엔터티-관계를 사용한 데이터 모델링

데이터 베이스 설계와 개념적 데이터 모델

➢ 데이터베이스 설계 과정

- 요구사항 수집 및 분석
  - 데이터 요구사항 (data requirement) 이해 및 문서화
  - **기능적 요구사항** (functional requirements) 명시: 사용자 정의 연산 또는 트렌잭션으로 구성 (검색과 갱신에 대한 연산도 포함)
- 개념 스키마 (conceptual schema) 설계
  - 사용자들의 데이터 요구사항을 기술: 데이터 타입, 관계, 제약조건들을 자세히 설명
  - 구현에 필요한 자세한 사항은 불포함: 데이터베이스 설계자가 기억장치나 구현에 구애받지 않고 개념적 데이터 베이스 설계를 쉽게 할수 있도록 도와줌
  - 개념 스키마가 모든 식별된 기능적 요구사항들을 만족시켰는지 확인 가능

➢ 데이터 베이스의 실제 구현

- 상용 DBMS를 활용하여 데이터베이스를 실제로 구현
- 논리적 설계 (logical design) 또는 데이터 모델 사상 (data model mapping)
  - 고수준 데이터 모델인 개념 스키마를 구현 데이터 모델로 변환해야 함

➢ 물리적 설계

- 데이터베이스 파일들에 대한 내부 저장 구조, 파일 구성, 인덱스, 접근 경로, 물리적 설계 파라미터 명시

데이터 베이스 설계와 개념적 데이터 모델

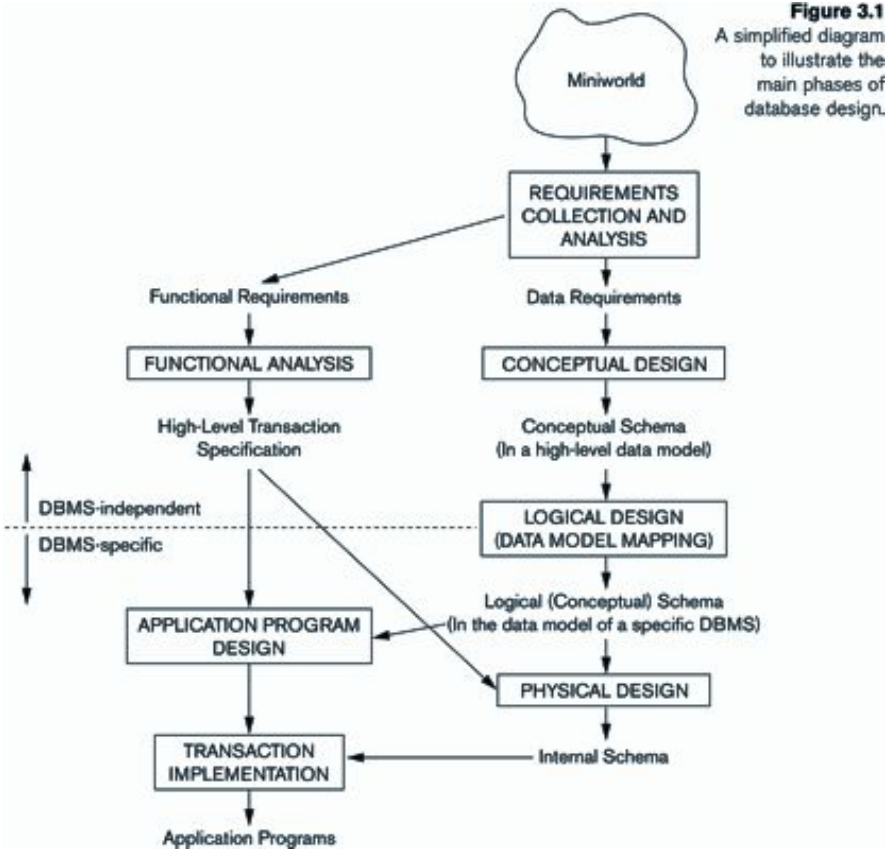


Figure 3.1
A simplified diagram to illustrate the main phases of database design.

데이터 베이스 예제: COMPANY

➢ 데이터 베이스 예제: COMPANY

- 회사의 사원, 부서, 프로젝트들에 대한 정보를 저장
- 데이터베이스 설계자가 제시한 요구사항
  - 회사는 여러 부성들로 구성되고 각 부서마다 고유한 이름, 번호, 부서를 관리하는 특정 사원이 존재, 사원이 부서를 관리하기 시작한 날짜도 유지함
  - 한 부서는 여러 위치에 있을 수 있음
  - 한 부서는 여러개의 프로젝트들을 관리
  - 각 프로젝트들은 고유한 이름, 고유한 번호, 한 개의 위치를 가짐
  - 각 사원에 대해서 이름, 사회보장번호, 주소, 급여, 성별, 생년월일을 저장
  - 한 사원은 한 부서에 속하지만, 여러 프로젝트들에 관여할 수 있음
  - 한 사원이 관리하는 프로젝트는 그 사원이 소속된 부서가 관리하는 프로젝트가 아니어도 무방함
  - 한 부서의 각 사원이 프로젝트를 위해 현재 일하는 주당 근무 시간을 기록
  - 각 사원의 직속상사도 유지
  - 보험 목적을 위해 각 사원의 부양 가족을 기록, 각 부양 가족에 대해서 이름, 성별, 생년월일, 사원과의 관계를 기록

Entity-Relationship (ER) 모델 개념

➢ 엔터티와 애트리뷰트

- 엔터티 (Entity): 실세계에서 독립적으로 존재하는 실체
  - 실제로 존재하는 객체: 사람, 자동차, 집 등
  - 개념적으로 존재하는 객체: 직업, 대학의 강좌 등
- 애트리뷰트 (Attribute): 엔터티를 기술하는 속성
  - 애트리뷰트 값은 데이터베이스에 저장되는 데이터의 주요 부분이 됨
  - 예) 사원의 애트리뷰트: 이름, 나이, 주소, 급여, 직업

➢ 애트리뷰트 유형

- 단순 (simple) vs. 복합 (composite)
- 단일값 (single-valued) vs. 다치 (multivalued)
- 저장된 (stored) vs. 유도된 (derived)

애트리뷰트 유형

➢ Simple vs. Composite attributes

- 단순 애트리뷰트 (simple attribute) 혹은 원자 애트리뷰트 (atomic attribute): 더이상 나눌 수 없는 애트리뷰트
- 복합 애트리뷰트 (composite attribute): 더 작은 구성요소들로 나눌 수 있음
  - 각 구성 요소는 그 자체로 독립적인 의미를 가진 기본적인 애트리뷰트를 나타냄 (예: 주소)
  - 계층 형성 가능
- 복합 애트리뷰트는 사용자가 **하나의 단위** 혹은 **특정 구성 요소만** 참조하게 되는 경우 유용함
  - 개별적인 구성요소를 참조할 필요가 없다면 전체를 하나의 단순 애트리뷰트로 설계 가능

