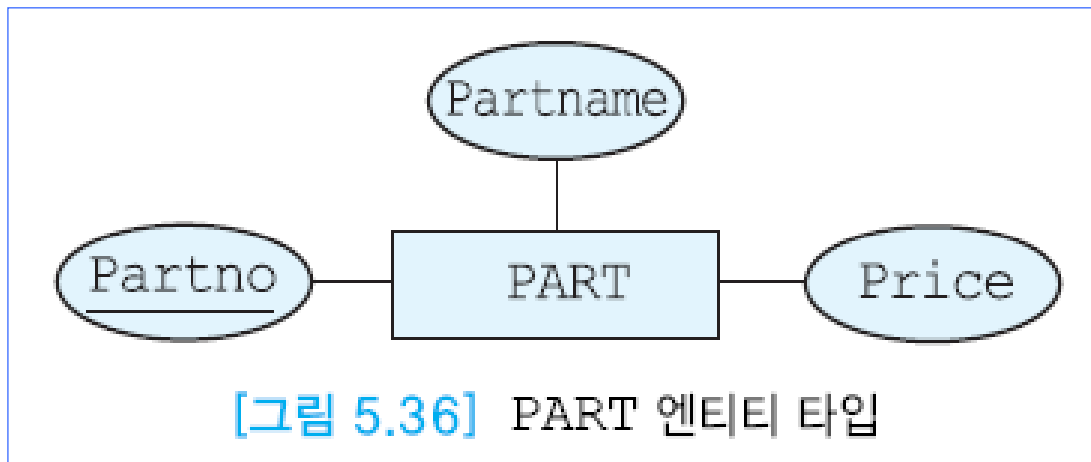
□ 엔티티 타입 및 애트리뷰트들을 식별(계속)



[그림 5.35] DEPENDENT 엔티티 타입

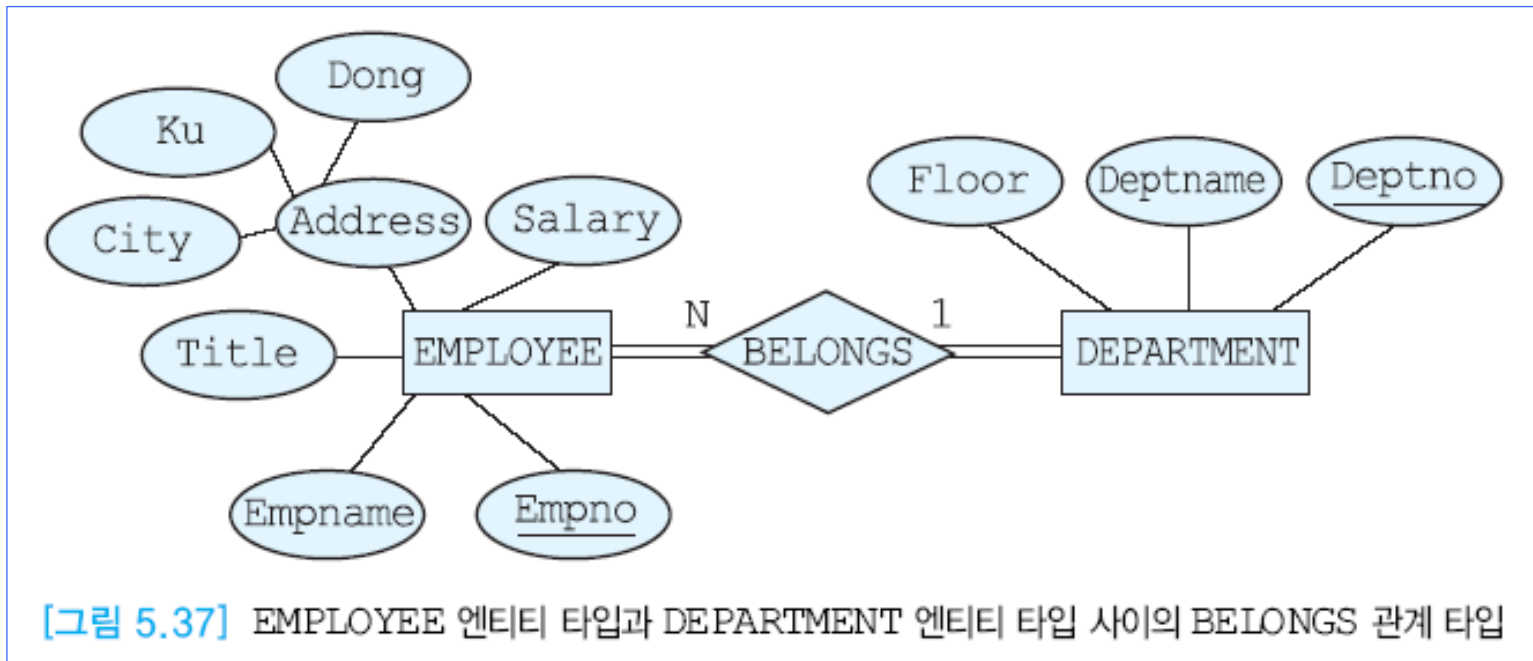
5.3 데이터베이스 설계 사례(계속)

□ 엔티티 타입 및 애트리뷰트들을 식별(계속)



