Chap 3. Data Modeling Using the Entity-Relationship Model

데이터베이스 설계

- Entity-Relationship Model
  - High-level data model
  - 광범위하게 사용되는 개념적 데이터 모델
- 개념적 데이터 모델(high-level data model)
  - DB 사용자가 이해하는 데이터베이스의 구조
  - 개체(entity):실세계의 물건, 개념[과제, 고용인]
  - 속성(attribute):개체의 성질, 속성
    [고용인의 이름, 월급]
  - 관계(relationship):둘이상의 개체간의 관계
    [고용인 <- works-on 관계 -> 과제]

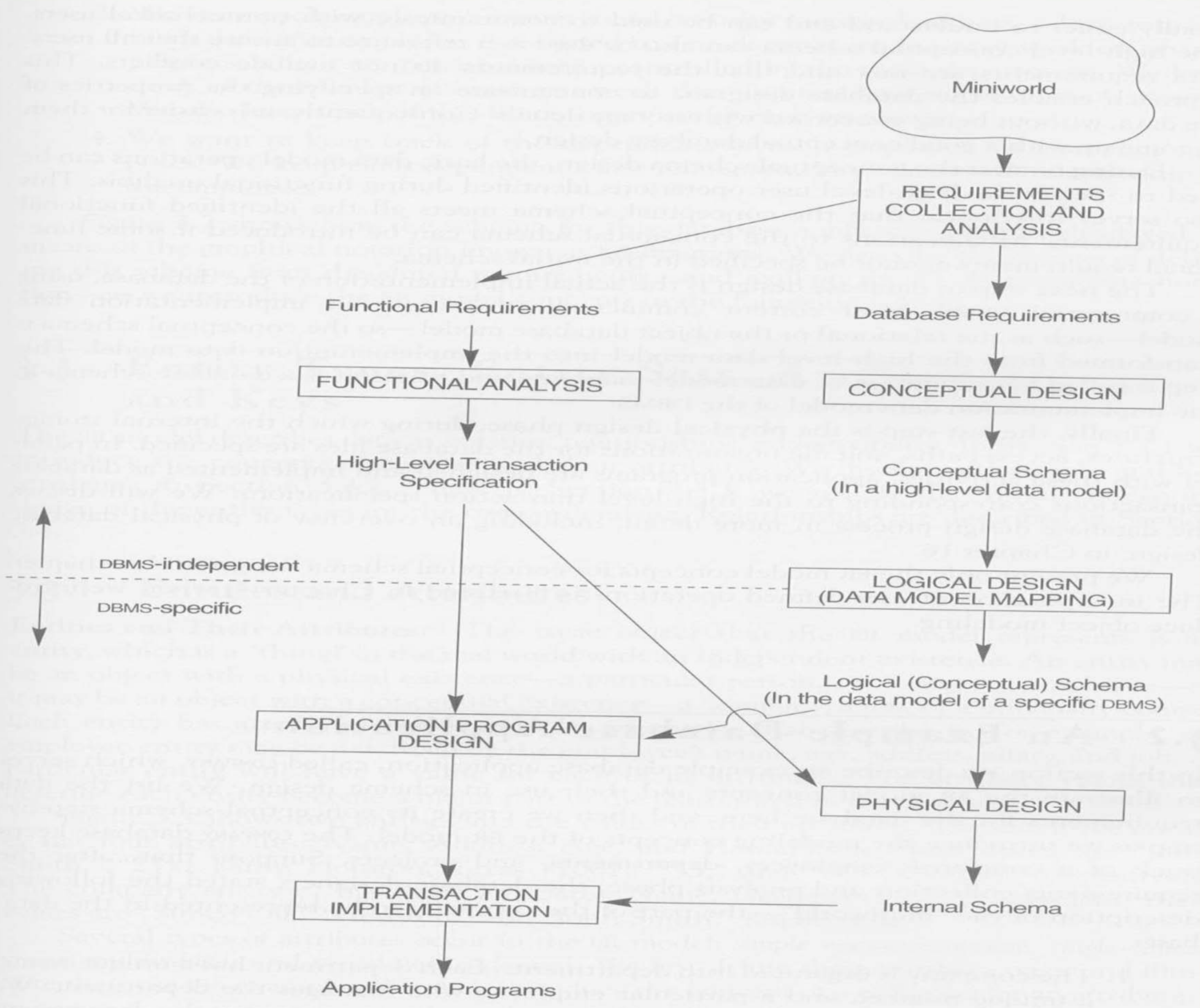

Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Database Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

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

데이터베이스 설계 단계

- 1. 요구 수집 및 분석
  - DB 사용자의 요구를 수집 및 분석(DB 설계자)
  - DB requirements : 데이터 저장 요구사항
  - functional requirement : 데이터 활용 요구사항 (데이터의 삭제, 삽입, 갱신 방법)

- 2. 개념적 스키마 설계
  - 데이터 형, 연관관계, 제약조건
    - high-level data model 사용
    - no implementation details (no storage spec.)

데이터베이스 설계 단계

- 3. Functional analysis
  - functional requirement를 사용하여 high-level transaction 설계
  - 개념적 스키마와 확인 / 조절

- 4. 데이터베이스 구현(논리적 데이터베이스 설계)
  - 상용화된 DBMS를 사용하여 데이타베이스 구현
  - 상용화된 DBMS는 구현 데이터 모델을 사용하기 때문에 개념적 스키마에서 구현 스키마로의 mapping이 필요
    - high-level data model <=> implementation data model

데이터베이스 설계 단계

- 5. 물리적 데이터베이스의 설계
  - 내부저장 구조 및 파일 구조 설정
- 6. 응용 프로그램 설계 및 구현
  - high-level transaction사용

Example : 회사 데이터베이스

- Mini-world description
  - 회사는 부서로 구성되고, 부서마다 각각 유일한 이름과 번호를 갖고 부서장이 존재한다.
  - 모든 부서장의 부서장직의 시작날짜를 기록하고, 한 부서는 여러군데 있을 수 있다.
  - 각 부서는 여러 개의 과제를 수행하고, 각 과제는 유일한 이름과 번호를 가지며 한 위치에 있다.
  - 모든 직원의 이름, 주민등록번호, 주소, 월급, 성별, 생일을 저장하고, 각 직원은 한 부서에 배정되어 다른 부서에서 관리하는 여러 개의 과제를 수행할 수 있다.
  - 각 직원이 과제 당 일하는 시간을 기록하고, 각 직원의 관리자를 기록한다.
  - 모든 직원의 가족 사항(가족 이름, 생일 관계)을 보험을 위해 기록한다.

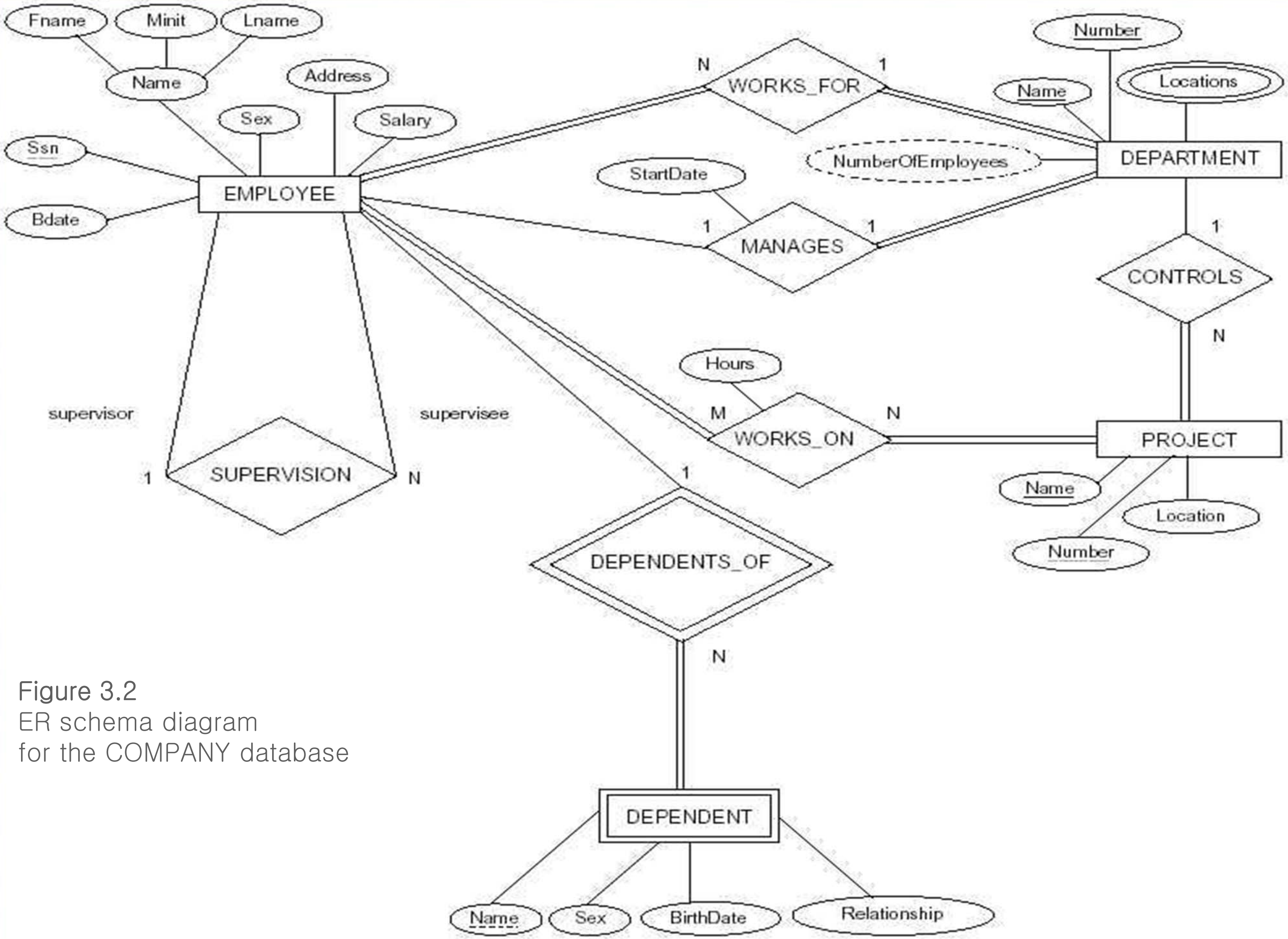
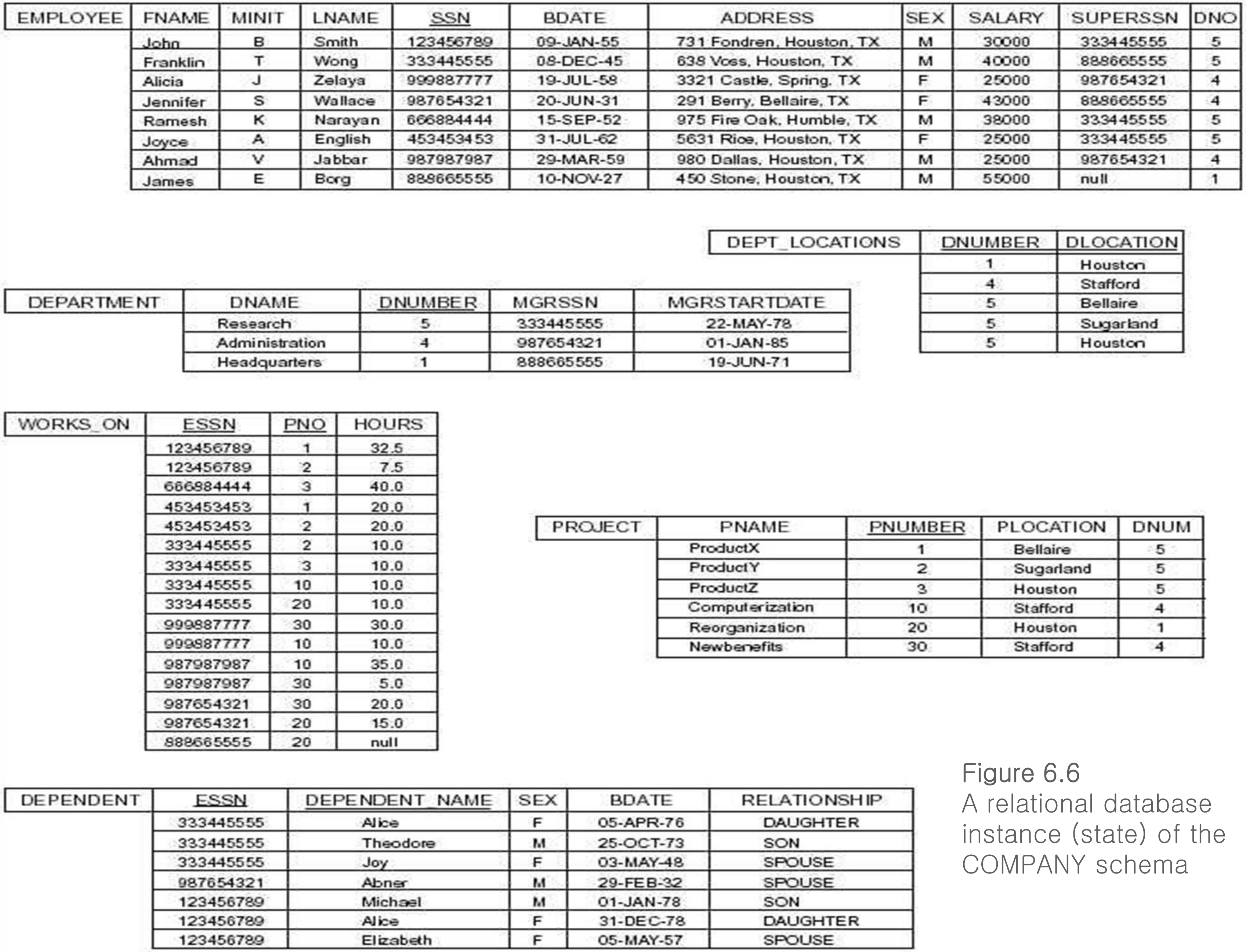