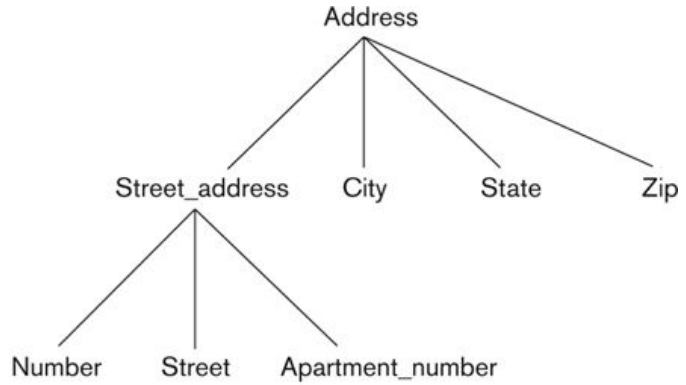


Figure 3.4
A hierarchy of composite attributes.

애트리뷰트 유형

➢ Simple vs. Composite attributes

- 단순 애트리뷰트 (simple attribute) 혹은 원자 애트리뷰트 (atomic attribute): 더이상 나눌 수 없는 애트리뷰트
- 복합 애트리뷰트 (composite attribute): 더 작은 구성요소들로 나눌 수 있음
  - 각 구성 요소는 그 자체로 독립적인 의미를 가진 기본적인 애트리뷰트를 나타냄 (예: 주소)
  - 계층 형성 가능
- 복합 애트리뷰트는 사용자가 **하나의 단위** 혹은 **특정 구성 요소만** 참조하게 되는 경우 유용함
  - 개별적인 구성요소를 참조할 필요가 없다면 전체를 하나의 단순 애트리뷰트로 설계 가능

➢ Single-valued vs. Multi-valued attributes

- 단일값 애트리뷰트 (single-valued attribute): 특정한 앤터티에서 **단일 값**을 가짐 (예: 나이)
- 다치 애트리뷰트 (multi-valued attribute): 한 엔터티에서 애트리뷰트가 여러 값을 가질 수 있음
  - 예) 자동차의 color, 사람의 college_degree
- 하한과 상한의 **범위**를 가질 수 있음

애트리뷰트 유형

➢ Stored vs. Derived attributes

- 두 개 이상의 애트리뷰트 값들이 상호 연관될 수 있음
  - 사람 (entity): birth date (+현재 날짜) → age
  - 저장된 애트리뷰트 (stored attribute): birth date, 유도된 애트리뷰트 (derived attribute): age
- 어떤 애트리뷰트 값은 관련된 엔터티 값으로부터 유도될 수 있음
  - 부서 (entity)의 number_of_enployees는 그 부서에서 일하는 사원들 숫자들을 세는 것으로부터 유도 가능

➢ Null value (널 값)

- 한 엔터티의 특정 애트리뷰트에 대해 적용할 값이 없는 경우
  - 적용할 수 없음 (예: Apartment_number)
  - 알려지지 않음: 애트리뷰트 값이 존재하지만 누락되었을 경우 (예: 키, 몸무게), 애트리뷰트 값이 존재하는지 여부를 알지 못할 경우 (예: Home_phone)

➢ Complex attributes

- 복합 에트리뷰트(complex attributes): composite attributes 와 multi-valued attributes 가 중첩된 경우
  - 예: PreviousDegrees of student = {PreviousDegrees(College, Year, Degree, Field)}

엔터티 타입과 키 애트리뷰트

➢ 엔터티 타입과 집합

- 엔터티 타입(entity type): 동일한 애트리뷰트들을 갖는 엔터티들의 집합
  - 이름과 애트리뷰트들의 리스트를 기술
  - in ER 다이아그램: 엔터티 타입의 **이름을 둘러싸는 사각형**
  - 엔터티들의 집합에 대한 **스키마(schema)** 또는 **내포(intension)**
- 엔터티 집합(entity set): 특정 엔터티 타입의 데이터베이스에 들어 있는 현재의 모든 엔터티의 집합을 포함
  - 엔터티 타입의 **외연 (extension)**

➢ 키 애트리뷰트 (key attribute)

- 엔터티 집합 내에서 각 엔터티 마다 서로 다른 값을 가지는 한개 이상의 애트리뷰트
  - Person(Entity type)에서의 SSN(Social Security Number) 애트리뷰트
- 키 애트리뷰트는 composite 일 수 있음
  - VehicleTagNumber(차 번호) = (Number + State)
- 엔터티 타입은 한 개 이상의 키를 가질 수 있음
  - Vehicle_ID, VehicleTagNumber
- in ER 다이아그램: **각 키**는 **밑줄**로 표시
- 키 제약조건은 특정 엔터티 집합에만 적용되는 거이 아니라 모든 시점의 엔터티 집합들에게

애트리뷰트의 값집합

- **값집합 (value set)**
  - 각 엔터티의 해당 애트리뷰트가 가질 수 있는 값들의 집합
  - 기본 데이터 형: integer, string, boolean, float, enumerated type, subrange
  - 추가적인 데이터 형: date, time

(a)

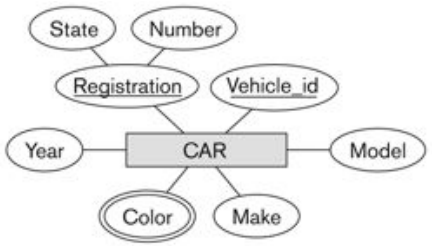


Figure 3.7
The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

(b)

CAR
Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

CAR_1
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR_2
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR_3
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