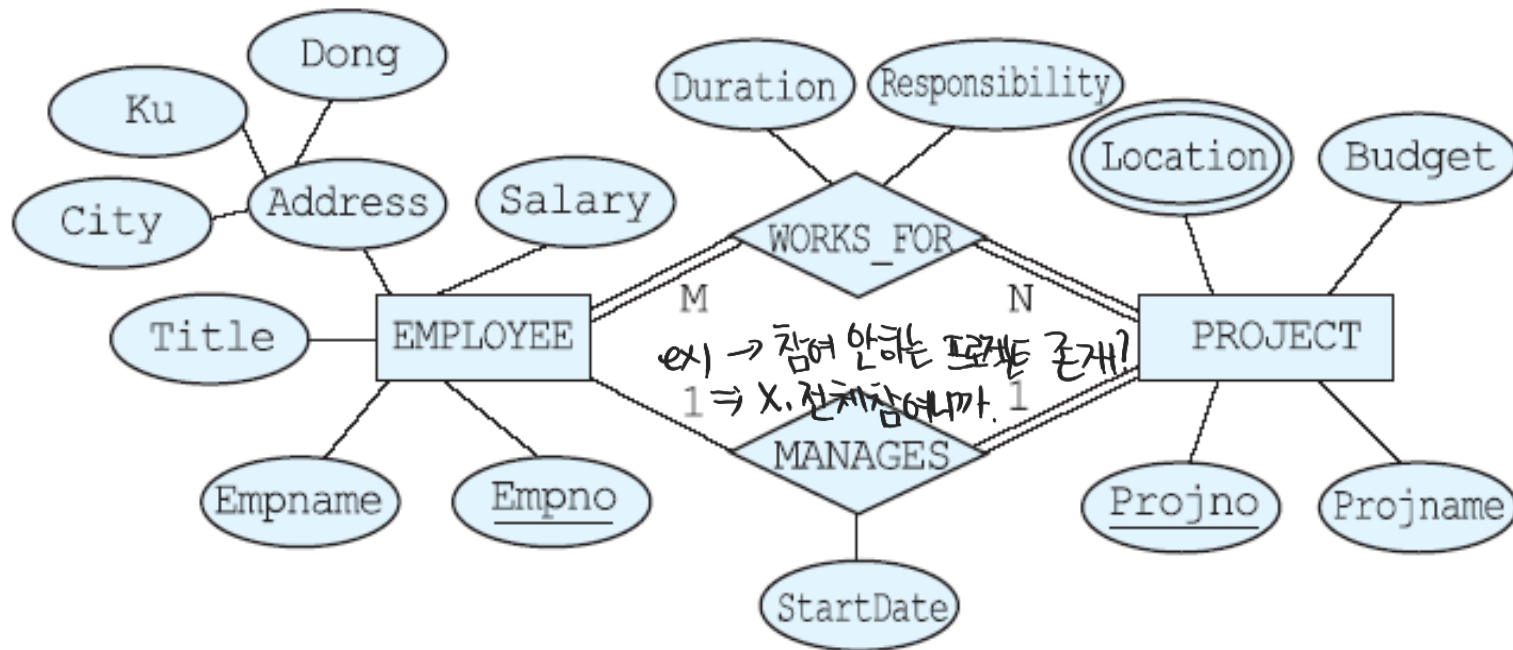
5.3 데이터베이스 설계 사례(계속)

□ 관계와 애트리뷰트들을 식별



5.3 데이터베이스 설계 사례(계속)

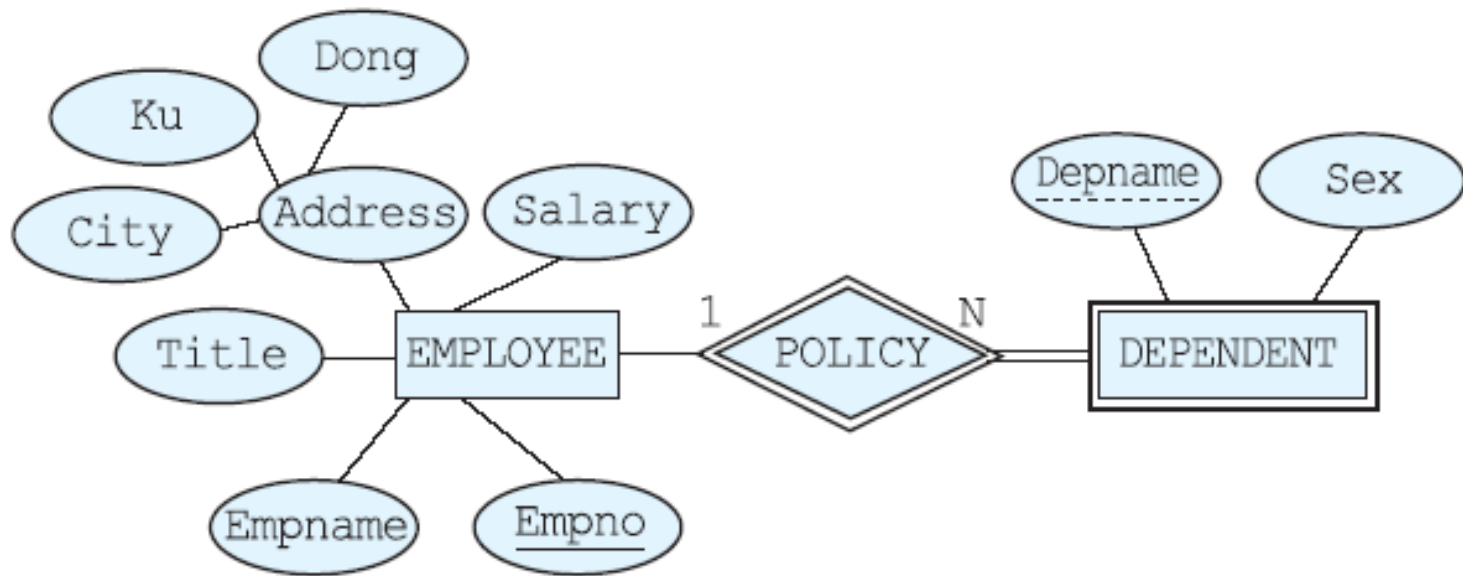
□ 관계와 애트리뷰트들을 식별(계속)



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

5.3 데이터베이스 설계 사례(계속)

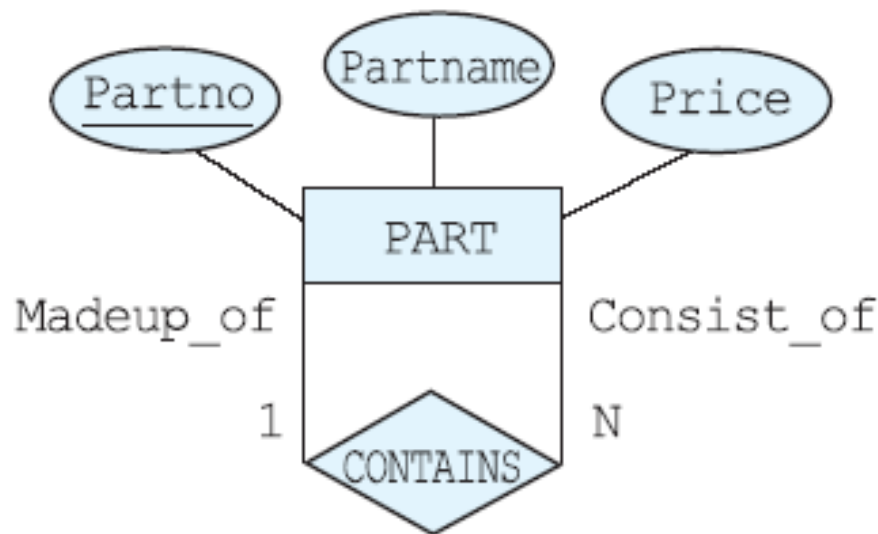
□ 관계와 애트리뷰트들을 식별(계속)



[그림 5.39] EMPLOYEE 엔티티 타입과 DEPENDENT 엔티티 타입 사이의 약한 관계 타입

5.3 데이터베이스 설계 사례(계속)