Figure 3.2
ER schema diagram
for the COMPANY database



EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	09-JAN-55	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	08-DEC-45	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	19-JUL-58	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	20-JUN-31	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	15-SEP-52	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	31-JUL-62	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	29-MAR-59	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	10-NOV-27	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

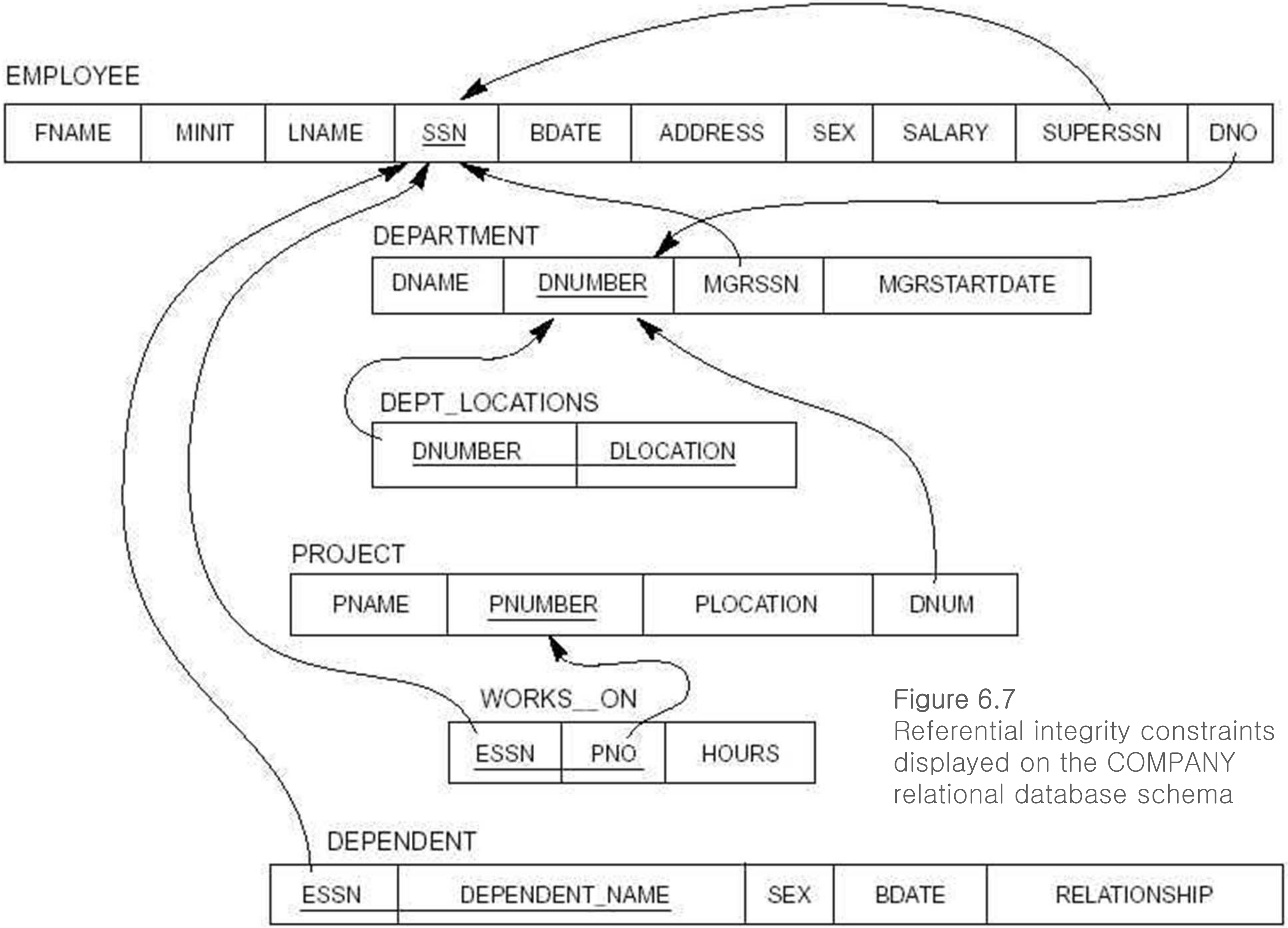
DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	22-MAY-78
	Administration	4	987654321	01-JAN-85
	Headquarters	1	888665555	19-JUN-71

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	05-APR-76	DAUGHTER
	333445555	Theodore	M	25-OCT-73	SON
	333445555	Joy	F	03-MAY-48	SPOUSE
	987654321	Abner	M	29-FEB-32	SPOUSE
	123456789	Michael	M	01-JAN-78	SON
	123456789	Alice	F	31-DEC-78	DAUGHTER
	123456789	Elizabeth	F	05-MAY-57	SPOUSE

Figure 6.6
A relational database instance (state) of the COMPANY schema



EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS__ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

Figure 6.7
Referential integrity constraints displayed on the COMPANY relational database schema

ER 모델의 개념

- Entity-Relationship
  - 데이타를 실체, 관계, 속성으로 표현
- Entity and Attribute
  - Entity : 실세계의 물리적 또는 개념적으로 존재하는 것
    - 물리적 : 자동차, 직원, 학생
    - 개념적 : 회사, 직업, 과목
  - Attribute : Entity의 성질
    - 직원 : 이름, 월급, 나이...
  - 각 Entity는 Attribute에 대한 값을 갖는다
    - 이름 : 홍길동

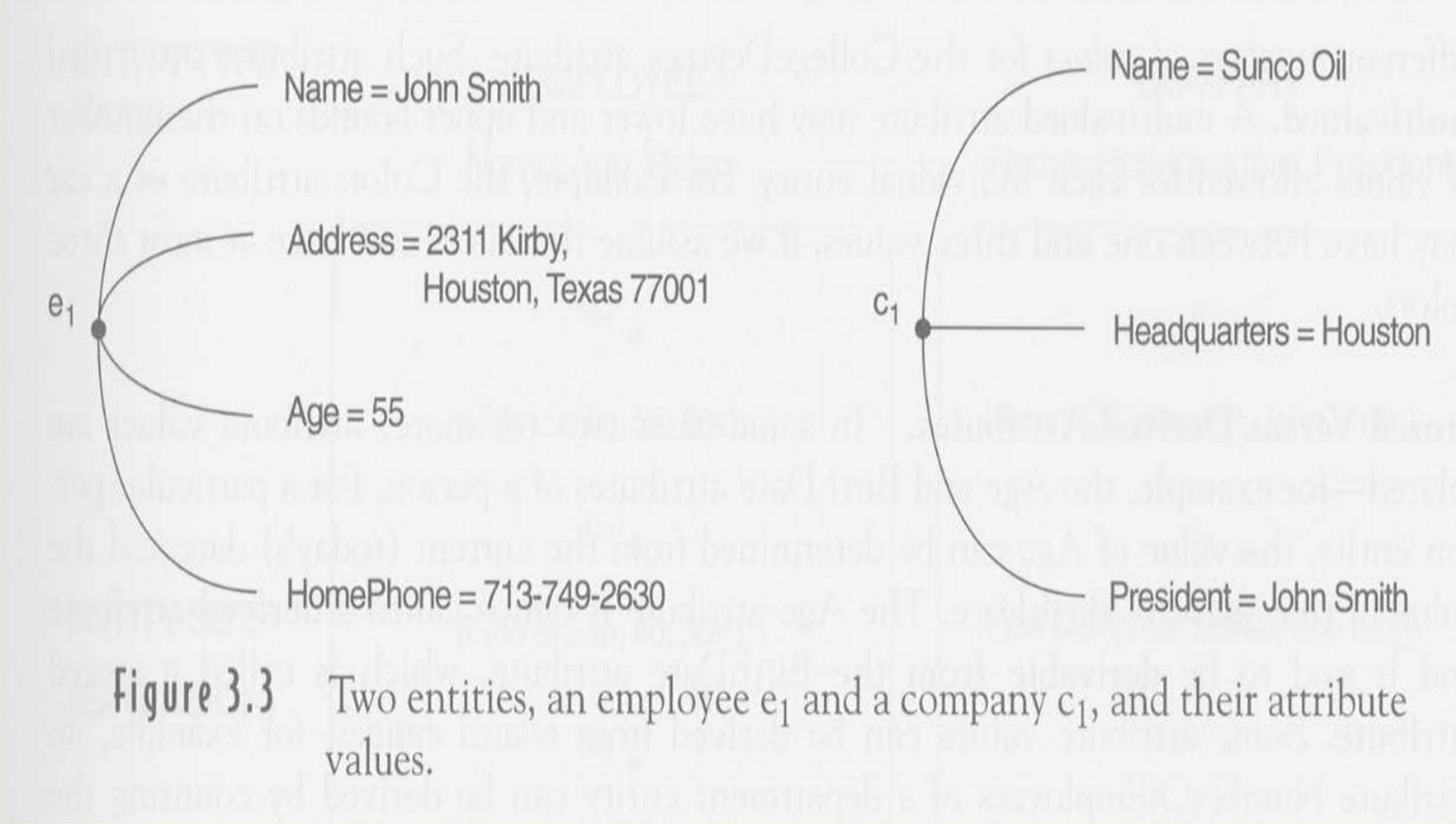


Figure 3.3 Two entities, an employee e_1 and a company c_1, and their attribute values.

Attribute의 형태

- simple(atomic) attributes
  - 나눌 수 없는 속성
    - 사람의 나이 : 오직 한 개의 나이 값
- composite attributes
  - 기본적인 여러 개의 속성으로 세분화 될 수 있는 속성

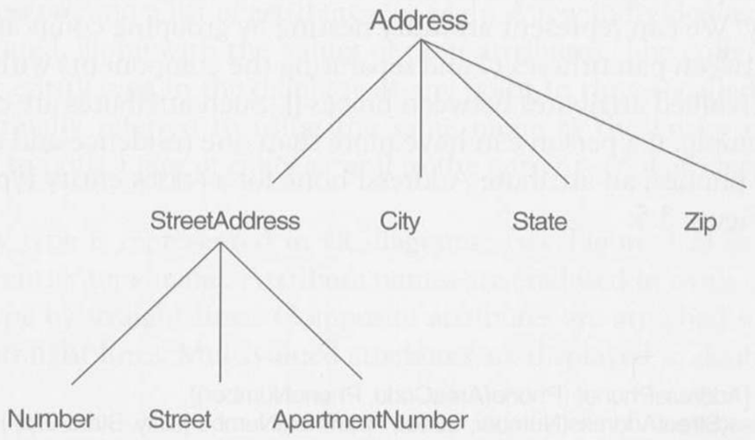


Figure 3.4 A hierarchy of composite attributes; the StreetAddress component of an Address is further composed of Number, Street, and ApartmentNumber.

Attribute의 형태

- single-valued attributes
  - 특정 entity에 오직 한 개의 값만을 갖는 속성
    - 사람의 나이 : 오직 한 개의 나이 값

- Multi-valued attributes
  - 한 개 이상의 값을 갖을 수 있는 속성 : set-value.
    - *자동차의 색 : 여러 가지의 색 (경찰차)*

Attribute의 형태

- stored attributes
  - 속성값이 DB에 저장됨
- derived attributes
  - 다른 속성의 값으로 결정 됨
    - *나이 = f(생일, 오늘 날짜)*

Null Value

- not applicable
  - 주소의 아파트 번호:
    - 단독 주택의 아파트 번호 속성 = null
  - 사람의 학사학위 :
    - 학사학위가 없는 사람의 학위 속성 = null
- not Known
  - missing : 속성의 값이 있지만 모름
    - 홍길동의 키 = null
  - not known whether it exists : 속성의 값의 존재 여부
    - 홍길동의 핸드폰 번호 = null

Entity Types, Entity Sets, Keys, and Value sets

- Entity sets
  - 동일한 구조를 갖는 entity의 집합
  - ER diagram에서 직사각형으로 표현
- Entity Types
  - 동일한 속성을 갖는 entity의 구조를 정의함
  - entity의 이름, 속성 이름 리스트
- ER diagram
  - entity type : 직사각형
  - attribute : 타원
  - multi valued attribute : double line

ER Model Basics

- *Entity:* Real-world object distinguishable from other objects. An entity is described (in DB) using a set of *attributes*.
- *Entity Set*: A collection of similar entities. E.g., all employees.
  - All entities in an entity set have the same set of attributes. (Until we consider ISA hierarchies, anyway!)
  - Each entity set has a *key*.
  - Each attribute has a *domain*.