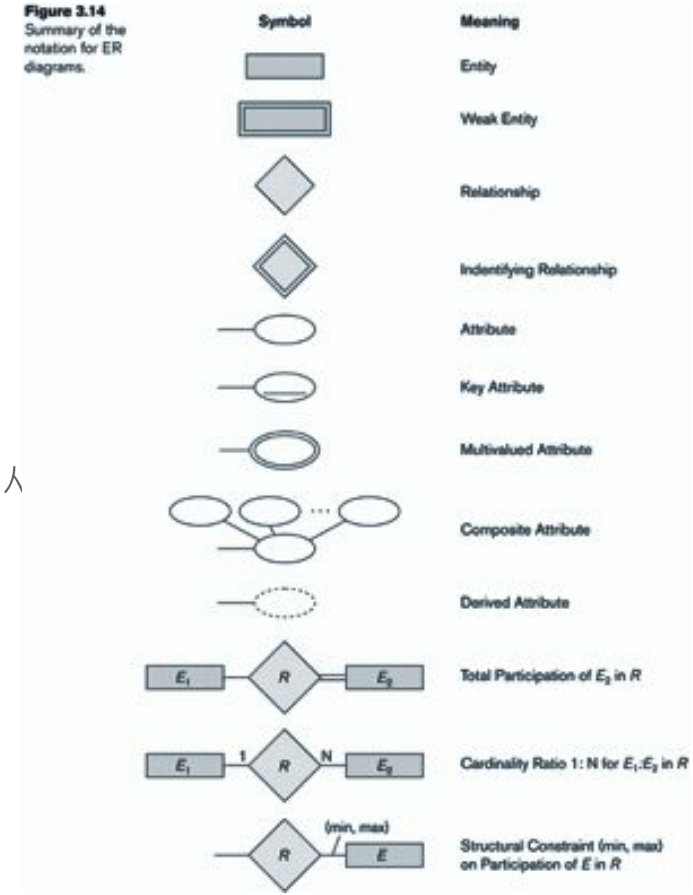
COMPANY 데이터베이스에 대한 초기 개념적 설계

➢ COMPANY 데이터베이스에 대한 요구 사항

- 엔터티 타입 **DEPARTMENT** 는 Name, Number, Locations, Manager, Manager_start_date 애트리뷰트들을 가짐, Locations 는 유일한 다치 애트리뷰트, Name과 Number 는 각 부서마다 고유
- 엔터티 타입 **PROJECT**는 Name, Number, Locations, Controlling_department 애트리뷰트를 가짐, Name과 Number 가 키 애트리뷰트
- 엔터티 타입 **EMPLOYEE** 는 Name, Ssn, Sex, Address, Salary, Birth_datte, Department, Supervisor 애트리뷰트들을 가짐, 협의에 따라 Name(First_name, Middle_initial, Last_name) 과 Address 는 composite attribute 가 될 수 있음
- 엔터티 타입 **DEPENDENT** 는 Employee, Dependent_name, Sex, Birth_date, Relationship(사원과의 관계) 애트리뷰트들을 가짐

COMPANY 데이터베이스 설계

- **초기 entity 타입**
  - DEPARTMENT
  - PROJECT
  - EMPLOYEE
  - DEPENDENT
  - 엔터티들은 직사각형으로 표시
- **애트리뷰트**
  - in ER 다이어그램: 애트리뷰트들은 **타원**으로 표시



COMPANY 데이터베이스 설계

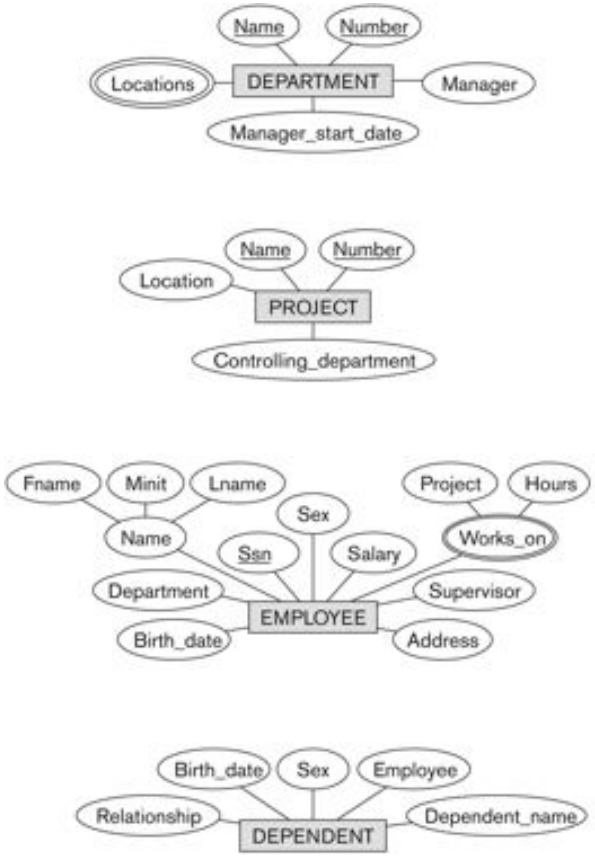


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

데이터베이스의 관계

➢ 관계 (relationship)

- 여러 엔터티 타입들 간에는 묵시적인 관계가 존재
- 구체적인 의미를 가지는 두개 이상의 서로 다른 엔터티들을 관련시킴
- 요구사항에서의 몇 가지는 관계로써 명시됨
  - DEPARTMENT 엔터티 타입의 Manager 애트리뷰트는 그 부서를 관리하는 사원을 참조
  - PROJECT 엔터티 타입의 Controlling_department 애트리뷰트는 그 프로젝트를 관리하는 부서를 참조
- 데이터베이스 **초기 설계**에서는 **애트리뷰트들로 표현**될 수 있지만 설계가 **개선**됨에 따라서 엔터티 타입들 간의 **관계로 변환**됨

➢ 관계와 관계 타입

- 관계 타입 (relationship type): **같은 엔터티 타입 사이의 관계**들은 **관계 타입**으로 정의됨
  - WORKS_ON 관계 타입: EMPLOYEEs 와 PROJECTSs 엔터티들의 참여함
  - MANAGES 관계타입: EMPLOYEEs 와 DEPARTMNETs 가 참여
- 관계 타입의 차수(degree): 참여하는 **엔터티 타입의 수**
  - Binary relationship, ternary relationship (Supply: 공급자, 부품, 프로젝트)

관계 타입과 관계 집합

- **관계 타입 (relationship type)**
  - 관계에 대한 스키마 기술
  - 관계 이름, 관계에 참여하는 엔터티들의 타입으로 구성
  - 특정한 관계 제약조건을 포함할 수 있음
  - in ER 다이아그램
    - 다이아몬드 모양의 박스로 표현
    - 참여하는 엔터티 타입들을 직선으로 연결
    - 관계 타입은 화살표로 나타내지는 않지만, 오니쪽에서 오른쪽, 위에서 아래로 읽도록 표시해야 함
- **관계 집합 (relationship set)**
  - 데이터베이스에 나타난 관계 인스턴스들의 현재 집합
  - 관계 타입의 현재 상태 (current state)

관계를 활용한 데이터베이스 개선

➢ 관계 타입의 도입

- 초기 엔터티 타입의 일부의 애트리뷰트들은 관계 타입으로 변환 가능
  - DEPARTMENT 의 Manager: MANAGES
  - EMPLOYEE 의 Works_on: WORKS_ON
  - EMPLOYEE 의 Departmnet: WORKS_FOR
- 일반적으로, 같은 엔터티 타입들이 참여하는 여러 관계 타입들이 존재할 수 있음
  - MANAGES 와 WORKS_FOR는 둘다 EMPLOYEE 와 DEPARTMENT 사이의 관계임
  - 하지만, 다른 관계 인스턴스들과 다른 의미를 가질 수 있음