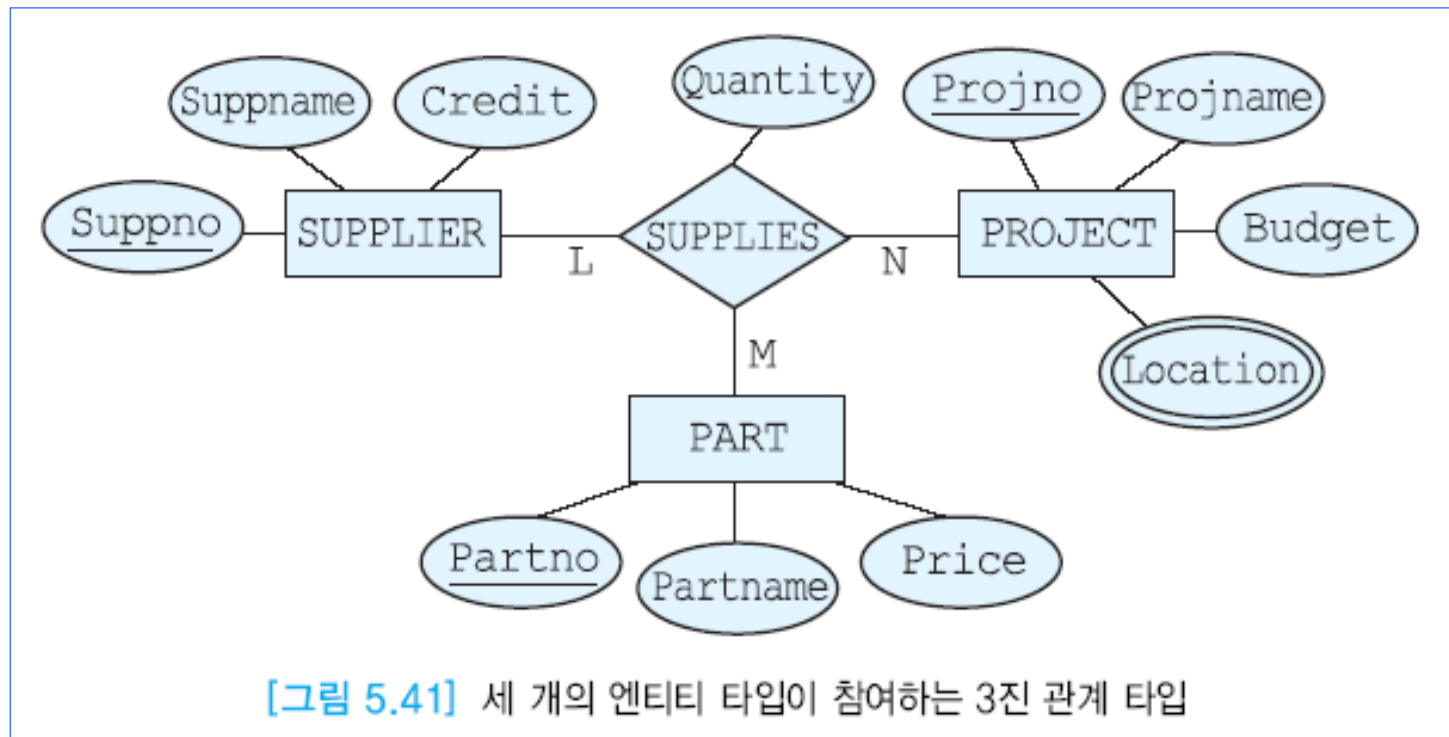
□ 관계와 애트리뷰트들을 식별(계속)



[그림 5.40] PART 엔티티 타입이 두 번 참여하는 순환적 관계 타입

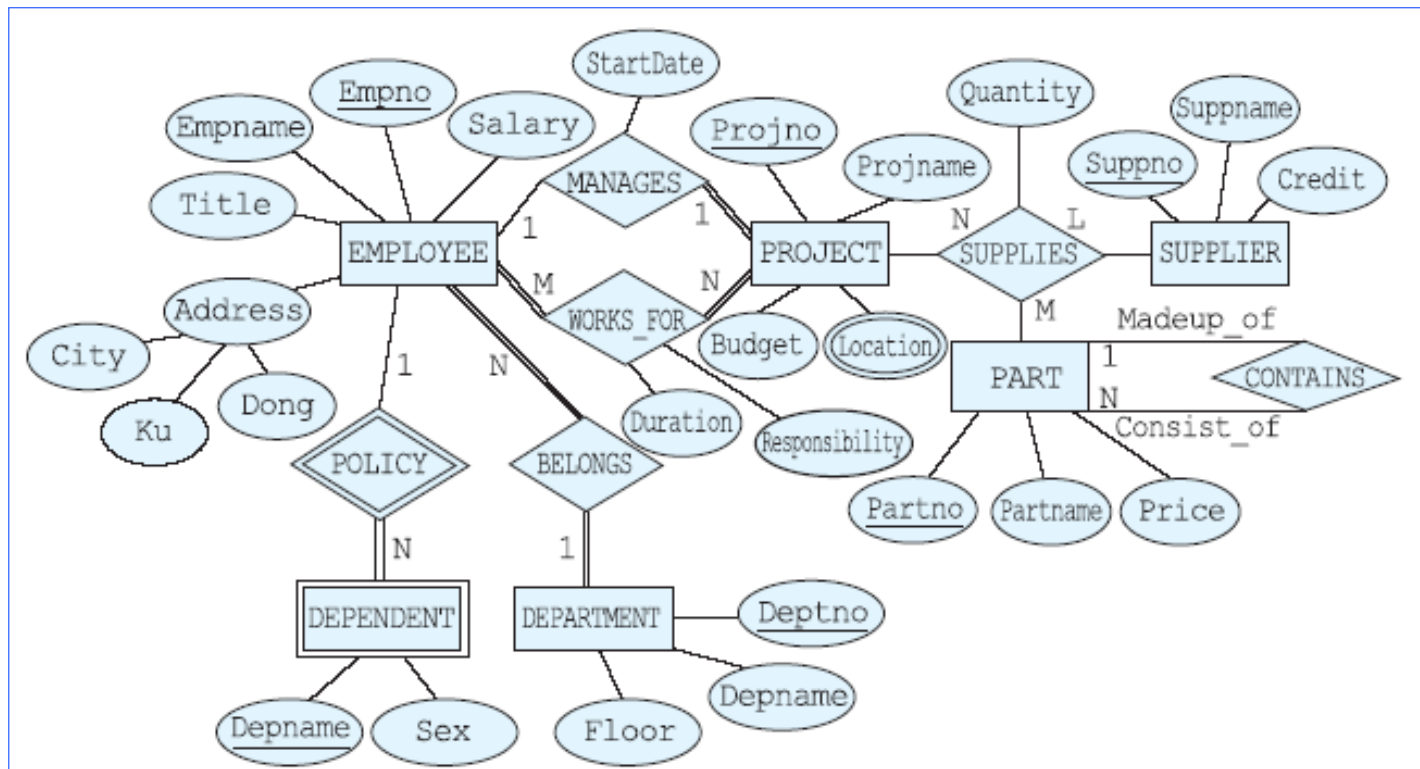
5.3 데이터베이스 설계 사례(계속)