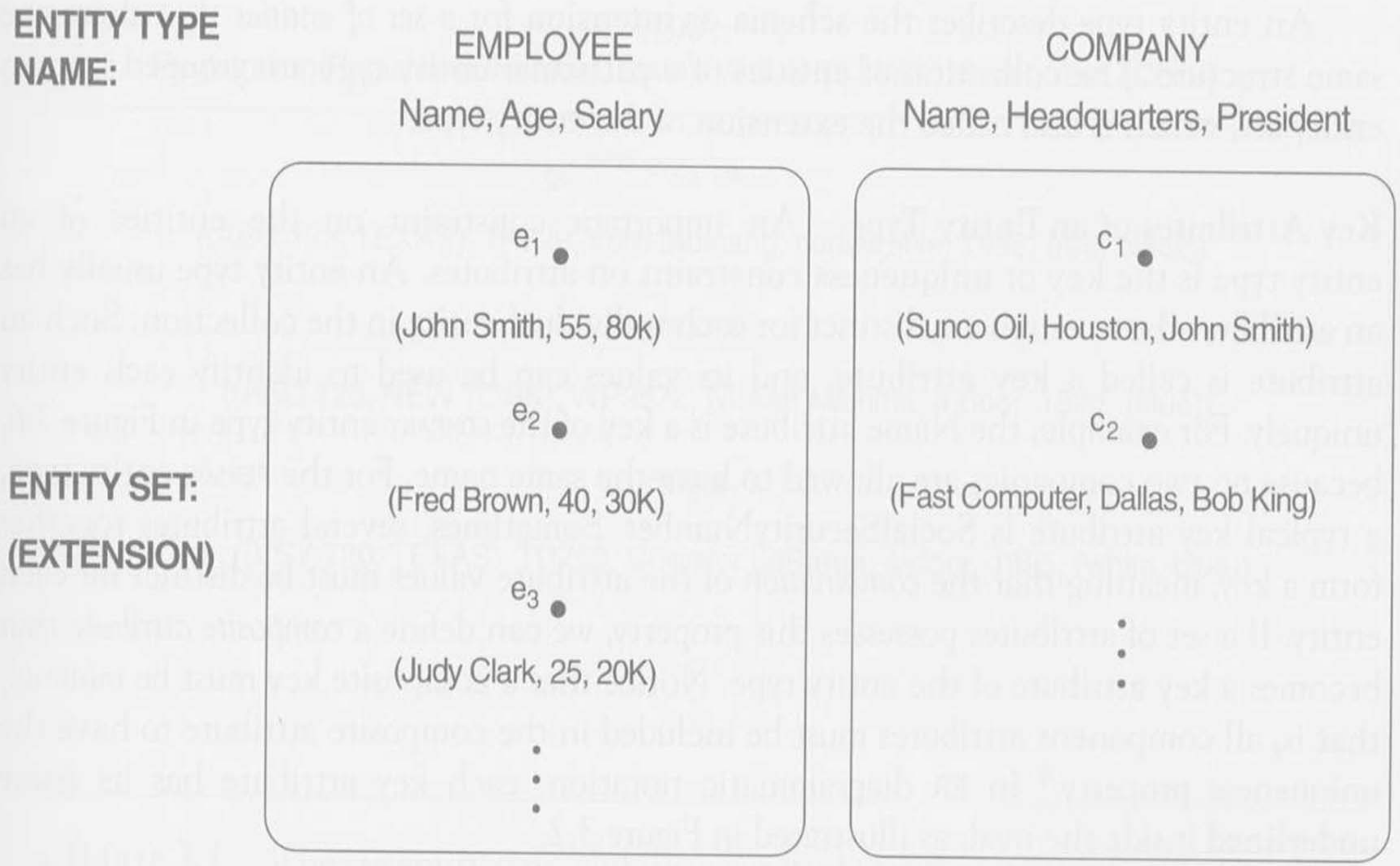
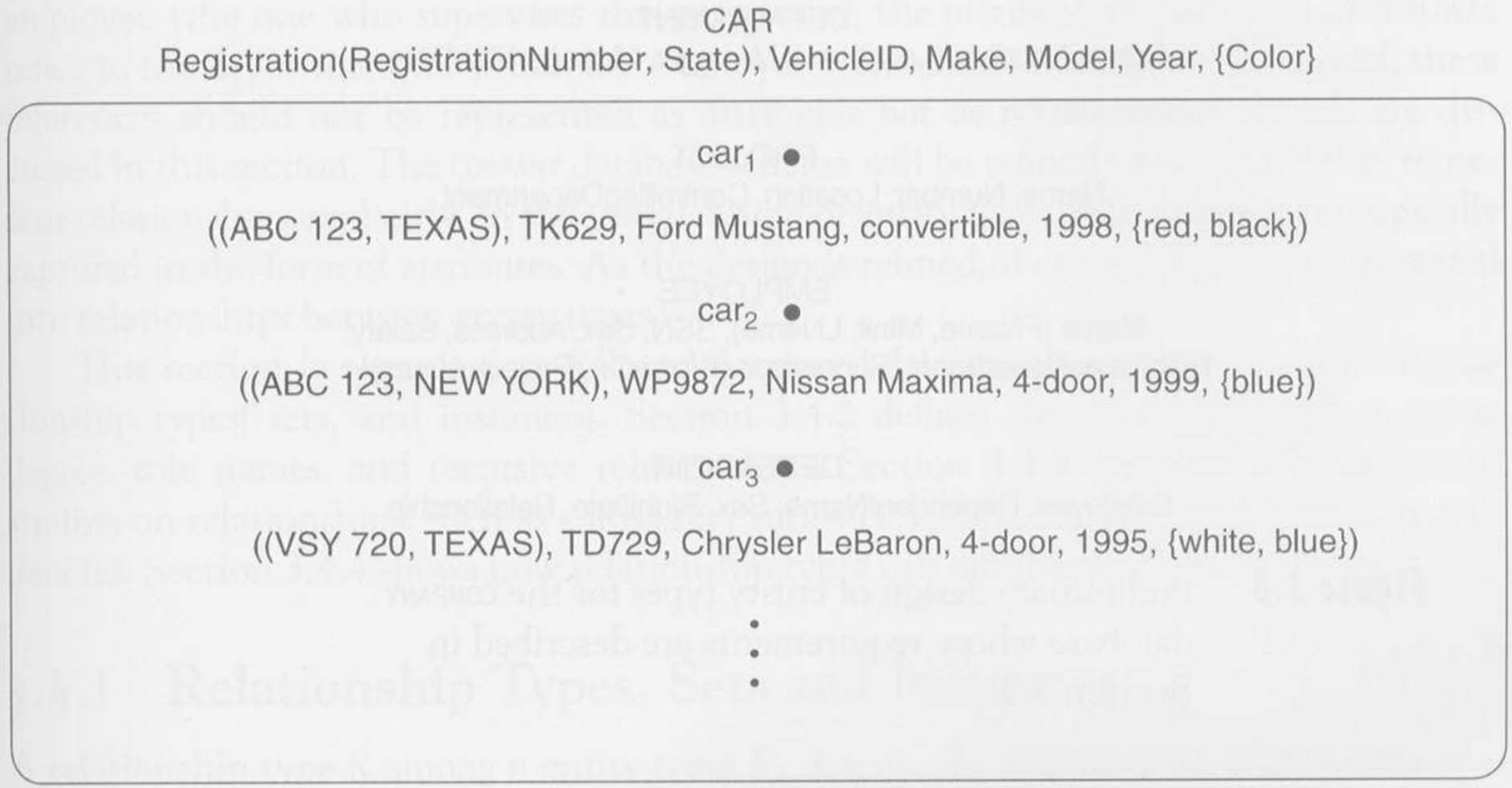


Figure 3.6 Two entity types named EMPLOYEE and COMPANY, and some of the member entities in the collection of entities (or entity set) of each type.

Entity Types, Entity Sets, Keys, and Value sets

- Key Attributes of an Entity Type
  - key attribute : 각 entity가 모두 다른 속성값을 갖는 속성
  - key attribute 값: entity = 1:1
    - 회사의 이름 : key attribute of company
    - 다수의 key attribute 존재 가능
  - entity set의 모든 entity에 적용됨
  - DB로 설계되는 mini-world의 성질
  - ER diagram : underline on name



CAR

Registration(RegistrationNumber, State), VehicleID, Make, Model, Year, {Color}

car$_1$ ●

((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 1998, {red, black})

car$_2$ ●

((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 1999, {blue})

car$_3$ ●

((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 1995, {white, blue})

•

•

•

Figure 3.7 The CAR entity type, with two key attributes Registration and VehicleID. Multivalued attributes are shown between set braces {}. Components of a composite attribute are shown between parentheses ().

Entity Types, Entity Sets, Keys, and Value sets

- Domain of an Attribute
  - 모든 단순 속성 <-> domain
    - 직원의 나이(16 – 70 세) => integer of 16 – 70 (domain)

E : entity, A : attribute, V : domain of an attribute

A : E → P(V) (P(V):power set of V, all subsets of V}

single-valued attribute : only one value

multi-valued attribute : set value

composite attribute : cartesian product

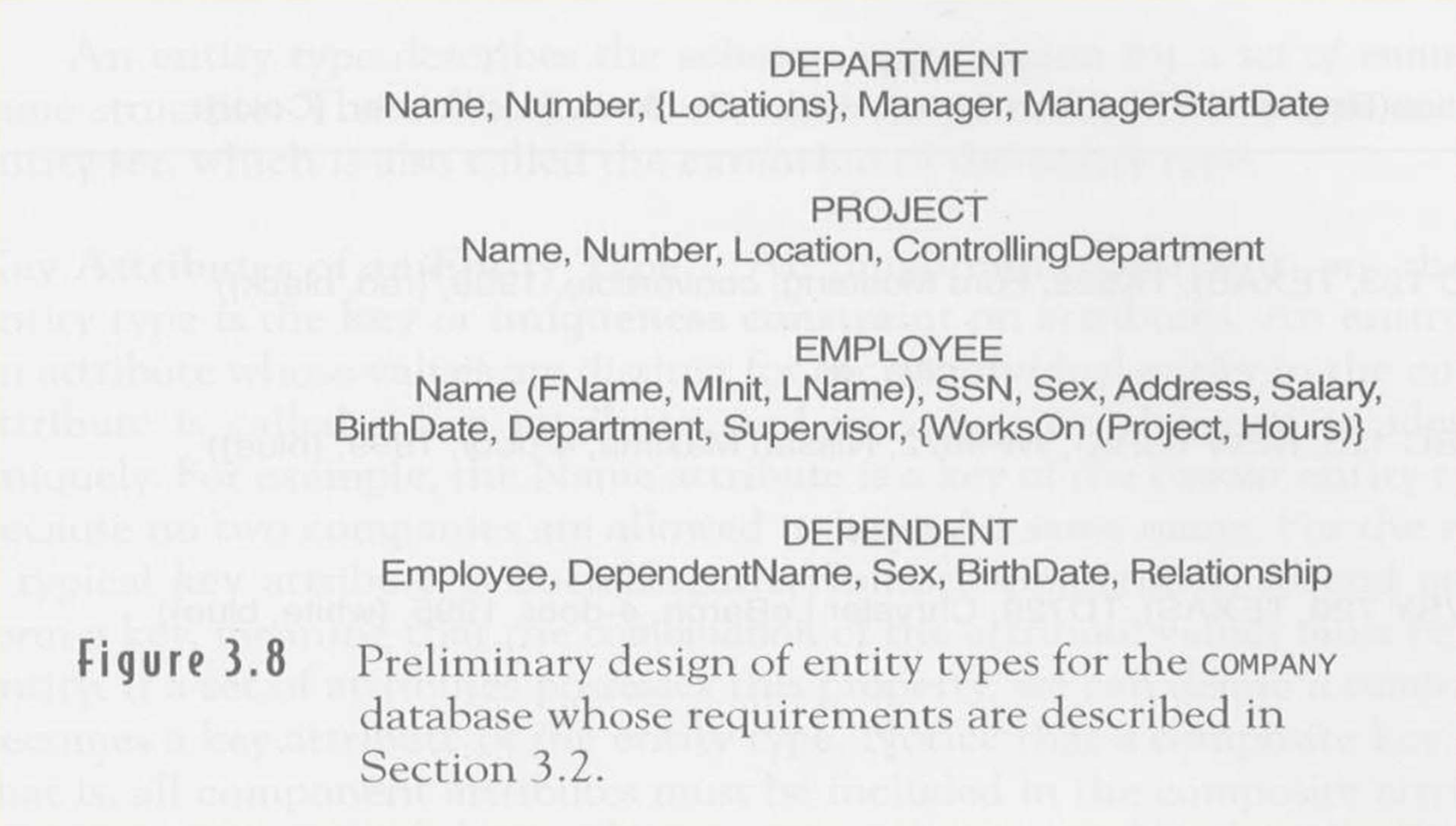
V = P(V1) x P(V2) x…x P(Vn)

Entity Types, Entity Sets, Keys, and Value sets

- Notation
  - ( ) : composite attribute
  - { } : multi-valued attribute
    - 한 개 이상의 집과 전화를 갖는 사람
    - {AddressPhone({ Phone(AreaCode, PhoneNumber)}, Address(StreetAddress(Number,Street, AptNum), City,State,Zip))}

Initial Conceptual Design of Company Database

- 4개의 entity 정의



DEPARTMENT
Name, Number, {Locations}, Manager, ManagerStartDate

PROJECT
Name, Number, Location, ControllingDepartment

EMPLOYEE
Name (FName, MInit, LName), SSN, Sex, Address, Salary,
BirthDate, Department, Supervisor, {WorksOn (Project, Hours)}

DEPENDENT
Employee, DependentName, Sex, BirthDate, Relationship

Figure 3.8 Preliminary design of entity types for the COMPANY database whose requirements are described in Section 3.2.