➢ 관계 타입에서의 제약 조건

- 카디날리티 비율(cardinality constraint)
  - 1:1, 1:N, N:1, M:N
- 관계에 대한 존재 종속적 제약조건 (참여 제약조건, 최소 카디날리티 제약조건)
  - 부분 참여 (zero): 선택적 참여로 참여에 해당 관계에 대한 제한이 없는 경우 (**단선**)
  - 전체 참여 (one or more): 모든 엔터티가 최소한 한 개 혹은 정해진 수 이상의 해당 관계에 참여해야함 (**이중선**)
  - (min, max) 표기 가능

관계 종류 (N:1 관계)

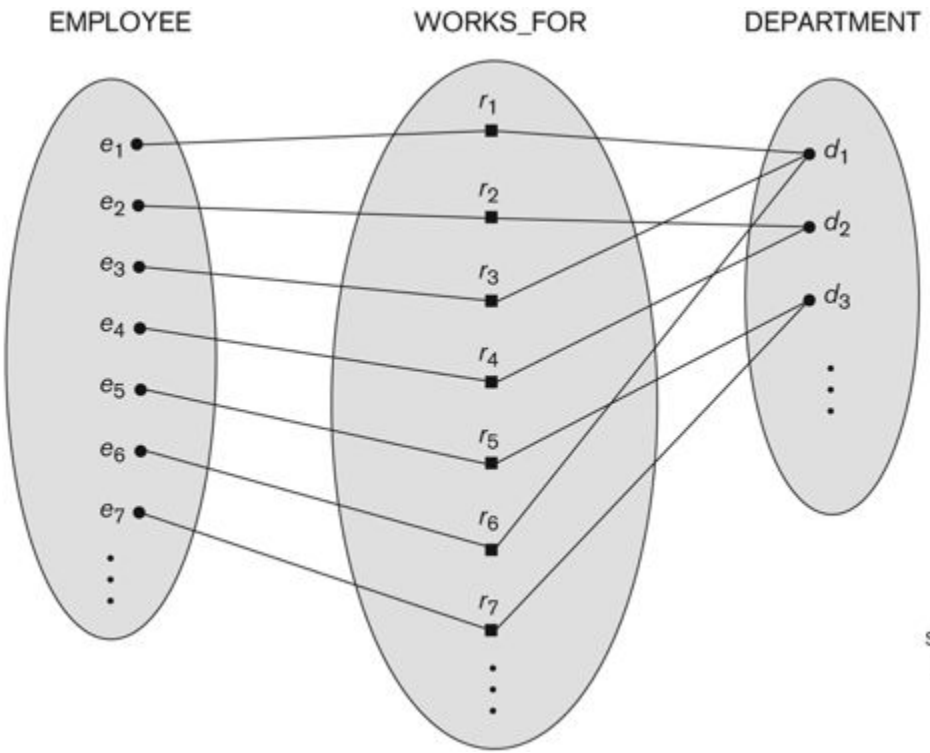


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

관계 종류 (M:N 관계)

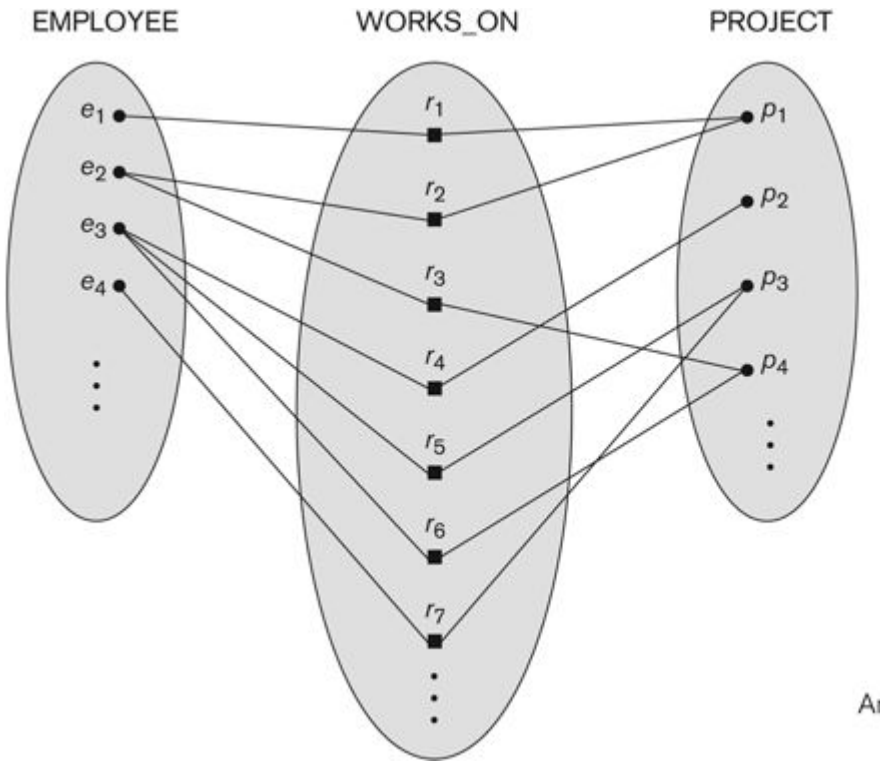


Figure 3.13
An M:N relationship, WORKS_ON.