□ 관계와 애트리뷰트들을 식별(계속)



5.3 데이터베이스 설계 사례(계속)

□ 관계와 애트리뷰트들을 식별(계속)



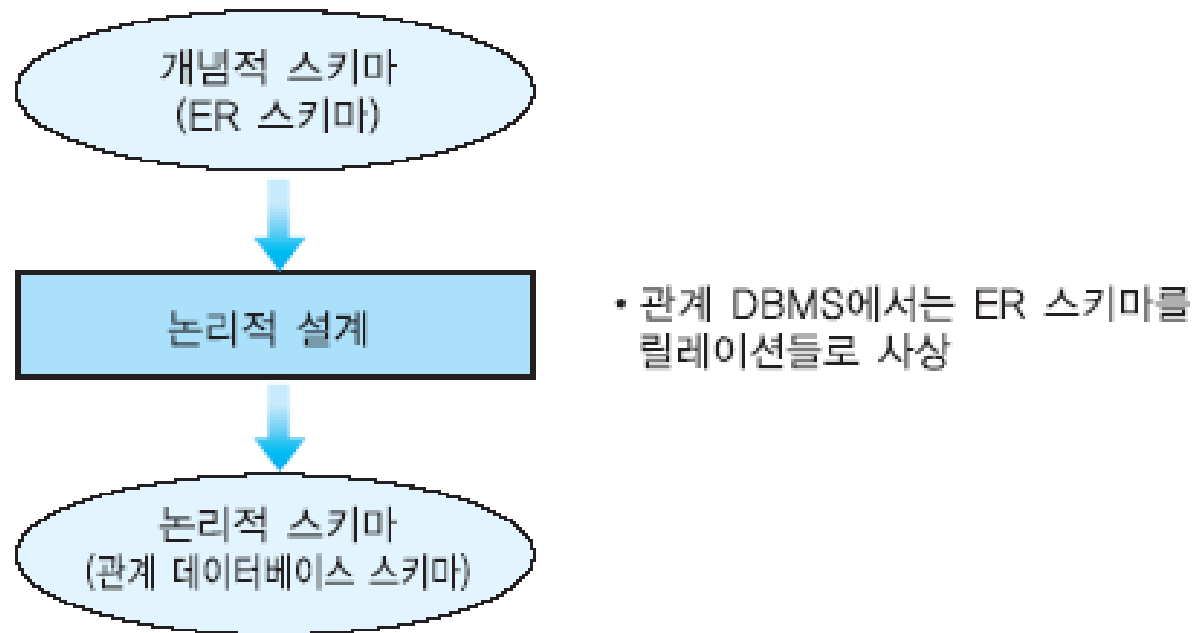
[그림 5.42] 회사의 ER 스키마 다이어그램

5.4 ER 스키마를 관계 모델의 릴레이션으로 사상

□ ER 스키마를 관계 모델의 릴레이션으로 사상

- ✓ 논리적 설계 단계에서는 ER 스키마를 관계 데이터 모델의 릴레이션들로 사상함
- ✓ ER 스키마에는 엔티티 타입과 관계 타입이 존재하지만 관계 데이터베이스에는 엔티티 타입과 관계 타입을 구분하지 않고 릴레이션들만 있음
- ✓ 릴레이션으로 사상할 대상이 ER 스키마에서 엔티티 타입인지 또는 관계 타입인지, 엔티티 타입이라면 정규 엔티티 타입인지 또는 약한 엔티티 타입인지, 관계 타입이라면 2진 관계 타입인지 3진 이상의 관계 타입인지, 애트리뷰트가 단일 값 애트리뷰트인지 또는 다치 애트리뷰트인지 등에 따라 사상하는 방법이 달라짐
- ✓ ER 모델을 릴레이션들로 사상하는 7개의 단계로 이루어진 알고리즘

5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)



[그림 5.43] 논리적 설계

5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)

〈표 5.4〉 알고리즘의 각 단계에서 릴레이션으로 사상되는 ER 스키마의 대상

사상할 대상	알고리즘의 단계
엔티티 타입과 단일 값 애트리뷰트	단계 1: 정규 엔티티 타입
	단계 2: 약한 엔티티 타입
2진 관계 타입	단계 3: 2진 1:1 관계 타입
	단계 4: 정규 2진 1:N 관계 타입
	단계 5: 2진 M:N 관계 타입
3진 이상의 관계 타입	단계 6: 3진 관계 타입
다치 애트리뷰트	단계 7: 다치 애트리뷰트

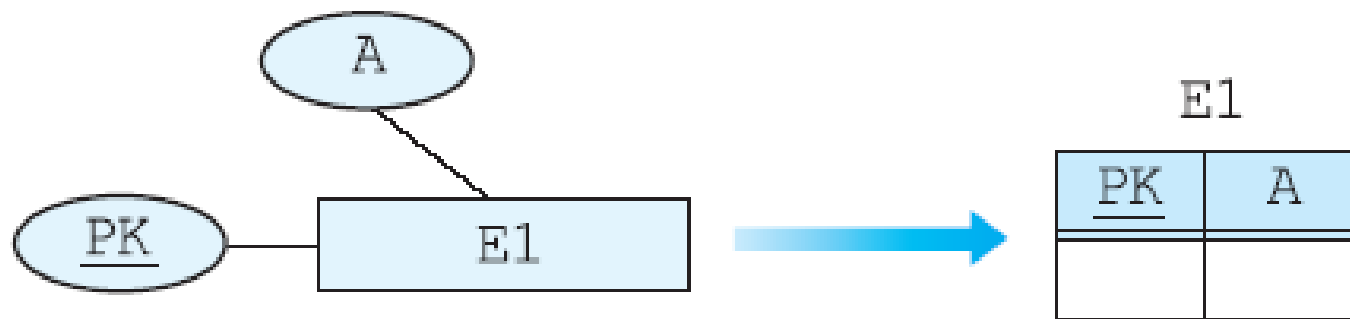
5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)

□ ER-관계 사상 알고리즘

단계 1: 정규 엔티티 타입과 단일 값 애트리뷰트

- ✓ ER 스키마의 각 정규 엔티티 타입 E에 대해 하나의 릴레이션 R을 생성함
- ✓ E에 있던 단순 애트리뷰트들을 릴레이션 R에 모두 포함시킴
- ✓ E에서 복합 애트리뷰트는 그 복합 애트리뷰트를 구성하는 단순 애트리뷰트들만 릴레이션 R에 포함시킴
- ✓ E의 기본 키가 릴레이션 R의 기본 키가 됨

5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)