Initial Conceptual Design of Company Database

- 부서
  - 이름, 번호, 위치, 부서장, 부서장 시작 날짜
    - 위치 => multivalued attribute
    - key attribute : 부서이름 또는 부서 번호
- 과제
  - 이름, 번호, 위치, 관리부서
    - key attribute : 이름 또는 번호
- 직원
  - 이름, 주민등록번호, 성별, 주소, 월급, 생일, 부서, 관리자
    - 이름, 주소가 복합속성이 될 수 있다 (not specified)
    - key attribute : 주민등록 번호

Initial Conceptual Design of Company Database

- 가족
  - 이름, 성별, 생일, 관계
    - key attribute : 이름 (가족에는 동명이인 없음)
- 표현하지 않은 조항
  - 한 직원이 여러 과제를 수행할 수 있다.
  - 한 직원이 과제 당 수행한 시간

=> Works-on : 속성으로 표시

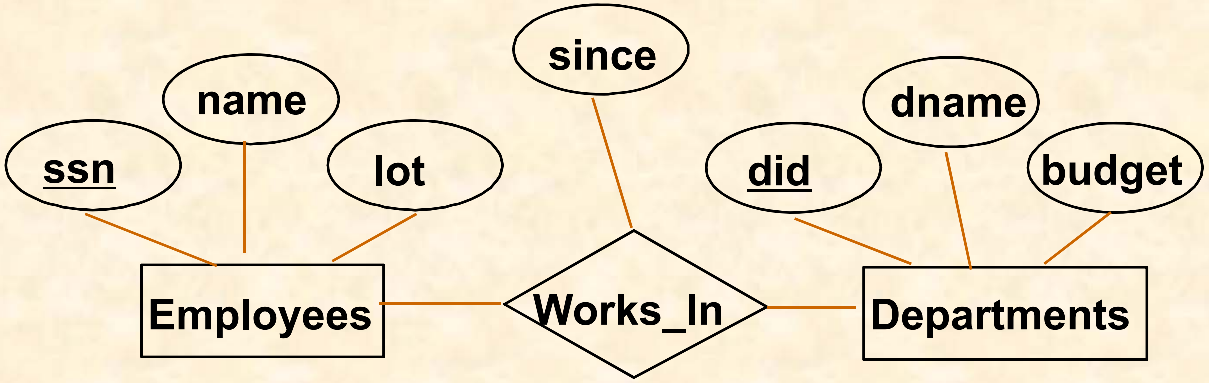
Initial Conceptual Design of Company Database

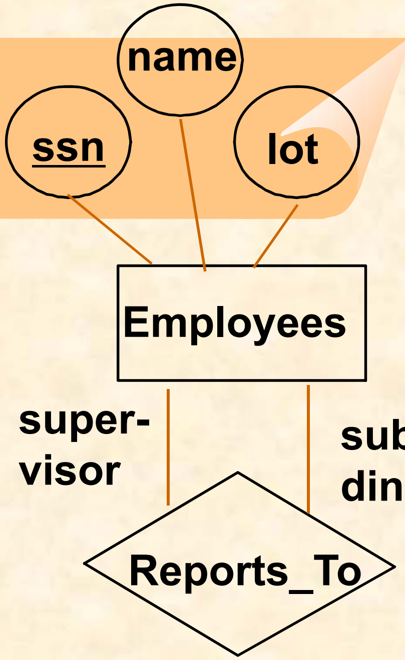
- Relationship between entity types
  - 부서장 : 부서를 관리하는 직원

    부서 ⇔ 직원

  - relationship
    - ER 초기과정 : entity의 속성간의 관계
    - ER 완성단계 : entity type간의 관계로 확정됨

ER Model Basics (Contd.)





- *Relationship*: Association among two or more entities.
- *Relationship Set*: Collection of similar relationships.
  - An n-ary relationship set R relates n entity sets E1...En
  - Each relationship in R involves entities e1 in E..,en in En

Relationship

- Relationship Type R:
  - n 개의 entity type E1, E2,…, En 간의 결합 집합 R
  - participation : E1, E2,…, En participate R
  - set of relationship instances
- Relationship Instance
  - entities e1, e2,…, en간의 결합 => instance ri (e1∈E1, e2∈E2, …, ri∈R)
  - participation : e1, e2,…,en participate ri
  - works_for relationship type on employee, department
- ER diagram
  - relationship type : 마름모

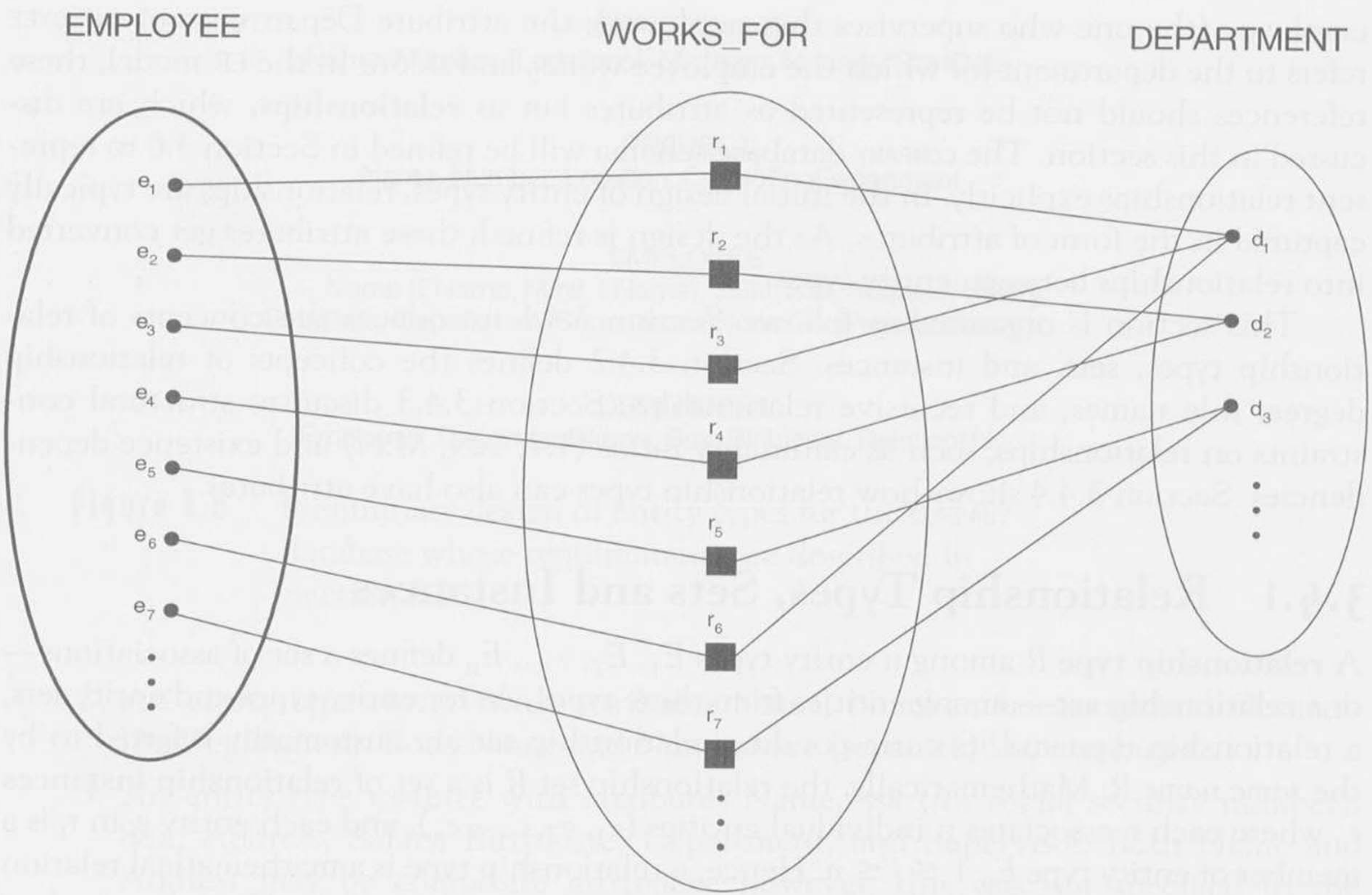


Figure 3.9 Some instances of the WORKS_FOR relationship between EMPLOYEE and DEPARTMENT.

Relationship

- degree of a relationship type
  - # of participating entity type on relationship type
  - binary relationship
    - works_for : degree = 2
  - ternary relationship
    - works_for : degree = 3
- Relationship as attributes (그림3.9)
  - 직원이 일하는 부서
    - 직원의 속성으로 부서를 봄
      - domain = all departments
  - 부서에서 일하는 직원
    - 부서의 속성으로 직원을 봄
      - domain = all employees

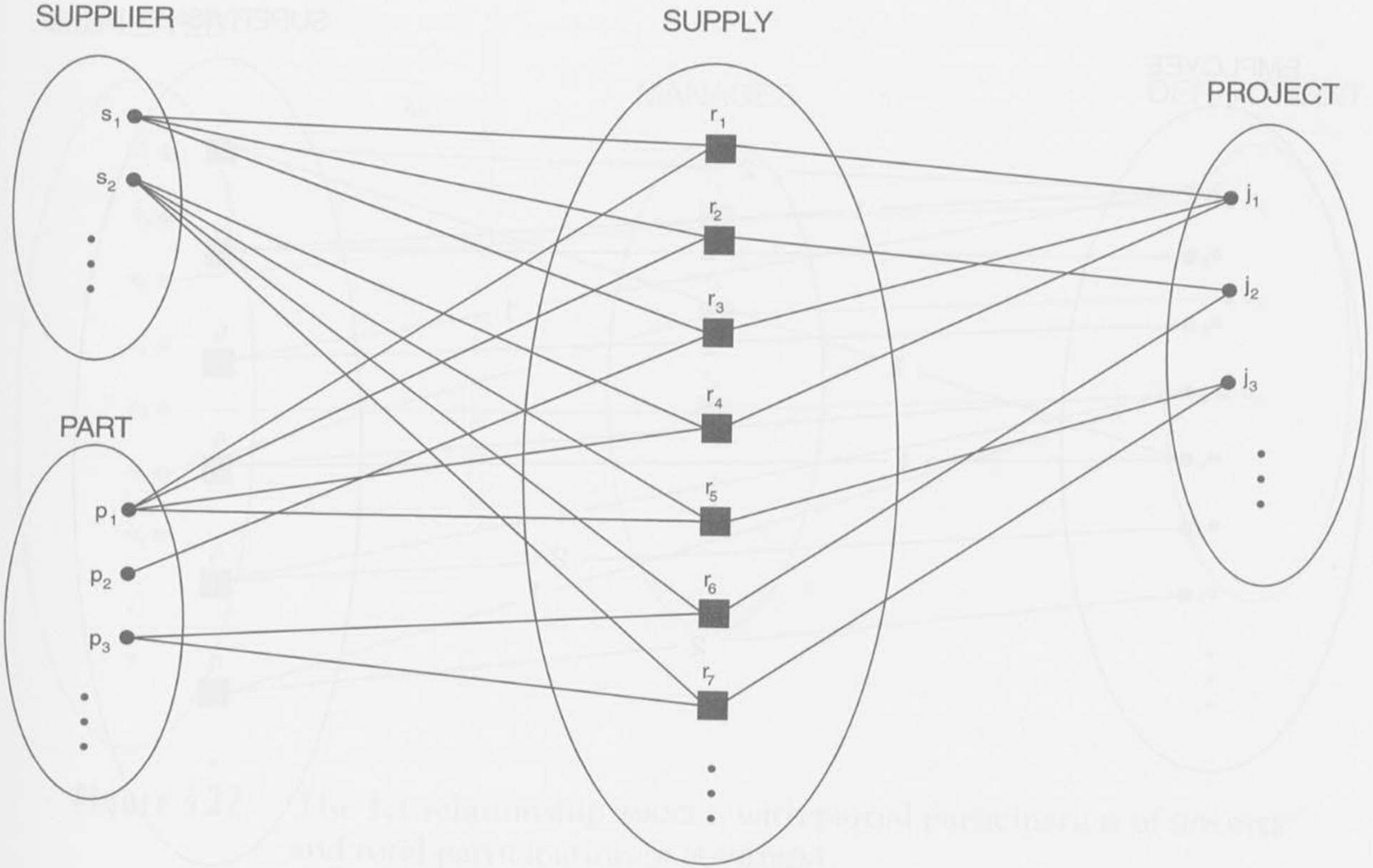


Figure 3.10 Some relationship instances of a ternary relationship SUPPLY.