ER 다이아그램: 전체, 부분 참여 제약조건

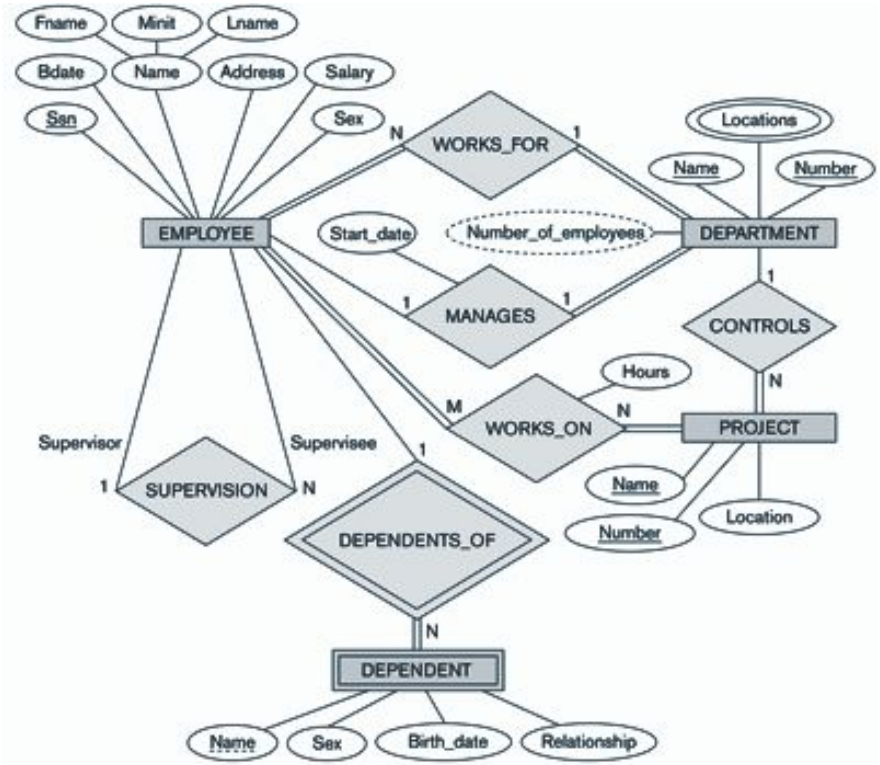
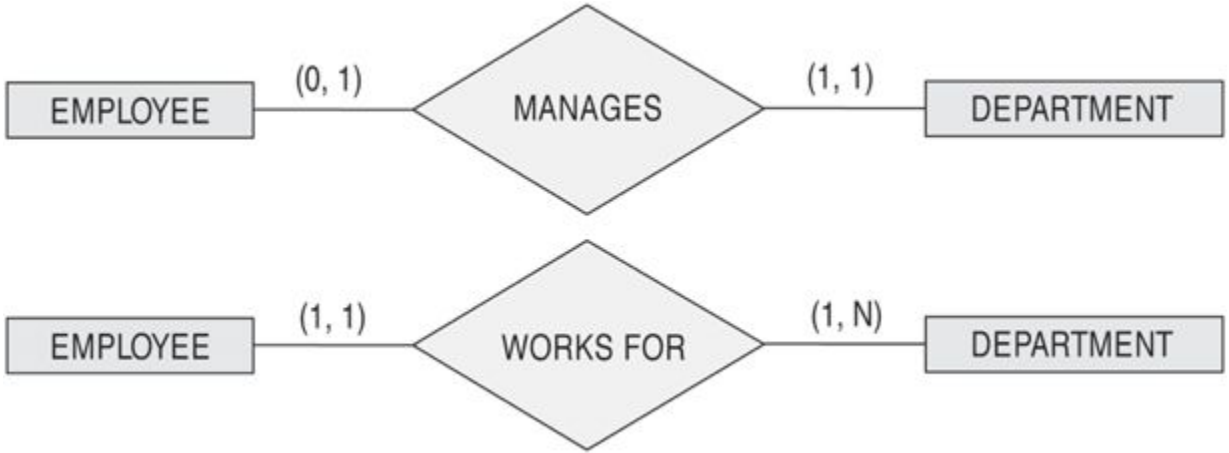


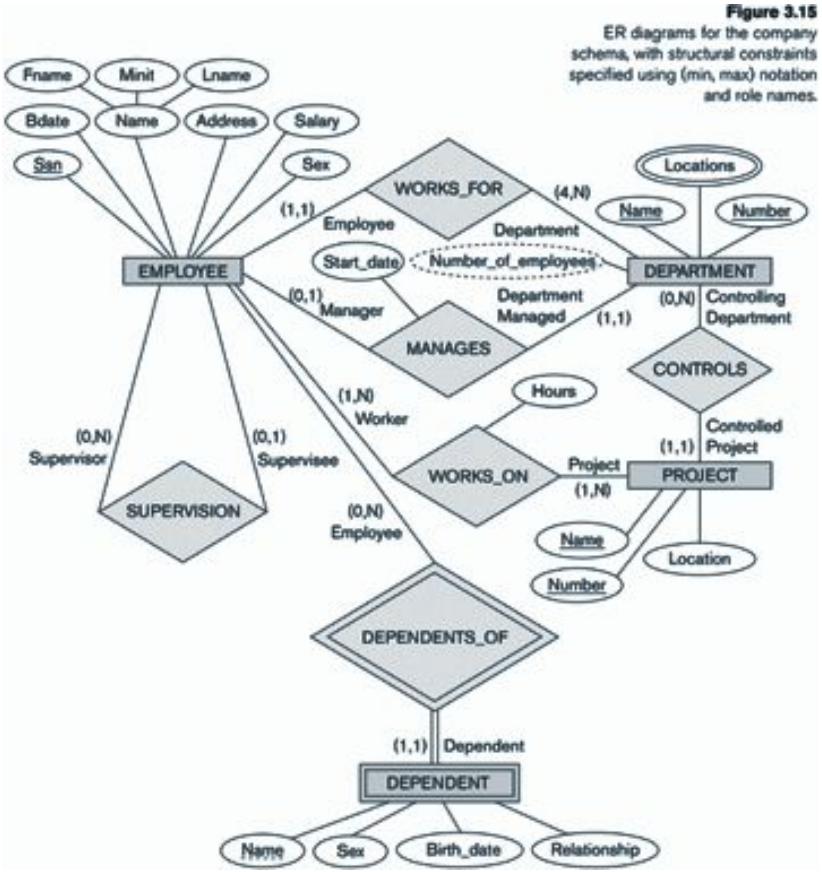
Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

ER 다이어그램: (min, max) 표기

➢ 엔터티 타입이 속해야할 최소, 최대 관계 수



ER 다이어그램: (min, max) 표기



약한 엔터티 타입

➢ 약한 엔터티 타입 (Weak Entity Type)

- 자신의 **키 애트리뷰트**가 **없는** 엔터티 타입
  - 키 애트리뷰트를 가지는 것은 **정규 엔터티 타입 (regular entity type)** 또는 **강한 엔터티 타입 (string entity type)**
- **식별 엔터티 타입** (identifying entity type) 혹은 **소유 엔터티 타입** (owner entity type): 약한 엔터티 타입의 엔터티들은 그들의 애트리뷰트 값들 중 하나를 통해 **다른 엔터티 타입**의 엔터티들과 연계됨
  - 약한 엔터테 타입들을 **식별할 수 있도록** 해주는 엔터티 타입임
  - 약한 엔터티 타입과 소유 엔터티 타입은 힝상 **식별 관계 (indentifying relationship)** 을 가짐: 약한 엔터티 타입은 식별 관계에 대해 **전체 참여 제약조건** (존재종속성)을 가짐
- 약한 엔터티 타입의 예
  - 부양 가족을 나타내는 DEPENDENT 엔터티는 관계된 사원 엔터티를 식별한 후에 식별될 수 있음
  - 서로 다른 부양 가족이 다른 애트리뷰트들이 다 같을 지라도 관계된 사원 엔터티를 식별한 후에 서로 다른 것으로 식별될 수 있음
    - DEPENDENT (약한 엔터티 타입) → EMPLOYEE (DEPENDENT 에 대한 소유 엔터티 타입)
- 부분키 (partial key): **동일한 소유 엔터티**에 연관되는 약한 엔터티들을 구분할 수 있게하는 애트리뷰트들
  - 약한 엔터티 타입이 **식별 관계 타입 이외에** 다른 관계 타입에 **독립적**으로 참여하지 않는 경우 **소유**