[그림 5.44] 정규 엔티티 타입을 릴레이션으로 사상

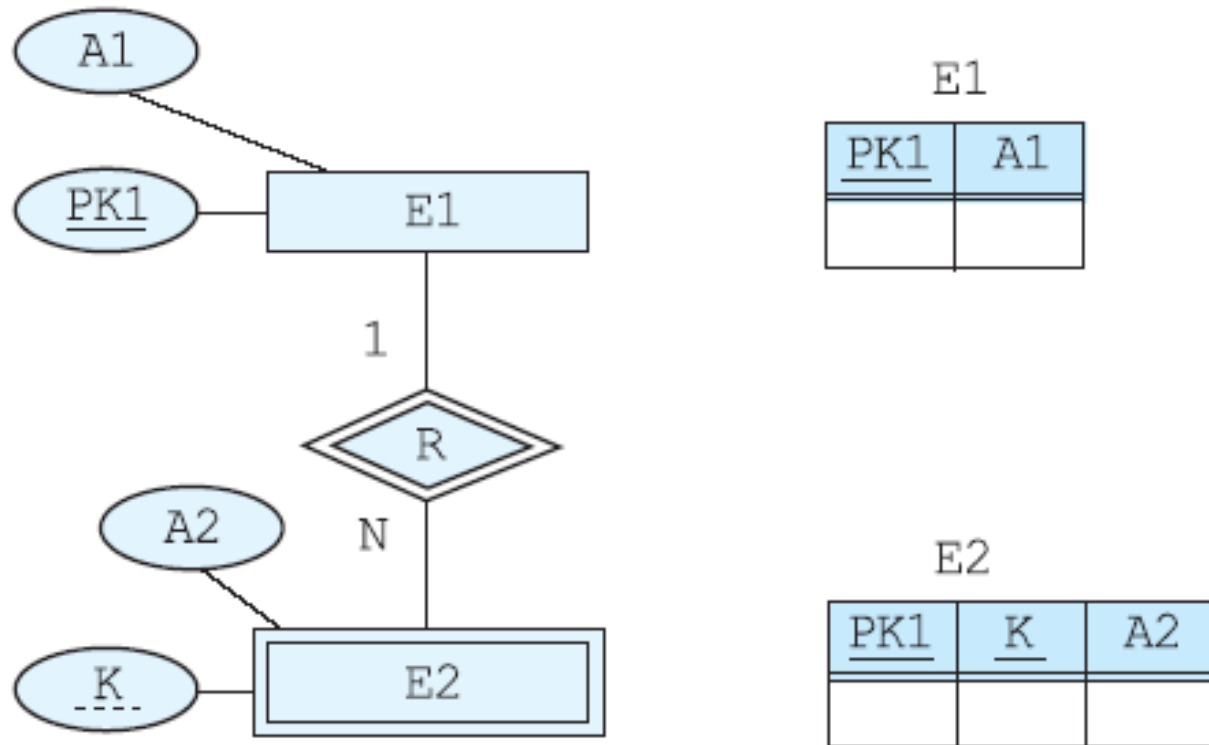
5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)

□ ER-관계 사상 알고리즘(계속)

단계 2: 약한 엔티티 타입과 단일 값 애트리뷰트

- ✓ ER 스키마에서 소유 엔티티 타입 E를 갖는 각 약한 엔티티 타입 W에 대하여 릴레이션 R을 생성함
- ✓ W에 있던 모든 단순 애트리뷰트들을 릴레이션 R에 포함시킴
- ✓ 소유 엔티티 타입에 해당하는 릴레이션의 기본 키를 약한 엔티티 타입에 해당하는 릴레이션에 외래 키로 포함시킴
- ✓ 약한 엔티티 타입에 해당하는 릴레이션 R의 기본 키는 약한 엔티티 타입의 부분 키와 소유 엔티티 타입에 해당하는 릴레이션을 참조하는 외래 키의 조합으로 이루어짐

5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)



[그림 5.45] 약한 엔티티 타입을 릴레이션으로 사상

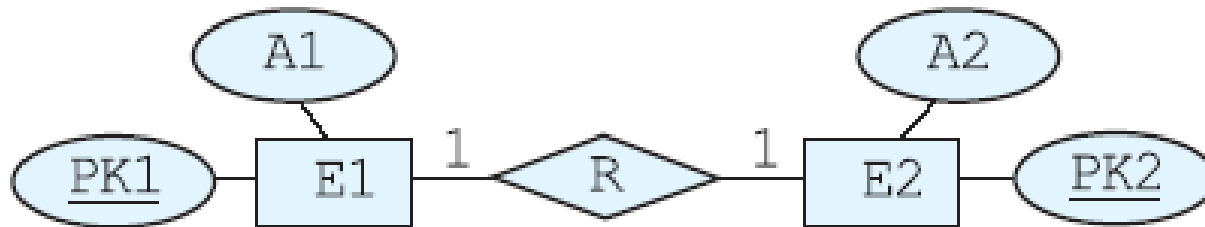
5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)

□ ER-관계 사상 알고리즘(계속)

단계 3: 2진 1:1 관계 타입

- ✓ ER 스키마의 각 2진 1:1 관계 타입 R에 대하여, R에 참여하는 엔티티 타입에 대응되는 릴레이션 S와 T를 찾음
- ✓ S와 T 중에서 한 릴레이션을 선택하여, 만일 S를 선택했다면 T의 기본 키를 S에 외래 키로 포함시킴
- ✓ S와 T 중에서 관계 타입에 완전하게 참여하는 릴레이션을 S의 역할을 하는 릴레이션으로 선택함
- ✓ 관계 타입 R이 가지고 있는 모든 단순 애트리뷰트(복합 애트리뷰트를 갖고 있는 경우에는 복합 애트리뷰트를 구성하는 단순 애트리뷰트)들을 S에 대응되는 릴레이션에 포함시킴
- ✓ 두 엔티티 타입이 관계 타입 R에 완전하게 참여할 때는 두 엔티티 타입과 관계 타입을 하나의 릴레이션으로 합치는 방법도 가능함

5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)



방법 1 :