Role Name and Recursive Relationship

- relationship에서의 entity type의 역활
  - 대부분 entity 이름으로 표현
  - recursive relationship
    - employee1 <--> supervision <--> employee2
    - 1 on edge : supervision role
    - 2 on edge : employee role
    - e1 supervise e2, e4 supervise e6 and e7

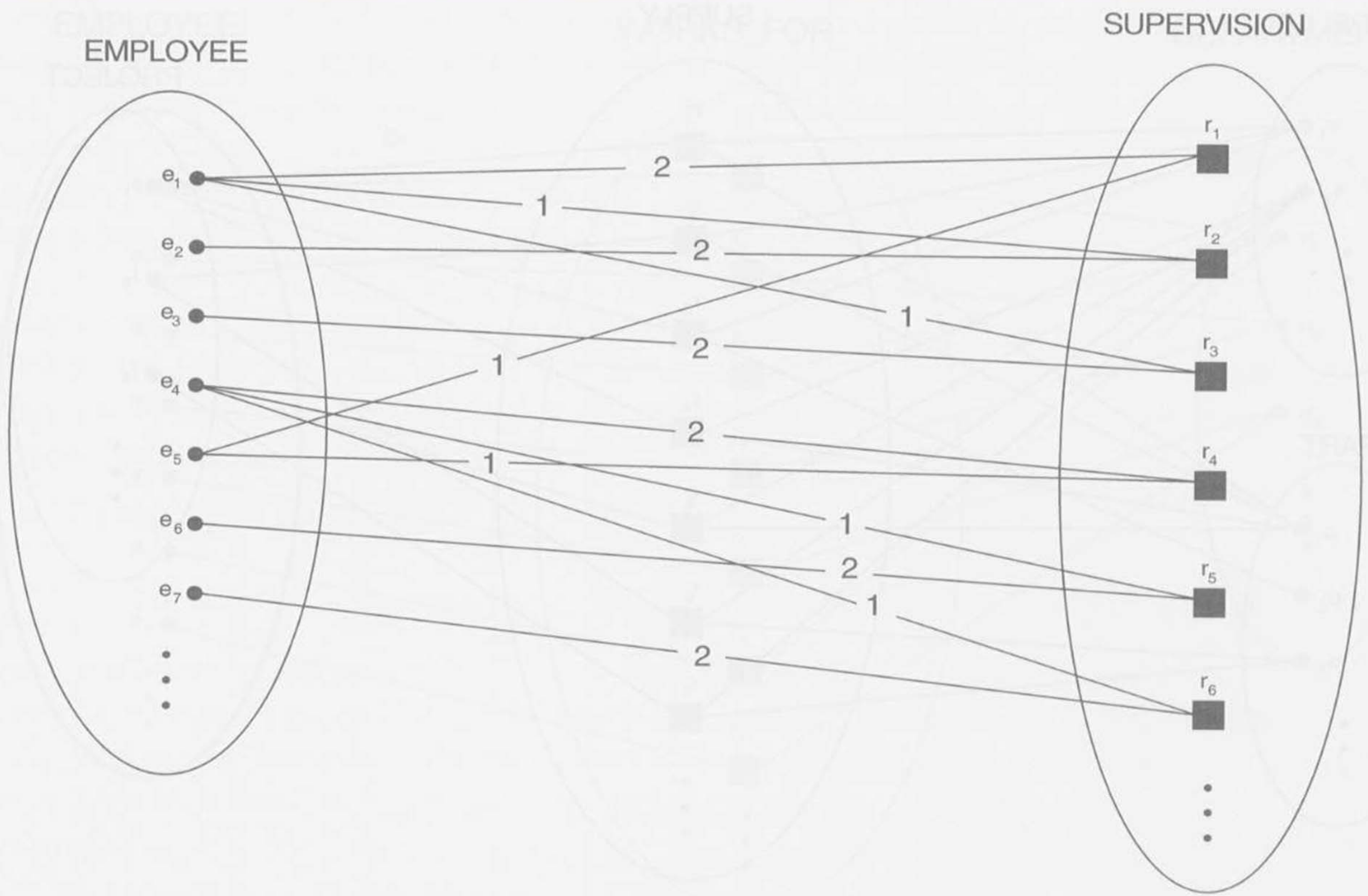


Figure 3.11 The recursive relationship SUPERVISION, where the EMPLOYEE entity type plays the two roles of supervisor (1) and supervisee (2).

Constraints on Relationship Types

- relationship <= constraints in mini-world
  - 예 : 만약 모든 직원은 오직 한 부서에 배치됨.(회사규정) => 스키마
- types of relationship constraints
  - cardinality ratio and participation
- cardinality ratio
  - # of relationship instances that an entity can participate in
    - 예 : Works_for – 부서:직원 = 1:N
      - 각 부서는 여러 직원이 일할 수 있지만 한 직원은 오직 한 부서로만 배치가 가능하다.

Constraints on Relationship Types

- Typical => 1:1, 1:N, M:N
  - 1:1 cardinality ratio (그림 3.12)
    - 부서장:부서 = 1:1
      - 부서에는 부서장이 한명이며, 한 부서장은 오직 한 부서만을 관리한다.
  - M:N cardinality ratio (그림 3.13)
    - 직원:과제 = M:N
      - 각 직원은 여러 개의 과제를 수행할 수 있고, 각 과제는 여러 직원이 수행할 수 있다.

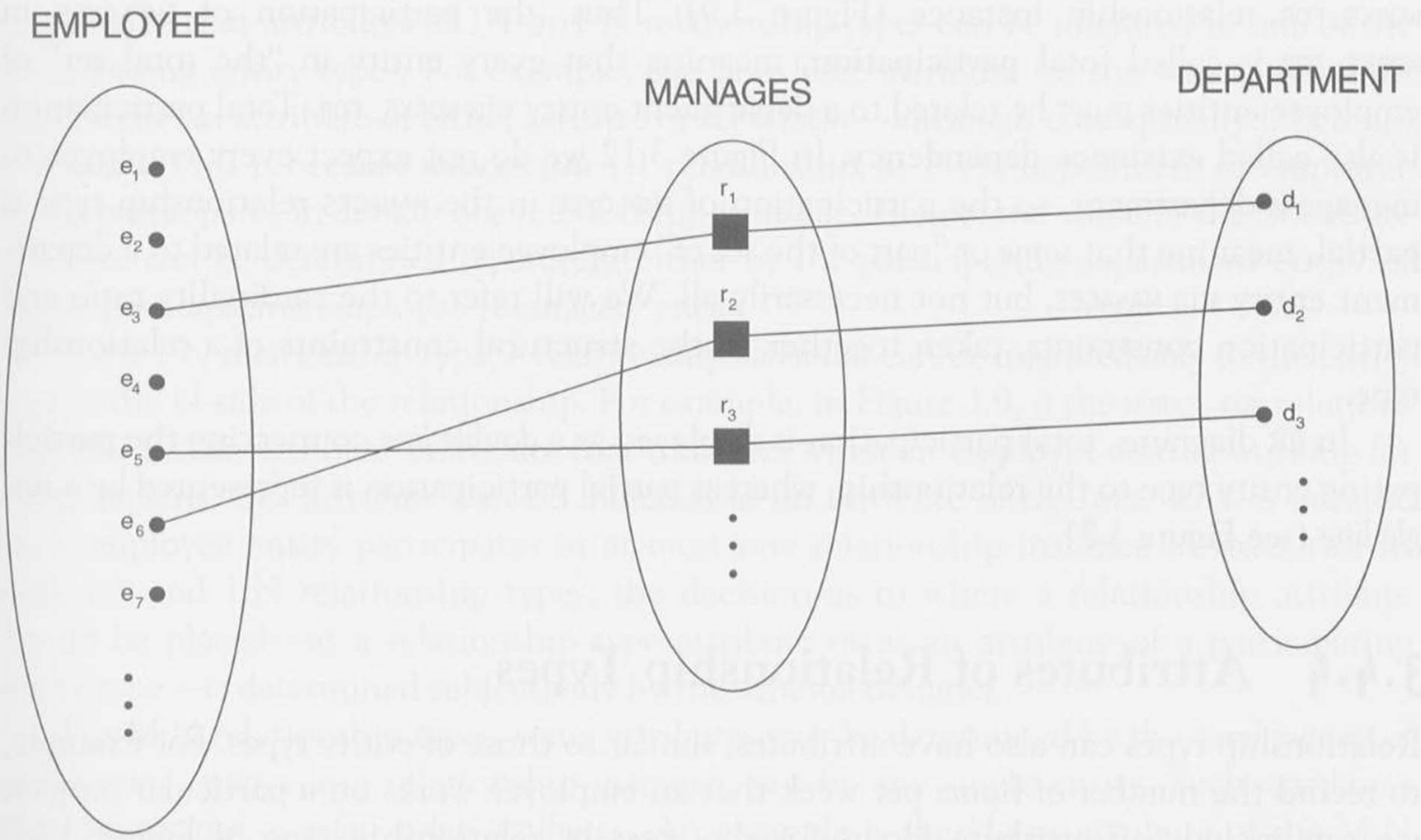


Figure 3.12 The 1:1 relationship MANAGES, with partial participation of EMPLOYEE and total participation of DEPARTMENT.