관계 데이터 모델
Relational Data Model

관계형 모델의 개념

➢ Informal Definition

- 릴레이션 (relation): 값들의 테이블, 레코드 들의 파일
- **튜플 (tuple)**: 각 **행(row)**은 관련된 데이터 값들의 모임, **실세계의 엔터티** 또는 **관계**와 대응
- **애트리뷰트 (attribute)**: 각 **열(column)**을 나타내며 각 행의 데이터 값들을 어떻게 해석할 것인가를 명시, **동일한 데이터 타입**을 가짐
- 릴레이션의 **키 (key)**: 테이블의 각 행을 **유일하게 식별**하는 애트리뷰트 혹은 애트리뷰트의 집합

➢ 도메인 (Domain: D)

- 도메인은 원자 값(더 이상 나누어질 수 없는 값)들의 집합.
- 일반적으로, 그 도메인에 속하는 값들의 데이터 타입이나 형식을 명시
  - 미국의 전화 번호는 10자리 번호로 이루어짐: (ddd)ddd-dddd
  - 한 회사의 15세와 80세 사이의 (정수 값의) 근로자들의 나이
- 도메인의 이름은 릴레이션에서의 도메인의 값들을 해석하는데 도움이 됨

관계형 모델의 개념

➢ 릴레이션 스키마 (schema)

- 릴레이션 이름(R)과 애트리뷰트들(A1,A2, … , An)로 이루어짐 → R(A1, A2, … , An)
- 애트리뷰트(Ai)의 도메인: dom(Ai)
- 릴레이션의 차수 (degree or arity of relation): 릴레이션 스키마의 애트리뷰트 수
- 예) STUDENT(Name, Ssn, Home_phone, Address, Office_phone, Age, Gpa)
  - 차수?, 타입?, 도메인?

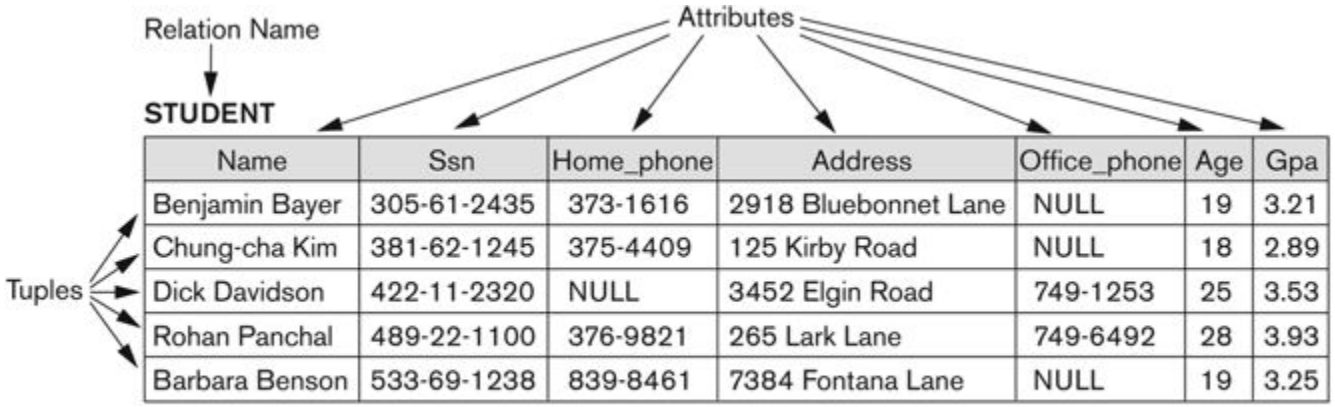
➢ 릴레이션 상태 (relation state)

- **n-튜플** (n-tuple): n개 값의 **순서 리스트** t = <v1,...,vn>, vi 는 dom(Ai)의 원소
  - 각 값은 적정한 도메인(domain)으로부터의 값이어야 함
- 릴레이션 또는 릴레이션 상태(relation state) **r**은 **n-튜플**들의 집합임
- n 개의 도메인 dom(A1),...,dom(An) 상의 **수학적 릴레이션**: R 을 정의하는 도메인들의 **카티션 곱** (Cartisan product)의 부분집합
  - $r(R) \subseteq (dom(A_1) \times \cdots \times dom(A_n))$
- 릴레이션 **스키마**는 상대적으로 **고정적**이지만, 릴레이션의 **상태**는 **변함**

관계형 모델의 개념

Informal Terms	vs	Formal Terms
Table		Relation
Column Header		Attribute
All possible Column Values		Domain
Row		Tuple
Table Definition		Schema of a Relation
Populated Table		State of the Relation

릴레이션 예: STUDENT



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

릴레이션의 특성

➢ 릴레이션과 튜플의 순서

- 릴레이션에서 튜플들은 특정 순서를 갖지 않음 (릴레이션은 튜플들의 집합임)
  - 튜플들의 순서는 릴레이션의 정의의 일부가 아님
- 한 튜플내에서는 애트리뷰트 값들의 순서가 정해짐 (튜플은 n개의 값의 순서 리스트임)
- 하지만, 더 일반적인 정의에서는 릴레이션 스키마를 애트리뷰트들의 집합으로 정의할 수도 있음
  - 이 경우에는 하나의 튜플이 (<애트리뷰트> <값>) 쌍들의 집합임

Figure 5.2
The relation STUDENT from Figure 5.1 with a different order of tuples.

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21

릴레이션의 특성

➢ 튜플에서의 값

- 평면 관계 (flat relational model): 제 1 정규형 (first normal form)
  - 튜플 내의 각 값은 원자값(atomic value)으로 간주
  - 기본적인 관계 모델 구조내에서는 composite attribute 와 multi-valued attribute 는 허용되지 않음
- 널 (NULL): 튜플 내의 어떤 애트리뷰트의 값을 알 수 없거나 값이 지정되지 않을 때
  - 값을 알 수 없는 경우, 가능하지 않은 값을 갖고 있는 경우, 이 튜플에는 이 애트리뷰트를 적용할 수 없는 경우

➢ 릴레이션의 해석 (의미)

- 릴레이션 스키마는 선언 또는 일종의 주장(assertation)으로 해석할 수 있음
  - 엔터티에 대한 사실을 표현
  - 엔터티 간의 관계에 대한 사실을 표현
- 프레디키트(predicate): 각 튜플의 값들이 프레디키트를 만족하는 값들로 해석

관계 모델 제약 조건

➢ 제약 조건 (constraints)