<u>PK1</u>	A1
K1	

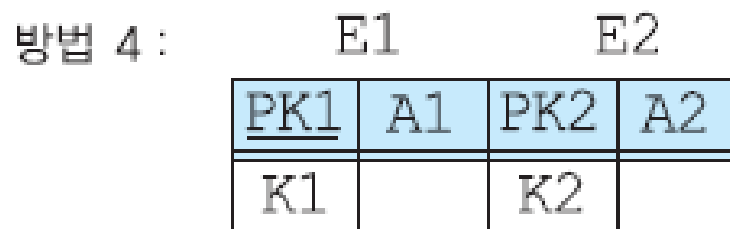
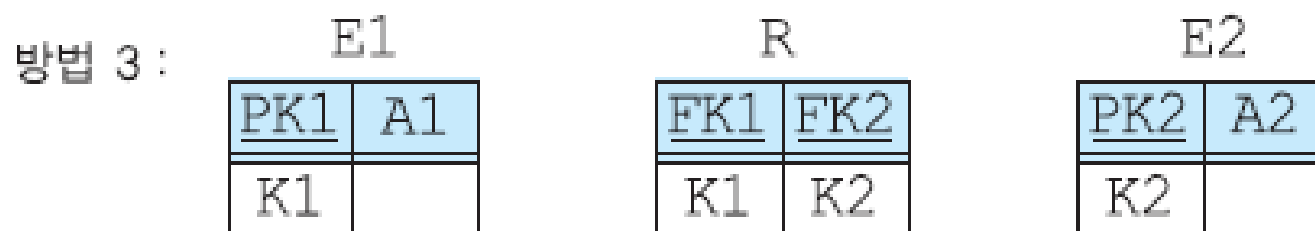
<u>PK2</u>	A2	FK1
K2		K1

방법 2 :

<u>PK1</u>	A1	FK2
K1		K2

<u>PK2</u>	A2
K2	

5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)



[그림 5.46] 2진 1:1 관계 타입을 릴레이션으로 사상

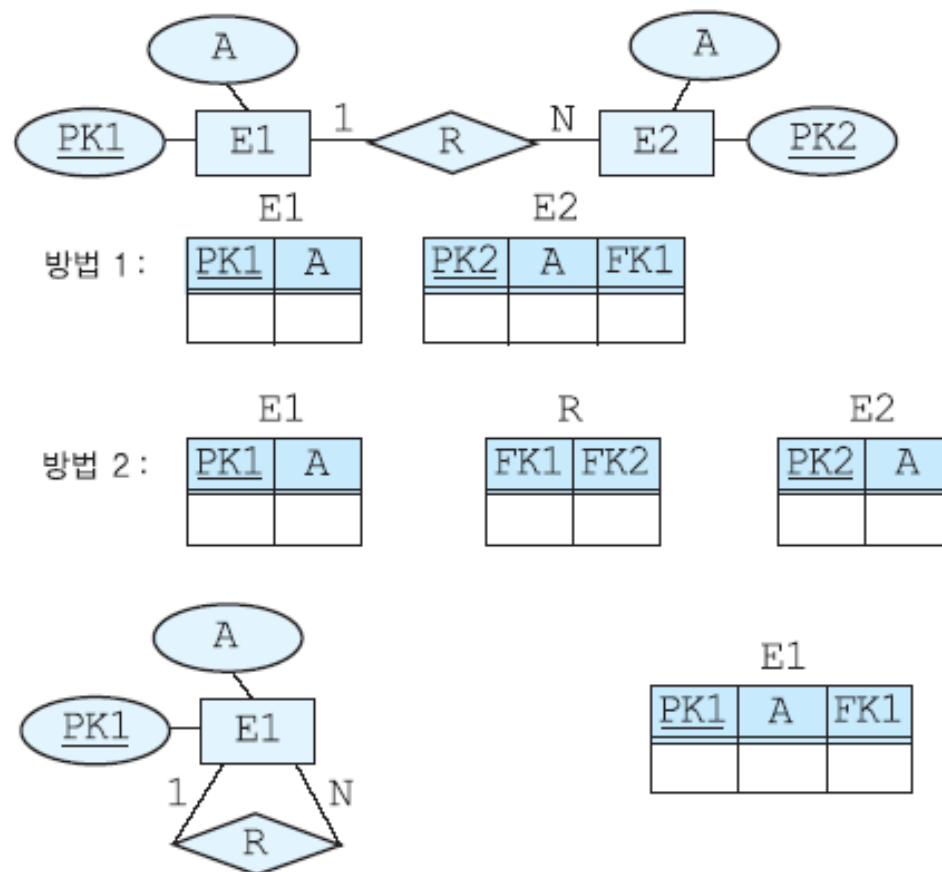
5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)

□ ER-관계 사상 알고리즘(계속)

단계 4: 정규 2진 1:N 관계 타입

- ✓ 정규 2진 1:N 관계 타입 R에 대하여 N측의 참여 엔티티 타입에 대응되는 릴레이션 S를 찾음
- ✓ 관계 타입 R에 참여하는 1측의 엔티티 타입에 대응되는 릴레이션 T의 기본 키를 릴레이션 S에 외래 키로 포함시킴
- ✓ N측의 릴레이션 S의 기본 키를 1측의 릴레이션 T에 외래 키로 포함시키면 애트리뷰트에 값들의 집합이 들어가거나 정보의 중복이 많이 발생함
- ✓ 관계 타입 R이 가지고 있는 모든 단순 애트리뷰트(복합 애트리뷰트를 갖고 있는 경우에는 복합 애트리뷰트를 구성하는 단순 애트리뷰트)들을 S에 해당하는 릴레이션에 포함시킴

5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)



[그림 5.47] 정규 2진 1:N 관계 타입을 릴레이션으로 사상

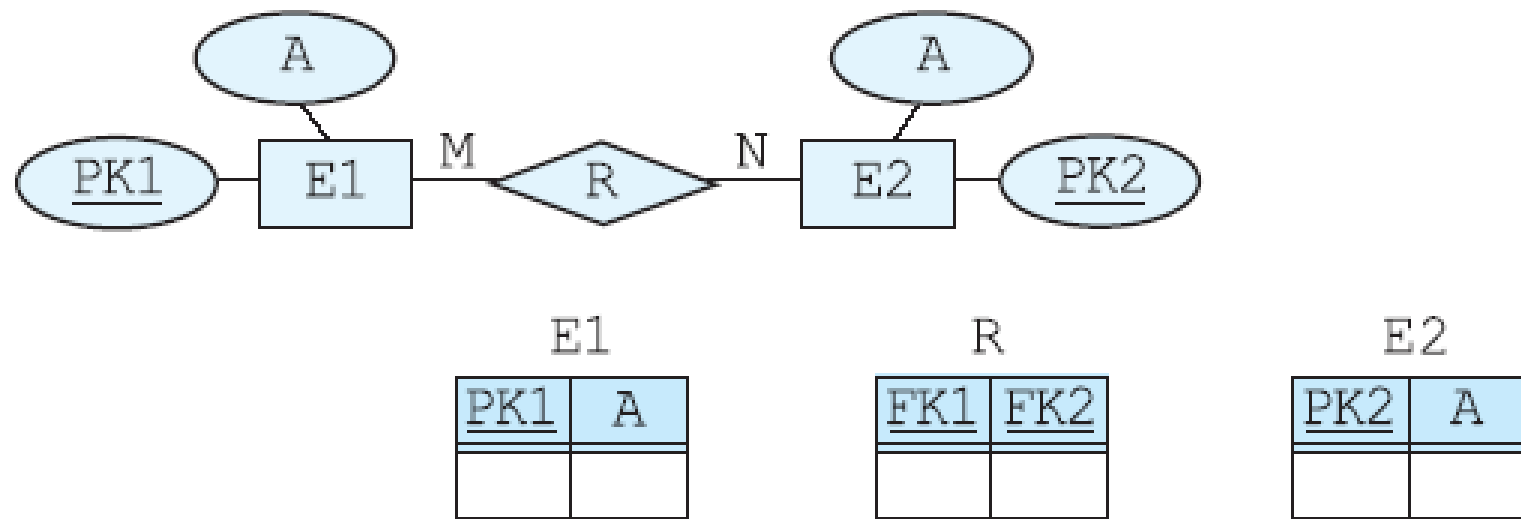
5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)