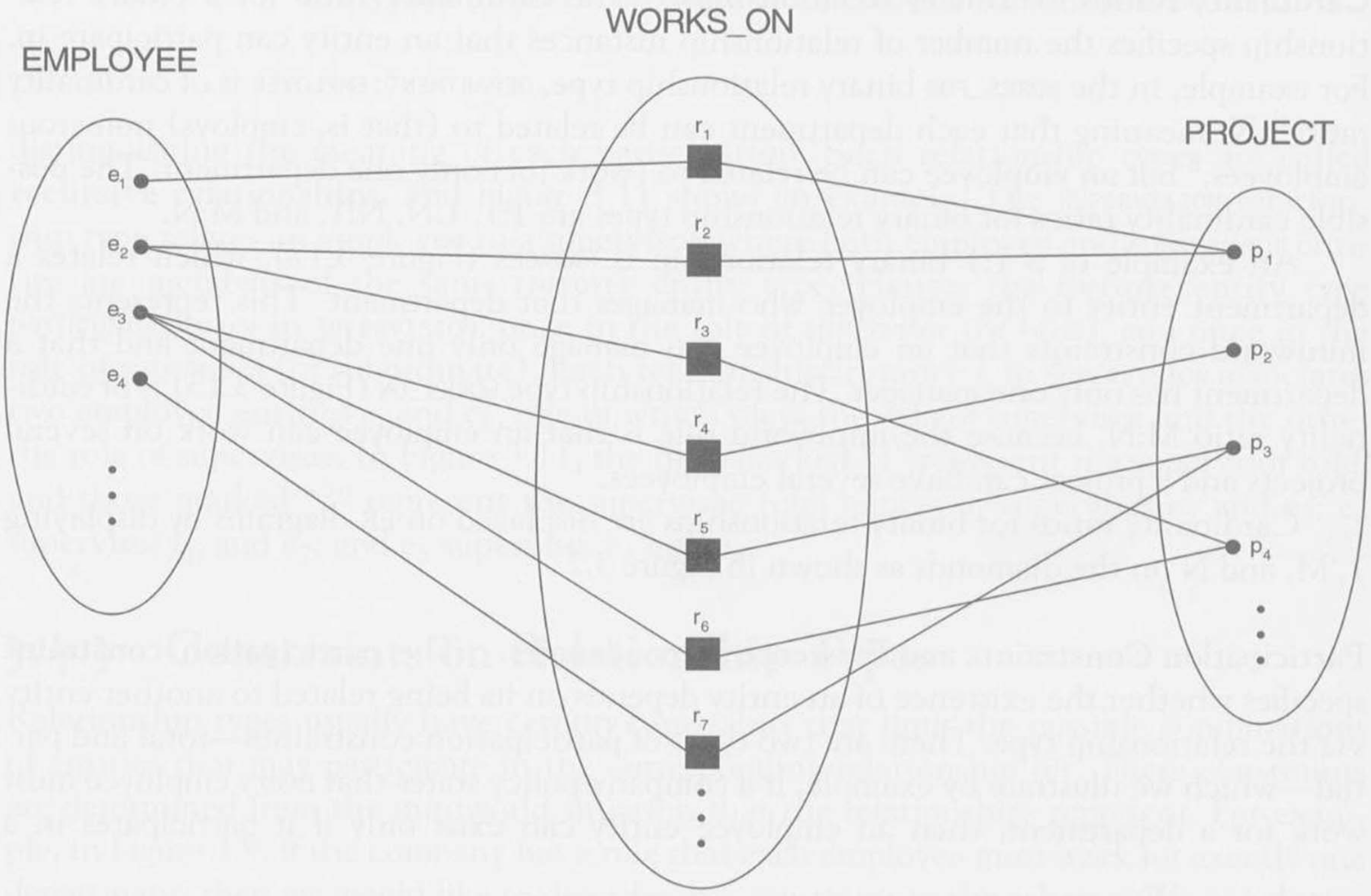


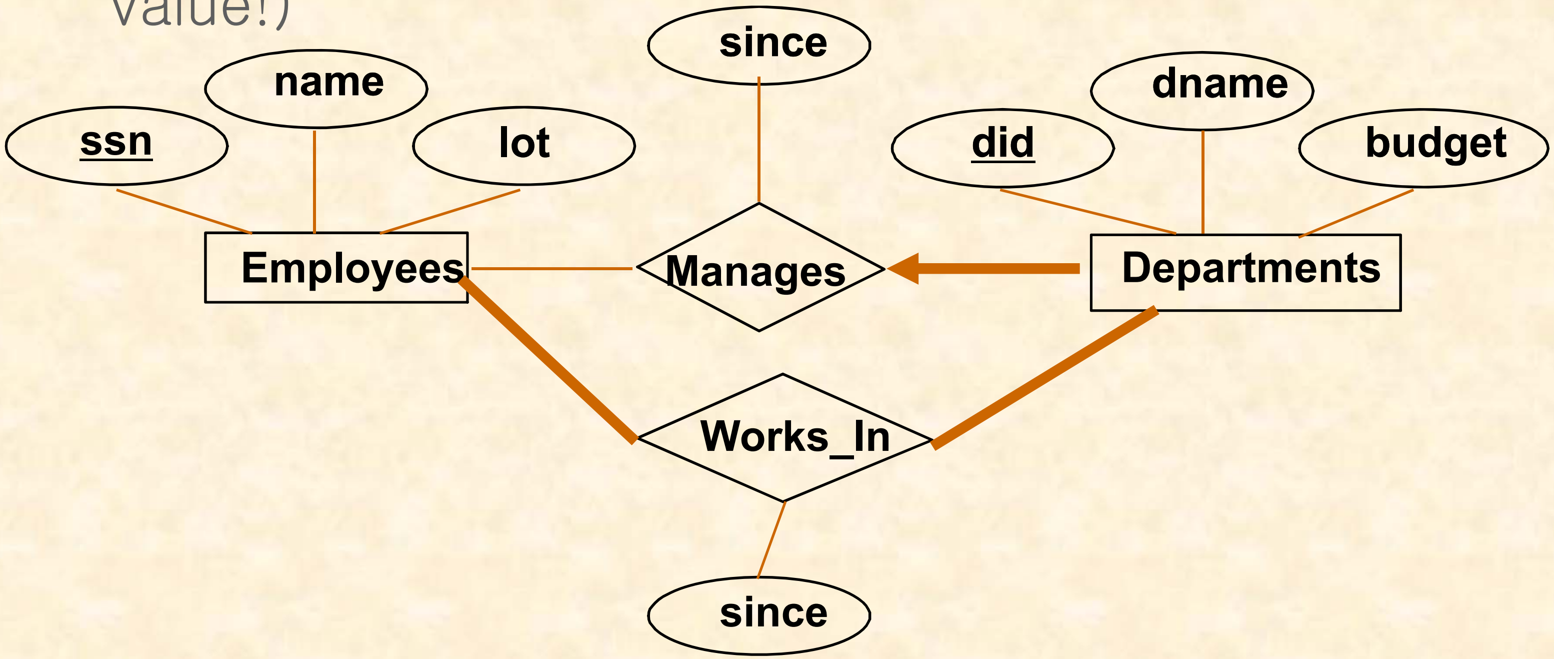
Figure 3.13 The M:N relationship WORKS_ON between EMPLOYEE and PROJECT.

Constraints on Relationship Types

- participation
  - entity의 relationship type의 참여 제약조건
  - total participation (existence dependency)
    - 만약 회사규정이 모든 직원은 한 부서에서 일해야 한다면 [employee <--- Works_for ---> department]
      - 모든 직원의 entity들은 한 부서와 works-for라는 관계가 있어야 한다.
    - ER diagram:double line entity type<->relationship
  - partial participation
    - 일부 직원은 다른 직원과 manage라는 관계로 연결됨
    - [employee <---manage ---> employee]
    - ER diagram:single line (1, M, N)
  - structural constraints of relationship type = cardinality constraints + participation constraints

Participation Constraints

- Does every department have a manager?
  - If so, this is a *participation constraint*: the participation of Departments in Manages is said to be *total* (vs. *partial*).
    - Every *did* value in Departments table must appear in a row of the Manages table (with a non-null *ssn* value!)



Attributes of Relationship Types

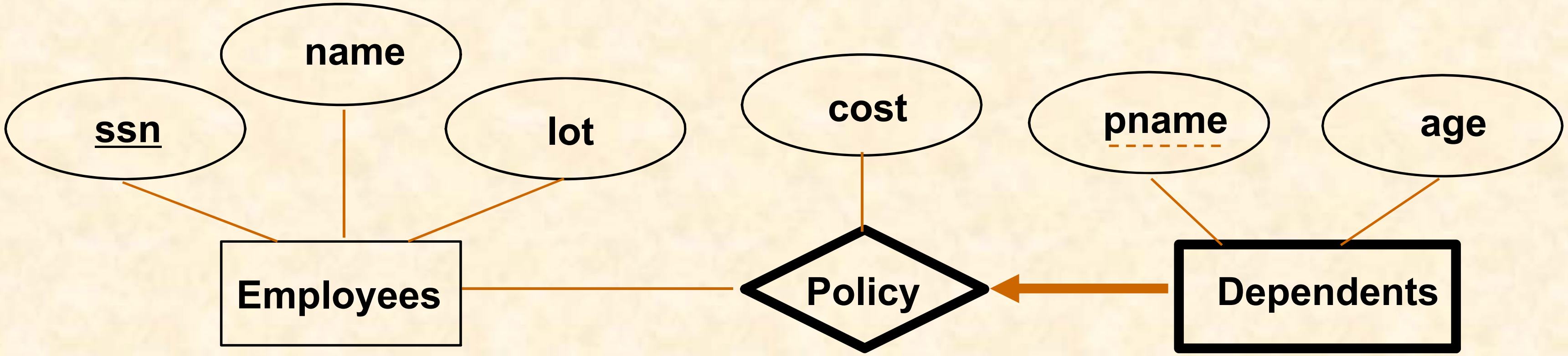
- attribute migration
  - 1:1 – attributes of a relationship type can be migrated to either side
  - 1:N – attributes of a relationship type can be migrated to an entity type at N side
  - M:N – must be relationship attribute
    (no migration)

Weak Entity Types

- weak entity types
  - entity type with no key attribute
  - *가족(이름, 성별, 생일, 관계) <- dependent_of -> 직원 = N:1 relationship*
    - *가족 : no key attributes*
    - Identifying relationship type & identifying owner entity type
  - identifying relationship type
    - 항상 total participation constraint
    - ER diagram : double line
  - weak entity type은 partial key 갖는다
    - partial key : 동일한 owner entity와 연관된 entity를 유일하게 찾는 key (ER diagram : dot line)

Weak Entities

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
  - Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
  - Weak entity set must have total participation in this *identifying* relationship set.



Refining Company ER diagram

- 초기 entity 설계 (그림 3.8)
  - 부서(이름,번호,{위치},부서장,부서장시작날짜)
  - 과제(이름,번호,위치,주관부서)
  - 직원(이름,이름,성),주민등록번호,성별,주소,월급,생일,부서,관리자,{수행과제(과제,시간)})
  - 가족(직원,가족이름,성별,생일,관계)
- define relationship type
  - manages => 1:1 relationship type, 직원 ↔ 부서
    - 직원 : partial participation
    - 부서 : not clear on requirements => total participation
    - 속성 : StartDate

Refining Company ER diagram

- define relationship type
  - works_for – 1 : N relationship type, 부서 ↔ 직원
    - both : total participation
  - controls – 1:N relationship type, 부서 ↔ 과제
    - 과제 : total participation
    - 부서 : partial participation(과제가 없는 부서존재)
  - supervision – 1:N relationship type, 직원 ↔ 직원
    - both : partial (관리자가 없는 직원이 있을 수 있고, 모든 직원이 관리자가 아님)
  - works_on – M:N relationship type 직원 ↔ 과제
    - both : total participation (과제가 없는 직원이 없고 직원에 할당되지 않은 과제가 없음)
    - 속성 : hour

Refining Company ER diagram

- define relationship types
  - dependent_of – 1 : N relationship type, 직원 ↔ 가족
    - identifying relationship
    - 가족 : weak_entity type, total participation (가족은 identifying owner가 항상 존재)
    - 직원 : identifying owner, partial participation (가족이 없는 직원이 있음)
- remove all attributes that have been refined
  - 부서장, 부서장시작날짜 from 부서
  - 주관부서 from 과제
  - 부서, 관리자, works_on from 직원
  - 직원 from 가족
- 중복의 최소화

Entity-Relationship(ER) Diagram

- ER diagram notation (그림 3.14)

- alternative notation for structural constraint
  - entity type E <- (min, max) -> relationship type R
    - e in E는 항상 적어도 min 보다는 많이 max 보다는 적게 R에 참여.
  - participation of E in R
    - min = 0 : partial participation
    - min > 0 : total participation
  - easy specification of structural constraints for relationship types of any degree