- 일반적으로 데이터베이스 상태에서는 실제 값에 대한 많은 제약조건이 있음
- 크게 세 가지로 구분됨
  - **본질적 모델 기반 제약조건** 또는 **함축적 제약 조건: 데이터 모델 자체**에 존재하는 제약조건
    - 릴레이션은 중복 튜플을 가질 수 없음
  - **스키마 제약조건** 또는 **명시적 제약 조건**: 데이터 모델의 **스키마**에서 **직접 표현 가능**한 제약 조건으로 주로 **DDL**(data description language)로 명시
  - **응용 기반 제약 조건** 또는 **의미적 제약 조건 (비즈니스 규칙)**: 데이터 모델의 스키마에서 **직접 표현이 불가능**한 제약 조건으로 **응용 프로그램**에 의해 표현되고 시행 (SQL assertation으로 명시)

➢ 스키마 제약 조건

- 도메인 제약 조건
- 키 제약조건 및 널에 대한 제약 조건
- 엔터티 제약조건 및 참조 무결성 제약 조건

키 제약조건 예

➢ 키 제약조건 (key constraint)

- 유일성 제약조건 (uniqueness constraint): 릴레이션은 **튜플들의 집합**이므로 모든 원소(튜플)가 **중복되지 않아야 함**
- **슈퍼키 (superkey)**: 릴레이션 스키마(R) 의 릴레이션 상태 r의 어떤 두 튜플도 서로 다른 값들을 갖는 애트리뷰트들의 부분 집합
  - 모든 릴레이션은 적어도 하나의 슈퍼키를 갖음
- **키(key, K)**: 어떤 릴레이션 스키마 (R)의 슈퍼키이고, K에서 어떤 애트리뷰트라도 빠지면 더이상 슈퍼키가 될 수 없는 성질을 지님
  - 서로 다른 두 튜플은 동일한 키 애트리뷰트 값을 가질 수 없음
  - 키는 **최소의 슈퍼키**임 **(minimal superkey)**
- 키를 포함하는 애트리뷰트들의 집합은 항상 슈퍼키임
- **후보키(candidate key)**: 릴레이션 스키마는 **하나 이상의 키**를 가질 수 있음
- **기본키(primary key)**: **후보키 중에 하나**를 지정 (밑줄) ↔ **유일키 (unique key)**: **다른 후보키**들
  - 하나 또는 적은 수의 애트리뷰트를 가진 기본키를 선택하는 것이 선호됨

스키마 제약 조건

- **도메인 제약 조건 (domain constraint)**
  - 각 튜플 내에서 각 **애트리뷰트 A**의 값이 반드시 **A**의 도메인 **dom(A)**에 속하는 원자값이어야 한다는 조건
- **널에 대한 제약조건 (NULL constraint)**
  - 널 값의 **허용 여부**에 대한 제약 조건
  - NOT NULL: **모든 튜플**이 **특정 에트리뷰트**에 대해 **널이 아닌 유효한 값**을 가져야 함

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

Figure 5.4
The CAR relation, with two candidate keys: License_number and Engine_serial_number.

관계 데이터베이스 스키마

➢ 관계 데이터베이스 스키마 (relational database schema; S)

- **같은 데이터베이스**에 속해 있는 **릴레이션 스키마**들의 **집합**과 **무결성 제약조건(integrity constraint)**들의 **집합 (IC)**
  - **S** 는 **전체 데이터베이스 스키마**의 이름
  - **S** = {R1,R2, … , Rm}; **Ri** 는 데이터베이스 S에 속해있는 **개별 릴레이션 스키마**들의 이름
- **관계 데이터베이스 상태** (relational database state; **DB**): **릴레이션 상태**들의 **집합**
  - DB = {r_1, … , r_m}; r_i 는 IC 에서 명시된 무결성 제약 조건들을 만족
  - **유효하지 않은 상태** (invalid state): **무결성 제약 조건**을 **준수하지 않는 데이터베이스 상태**
  - **유효한 상태** (valid state): **무결성 제약 조건**을 **준수**하는 **데이터베이스 상태**
- **관계 데이터베이스**라고 할 때에는 묵시적으로 그것의 **스키마**와 **현재 상태**를 모두 포함
- 여러 릴레이션에서 동일한 실세계 개념을 나타내는 애트리뷰트들이 같은 이름을 가질 수도 있음
- 반대로, 두 개의 애트리뷰트가 같은 이름을 공유하지만 서로 다른 실세계 개념을 나타낼 수도 있음
- 관계 DBMS는 관계 데이터 베이스 스키마를 정의하기 위해 SQL DDL을 사용함

관계 데이터베이스 스키마의 예

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 5.5
Schema diagram for the COMPANY relational database schema.

엔터티 무결성과 참조 무결성

➢ 엔터티 무결성 제약조건 (entity integrity constraint)

- 어떠한 기본키 값도 널 값이 될 수 없음
  - 기본키 값은 한 릴레이션 내의 각 튜플을 식별하는데 사용되기 때문
- **각 릴레이션**에 명시되는 조건임

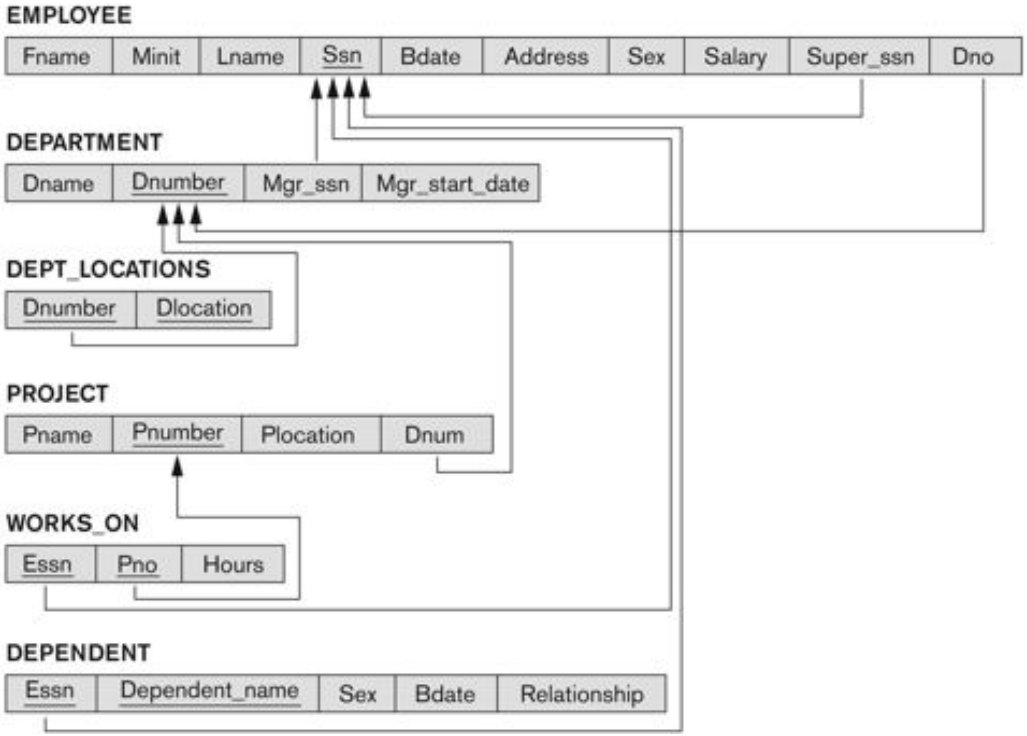
➢ 참조 무결성 제약 조건 (referential integrity constraint)