□ ER-관계 사상 알고리즘(계속)

단계 5: 2진 M:N 관계 타입

- ✓ 2진 M:N 관계 타입 R에 대해서는 릴레이션 R을 생성함
- ✓ 참여 엔티티 타입에 해당하는 릴레이션들의 기본 키를 릴레이션 R에 외래 키로 포함시키고, 이들의 조합이 릴레이션 R의 기본 키가 됨
- ✓ 관계 타입 R이 가지고 있는 모든 단순 애트리뷰트(복합 애트리뷰트를 갖고 있는 경우에는 복합 애트리뷰트를 구성하는 단순 애트리뷰트)들을 릴레이션 R에 포함시킴

5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)



[그림 5.48] 2진 M:N 관계 타입을 릴레이션으로 사상

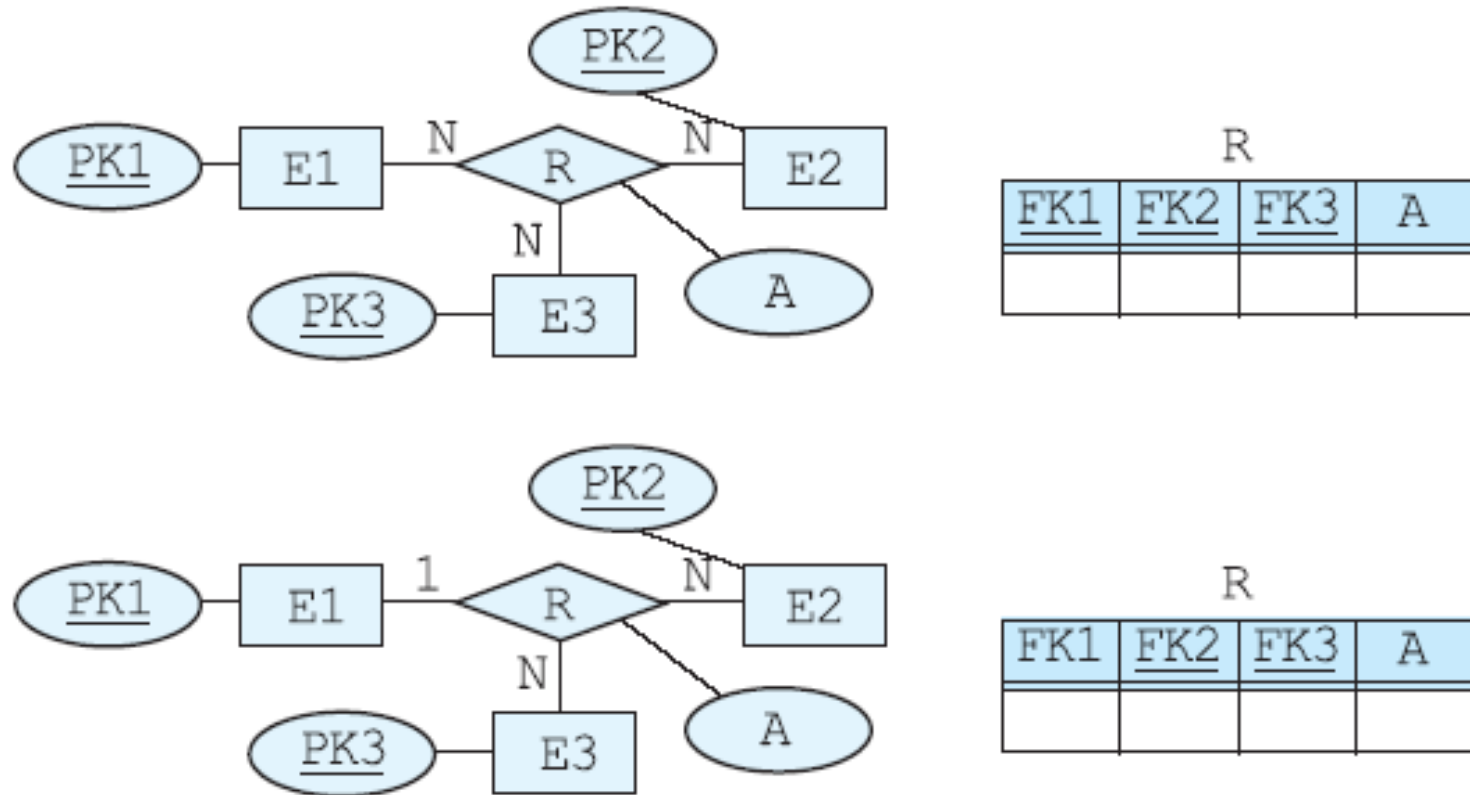
5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)

□ ER-관계 사상 알고리즘(계속)

단계 6: 3진 이상의 관계 타입

- ✓ 3진 이상의 각 관계 타입 R에 대하여 릴레이션 R을 생성함
- ✓ 관계 타입 R에 참여하는 모든 엔티티 타입에 대응되는 릴레이션들의 기본 키를 릴레이션 R에 외래 키로 포함시킴
- ✓ 관계 타입 R이 가지고 있는 모든 단순 애트리뷰트(복합 애트리뷰트를 갖고 있는 경우에는 복합 애트리뷰트를 구성하는 단순 애트리뷰트)들을 릴레이션 R에 포함시킴
- ✓ 일반적으로 외래 키들의 조합이 릴레이션 R의 기본 키가 됨
- ✓ 관계 타입 R에 참여하는 엔티티 타입들의 카디널리티가 1:N:N이면 카디널리티가 1인 릴레이션의 기본 키를 참조하는 외래 키를 제외한 나머지 외래 키들의 조합이 릴레이션 R의 기본 키가 됨

5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)