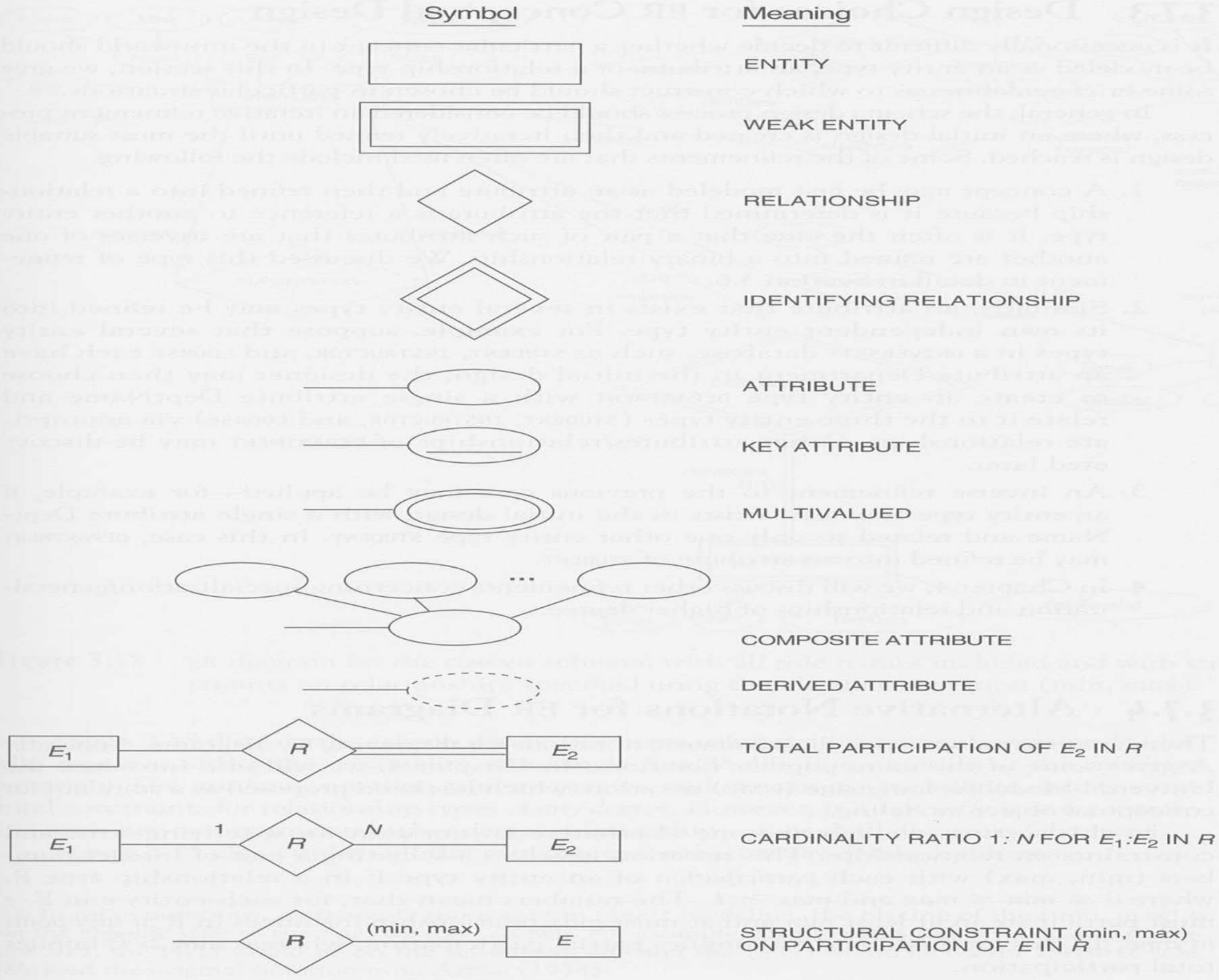
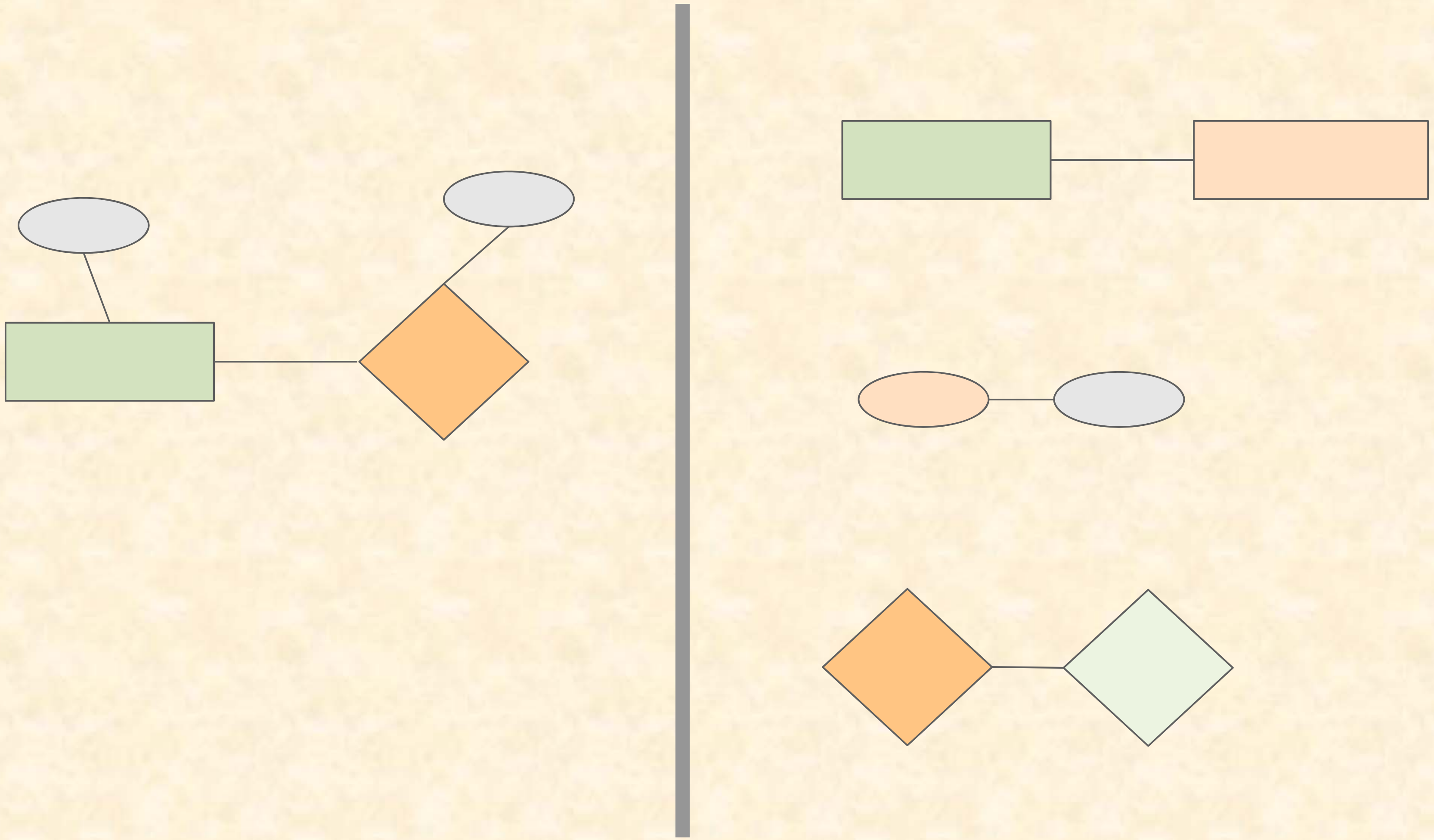
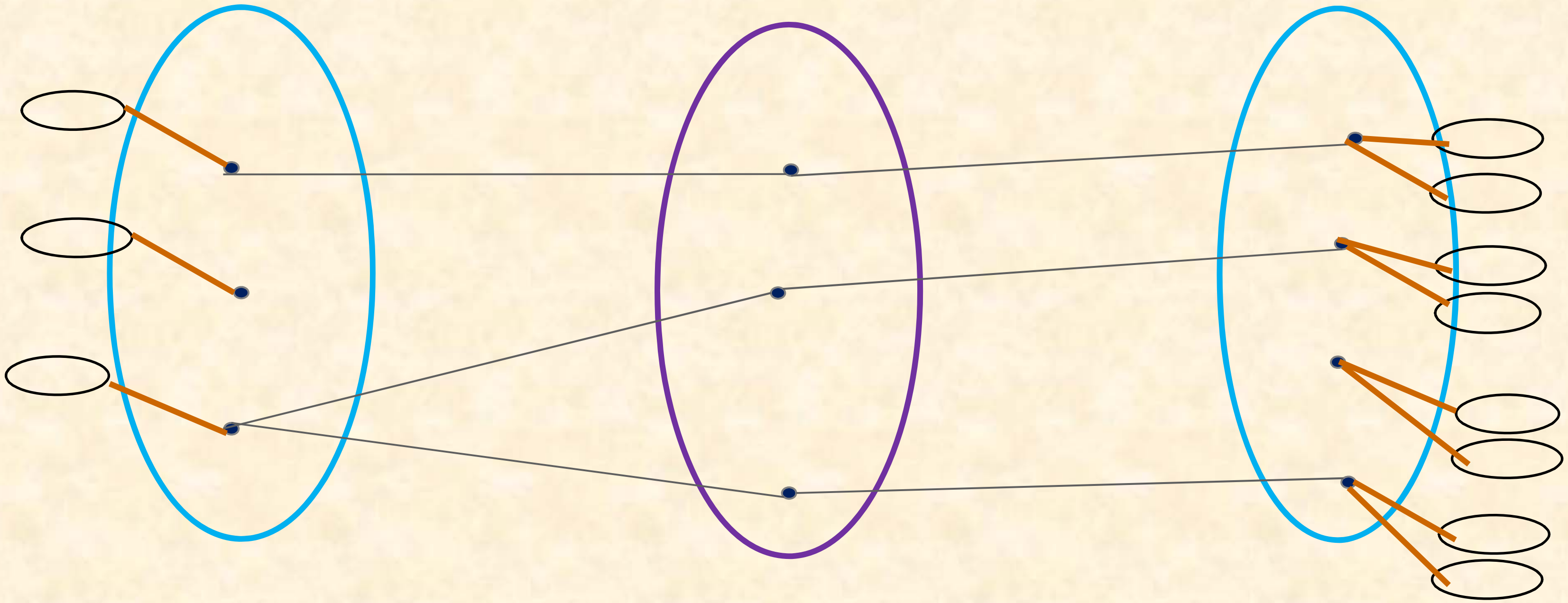


Figure 3.14 Summary of ER diagram notation.

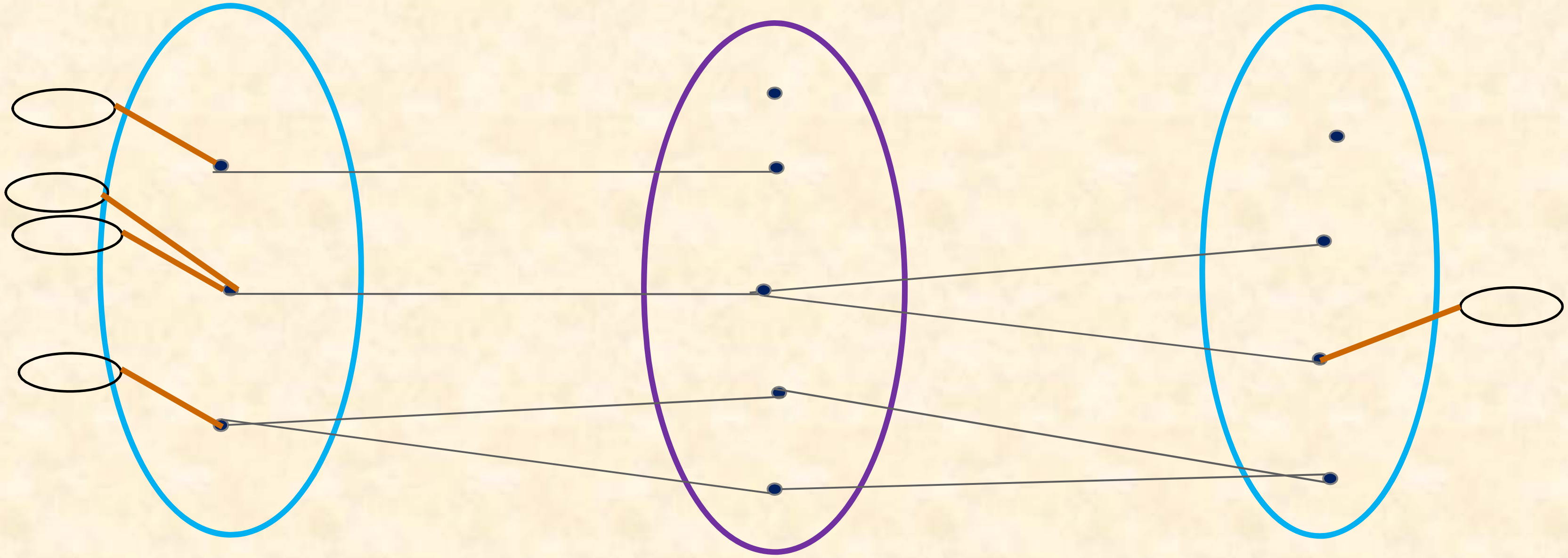
Entity vs. Attribute (Contd.)



Entity vs. Attribute (Contd.)



Entity vs. Attribute (Contd.)



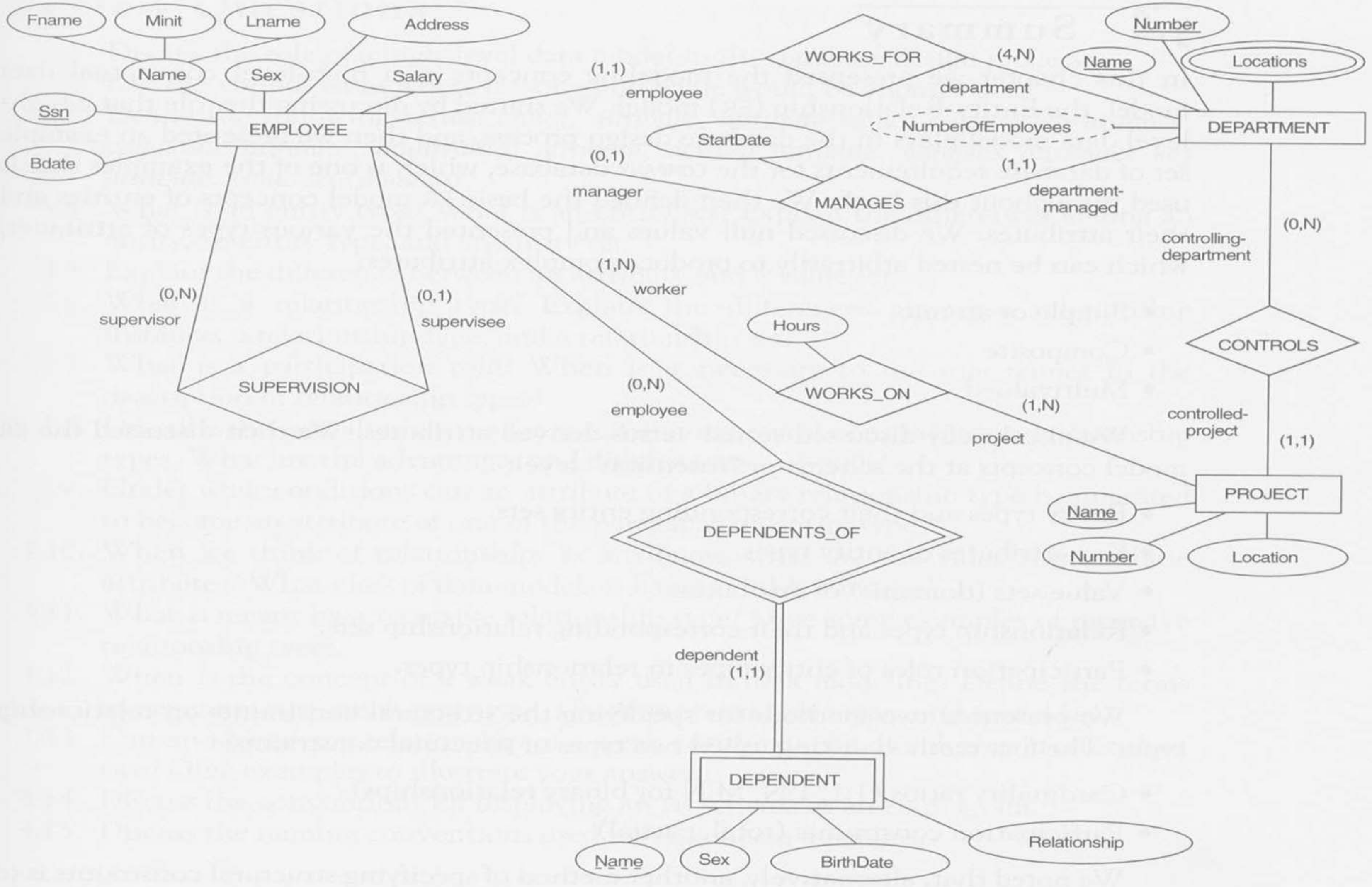


Figure 3.15 ER diagram for the COMPANY schema, with all role names included and with structural constraints on relationships specified using the alternate notation (min, max).

스키마 요소의 이름

- 스키마 요소 이름
  - entity type, attributes, relationship type의 작명중요
  - entity type : 단수 사용 (각 entity에 적용됨)
  - entity type and relationship type : uppercase 사용
  - attribute name : uppercase(첫 문자) + lowercase
  - role name : lowercase letter
- From DB requirements
  - 명사 : entity type name
  - 동사 : relationship type name
  - entity type 명사를 묘사하는 명사 : attribute name

스키마 요소의 이름

- relationship type 이름 (in ER diagram)
  - left → right
  - top → bottom
  - more readable ER diagram

Overview of Database Design

- *Conceptual design*: (*ER Model is used at this stage.)*
  - What are the *entities* and *relationships* in the enterprise?
  - What information about these entities and relationships should we store in the database?
  - What are the *integrity constraints* or *business rules* that hold?
  - A database schema in the ER Model can be represented pictorially (*ER diagrams*).
  - Can map an ER diagram into a relational schema.