- **두 릴레이션 사이**에 명시되는 조건임
  - 한 릴레이션에 있는 튜플이 다른 릴레이션에 있는 튜플을 **참조하려면** 반드시 **참조되는 튜플**이 그 **릴레이션 내**에 **존재**해야 함
  - 예) EMPLOYEE의 Dno 애트리뷰트는 각 사원이 근무하는 부서 번호를 나타내므로 DEPARTMENT 릴레이션의 어떤 튜플의 Dnumber 값과 반드시 일치해야 함
- 외래 키 (foreign key): 두 릴레이션 스키마 R1(참조한 릴레이션; referencing relation)과 R2(참조된 릴레이션; referenced relation) 사이에 참조 무결성 제약조건이 만족 되어야 함
  - 릴레이션 스키마 R1의 어떤 애트리뷰트들의 집합 FK 가 다음의 규칙을 만족하면 FK는 릴레이션 R2를 참조하는 R1의 외래키임

1. **FK의 애트리뷰트**는 **R2의 기본키 PK** 의 애트리뷰트와 **동일한 도메인**을 가진다. 이때 FK 는 릴레이션 R2를 참조한다고 말한다.
2. 현재 상태 r1(R1) 의 한 튜플 t1 내의 FK 값은 현재 상태 r2(R2)의 어떤 튜플 t2내의 **PK 값과 일치**하거나 **널 값**을 가져야 한다. 전자의 경우에는 t1[FK] = t2[PK]이고 튜플 t1이 튜플 t2를 참조한다고 말한다

참조 무결성 제약조건의 예

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



다른 형태의 제약 조건

- **의미적 무결성 제약조건 (semantic integrity constraint)**
  - DDL의 일부가 아니고 데이터베이스 응용에서 발생하기 때문에 다른 방법으로 명시되고 지켜져야 함
    - 사원의 급여는 상사의 급여를 초과해서는 안됨
    - 한 사원이 주당 모든 프로젝트에서 일할 수 있는 시간의 합은 최대 56시간임
  - 응용 프로그램 내에서 또는 제약 조건 명세 언어 (constraint specification language)를 사용하여 정의하고 따라야 함
    - 응용 프로그램 내에서 트리거(trigger)와 주장(assertion) 이라는 기법 사용 하는 것이 더 일반적임
- **전이 제약 조건 (transition constraint)**
  - 상태 제약 조건(state constraint): 지금까지의 제약 조건은 데이터베이스의 유효한 상태가 만족야 하는 조건임
  - 전이 제약 조건(transition constraint): 데이터베이스 내의 상태 변화를 다루기 위해서 정의되는 제약조건
  -

갱신 연산

➢ 관계 모델의 연산

- 추출(retrieval)과 갱신(update)으로 나눌 수 있음
  - 추출에 사용되는 관계 대수 표현은 기존의 릴레이션 집합에 여러 대수 연산자를 적용 후 새로운 릴레이션을 생성함
- 릴레이션에 대한 기본 갱신 연산
  - 삽입(insert) | 삭제(delete) | 갱신 또는 수정 (update or modify)
- 각 연산이 위반을 유발할 수 있는 제약조건들이 존재

➢ 삽입 (insert)

- 릴레이션 R에 새로운 튜플 t 를 삽입
- 도메인 제약조건, 키 제약조건, 엔터티 제약조건, 참조 제약 조건을 위반 할 수 있음
  - 예: 외래키의 값이 참조되는 릴레이션에 존재하지 않는 튜플을 가리킴
- 해결 방법
  - 삽입을 거부
  - 삽입이 거부된 이유를 정정

갱신 연산

➢ 삭제 연산 (delete)

- 튜플을 삭제
- 참조 제약 조건 위반 가능
  - 삭제되는 튜플이 데이터베이슷 다른 튜플에 의해 외래키로 참조되고 있는 경우
- 해결 방법
  - 삭제를 거부
  - 삭제되는 튜플을 참조하는 모든 튜플을 삭제하는 연쇄 삭제 (cascade)
  - 제약 조건 위반의 원인이 되는 참조하는 애트리뷰트 값을 변화시킴 (널 값으로 바꾸거나 디폴트 튜플을 참조 하도록 바꿈)
  - 경우에 따라서, 위에 세 가지의 적절한 조합 가능

➢ 갱신 연산 (update)

- 릴레이션 R에 있는 튜플들 중에서 하나 이상의 애트리뷰트 값을 변경
- 기본키나 외래 키가 아닌 애트리뷰트의 갱신에는 아무런 문제가 없음

트랜잭션 개념

➢ 트랜잭션 (transaction)

- 트랜잭션 (transaction): 데이터베이스로부터 읽기, 삽입, 삭제, 또는 갱신과 같은 데이터베이스 연산들을 수행하는 프로그램
- 데이터베이스 응용 프로그램은 관계 데이터베이스에 대해 하나 이상 트랜잭션들을 수행함
  - 트랜잭션은 데이터베이스에 대해 명세화된 모든 제약 조건들을 만족하는 일관된(또는 타당한) 상태로 수행해야 함
- 하나의 트랜잭션은 검색 연산과 갱신연산 여러개를 포함할 수 있음
  - 검색과 갱신은 데이터베이스 내에서 원자 단위로 수행
  - 예: 은행의 예금 인출 트랜잭션
- 온라인 트랜잭션 프로세싱 시스템(OLTP): 초당 수백 개의 트랜잭션들을 수행
  - 트랜잭션의 동시 수행, 실패로부터의 회복에 대한 문제가 중요

연습 문제

[Taken from Exercise 4.16]

학생들의 강좌 수강과 각 강좌에 사용된 교재를 기록하는 데이터베이스를 위한 릴레이션들이 다음과 같다:

STUDENT(SSN, Name, Major, Bdate)

COURSE(Course#, Cname, Dept)

ENROLL(SSN, Course#, Quarter, Grade)

BOOK_ADOPTION(Course#, Quarter, Book_ISBN)

TEXT(Book_ISBN, Book_Title, Publisher, Author)

필요하면, 스스로 가정을 만들고 위의 스키마의 외래키를 명시하라.