[그림 5.49] 3진 이상의 관계 타입을 릴레이션으로 사상

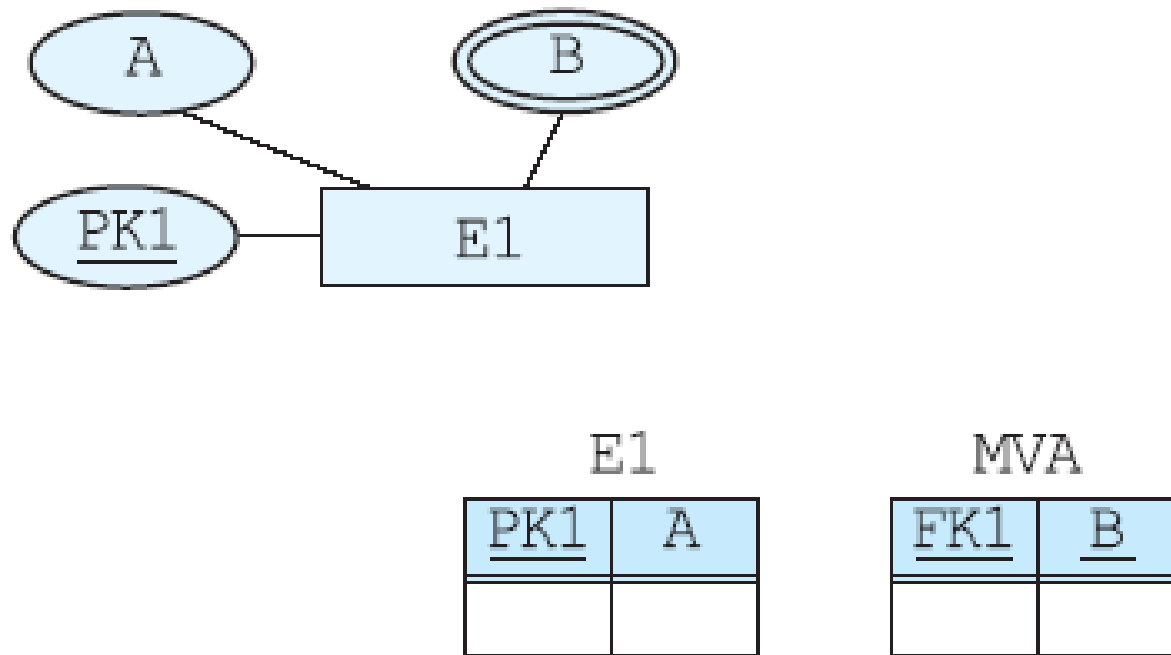
5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)

□ ER-관계 사상 알고리즘(계속)

단계 7: 다치 애트리뷰트

- ✓ 각 다치 애트리뷰트에 대하여 릴레이션 R을 생성함
- ✓ 다치 애트리뷰트에 해당하는 애트리뷰트를 릴레이션 R에 포함시키고, 다치 애트리뷰트를 애트리뷰트로 갖는 엔티티 타입이나 관계 타입에 해당하는 릴레이션의 기본 키를 릴레이션 R에 외래 키로 포함시킴
- ✓ 릴레이션의 R의 기본 키는 다치 애트리뷰트와 외래 키의 조합

5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)



[그림 5.50] 다치 애트리뷰트를 릴레이션으로 사상

5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)

❑ 데이터베이스 설계 사례에 알고리즘 적용

단계 1: 정규 엔티티 타입과 단일 값 애트리뷰트

EMPLOYEE(Empno, Empname, Title, City, Ku, Dong,
Salary)

PROJECT(Projno, Projname, Budget)

DEPARTMENT(Deptno, Deptname, Floor)

SUPPLIER(Suppno, Suppname, Credit)

PART(Partno, Partname, Price)

5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)

❑ 데이터베이스 설계 사례에 알고리즘 적용(계속)

단계 2: 약한 엔티티 타입과 단일 값 애트리뷰트

DEPENDENT(Empno, Depname, Sex)

단계 3: 2진 1:1 관계 타입

PROJECT(Projno, Projname, Budget, StartDate, Manager)

단계 4: 정규 2진 1:N 관계 타입

EMPLOYEE(Empno, Empname, Title, City, Ku, Dong,
Salary, Dno)

PART(Partno, Partname, Price, Subpartno)

5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)

❑ 데이터베이스 설계 사례에 알고리즘 적용(계속)

단계 5: 2진 M:N 관계 타입

WORKS_FOR(Empno, Projno, Duration, Responsibility)

단계 6: 3진 이상의 관계 타입

SUPPLY(Suppno, Projno, Partno, Quantity)

단계 7: 다치 애트리뷰트

PROJ_LOC(Projno, Location)

5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)

- ❑ 회사 ER 스키마는 관계 데이터베이스에서 총 9개의 릴레이션으로 사상되었음

EMPLOYEE(Empno, Empname, Title, City, Ku, Dong,
Salary, Dno)

PROJECT(Projno, Projname, Budget, StartDate, Manager)

DEPARTMENT(Deptno, Deptname, Floor)

SUPPLIER(Suppno, Suppname, Credit)

PART(Partno, Partname, Price, Subpartno)

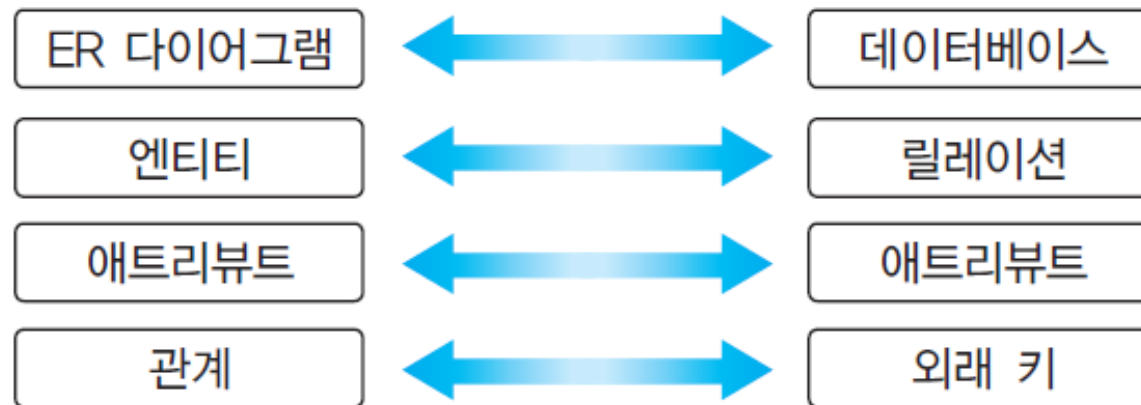
DEPENDENT(Empno, Depname, Sex)

WORKS_FOR(Empno, Projno, Duration, Responsibility)

SUPPLY(Suppno, Projno, Partno, Quantity)

PROJ_LOC(Projno, Location)

5.4 ER 스키마를 관계 모델의 릴레이션으로 사상(계속)



[그림 5.51] ER 개념과 데이터베이스 개념들의 대응 관계