Conceptual Design Using the ER Model

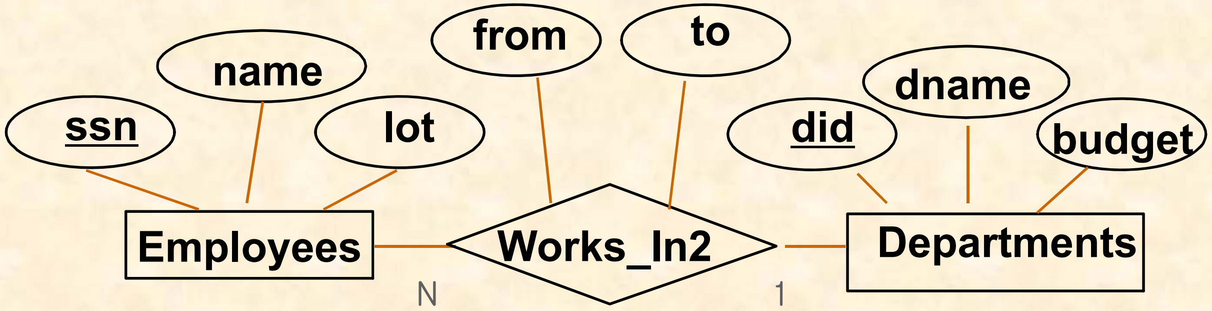
- Design choices:
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: Binary or ternary?
- Constraints in the ER Model:
  - A lot of data semantics can (and should) be captured.
  - But some constraints cannot be captured in ER diagrams.

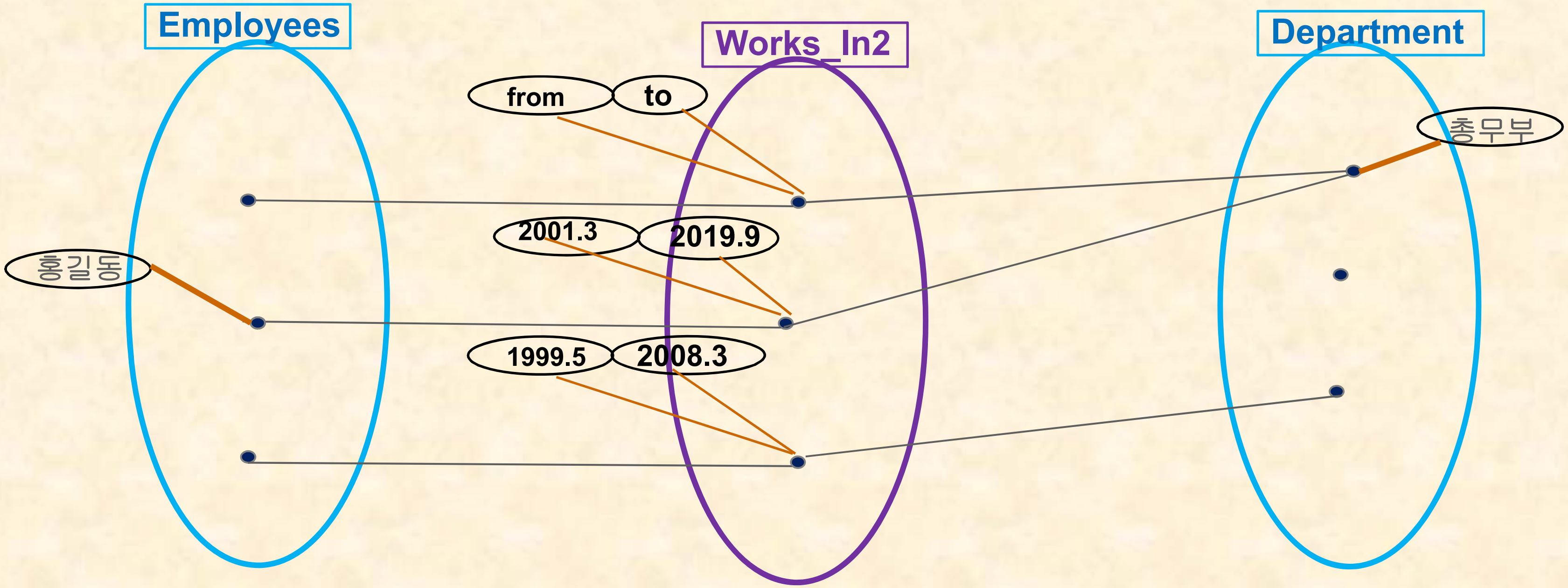
Entity vs. Attribute

- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends upon the use we want to make of address information, and the semantics of the data:
  - If we have several addresses per employee, *address* must be a multi-valued attribute entity
  - If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, *address* must be modeled as a composite attribute

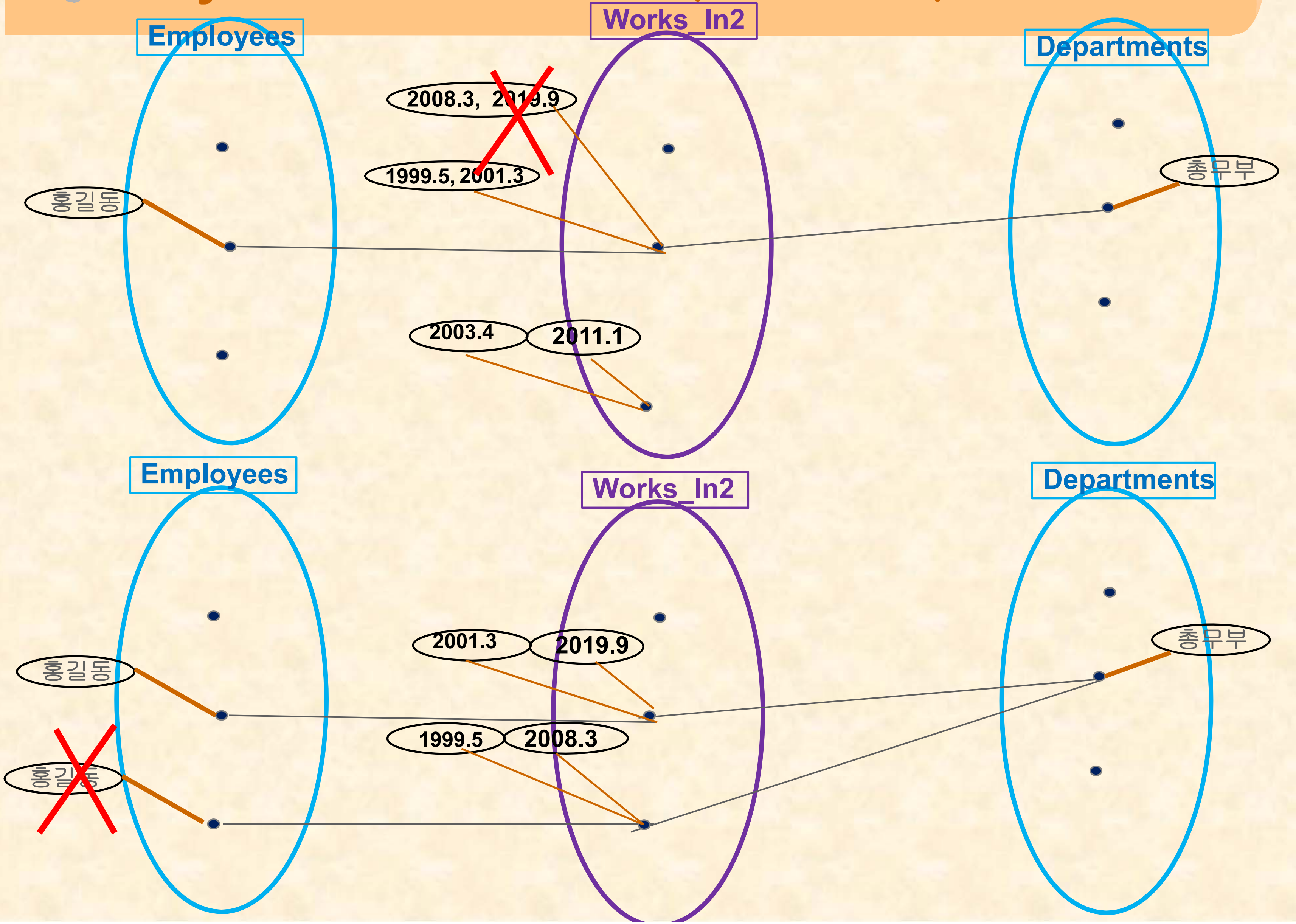
Entity vs. Attribute (Contd.)

- Works_In2 does not allow an employee to work in a department for two or more periods.

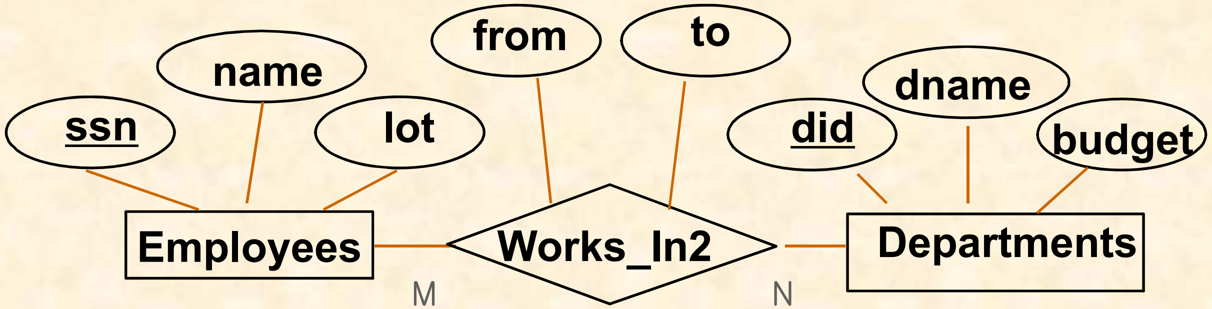


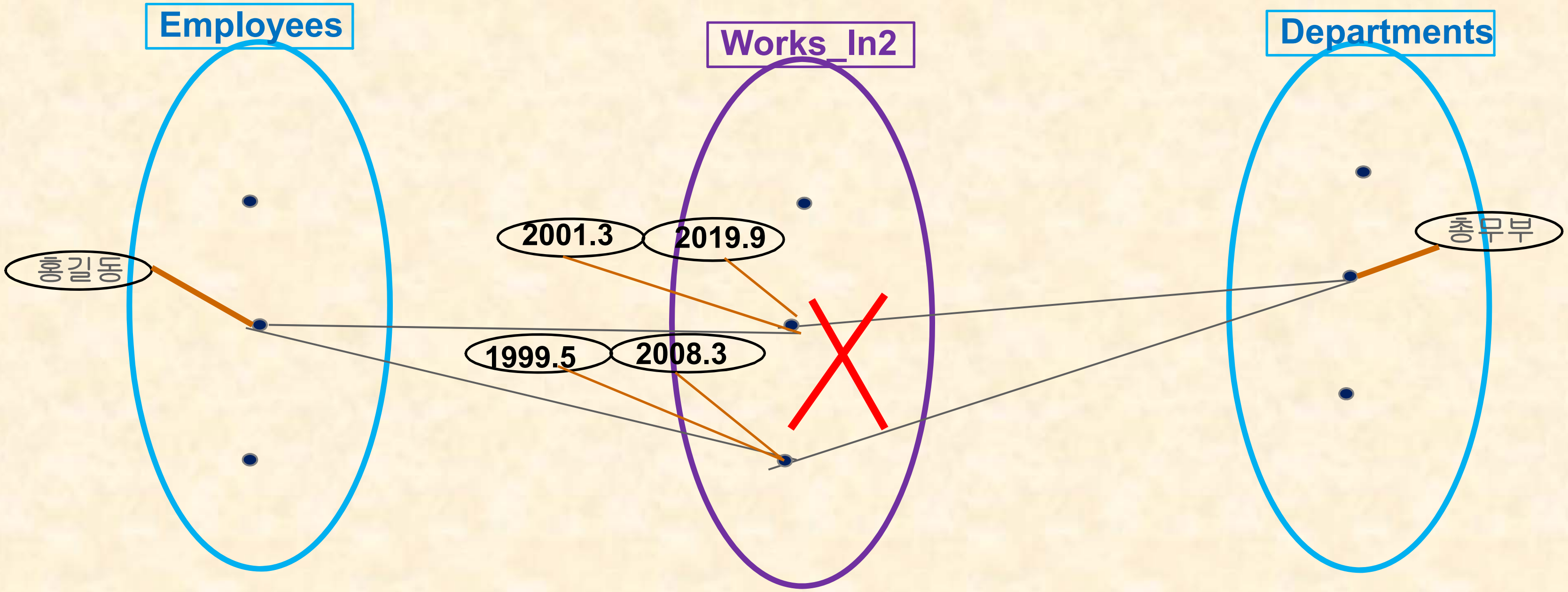


Entity vs. Attribute (Contd.)



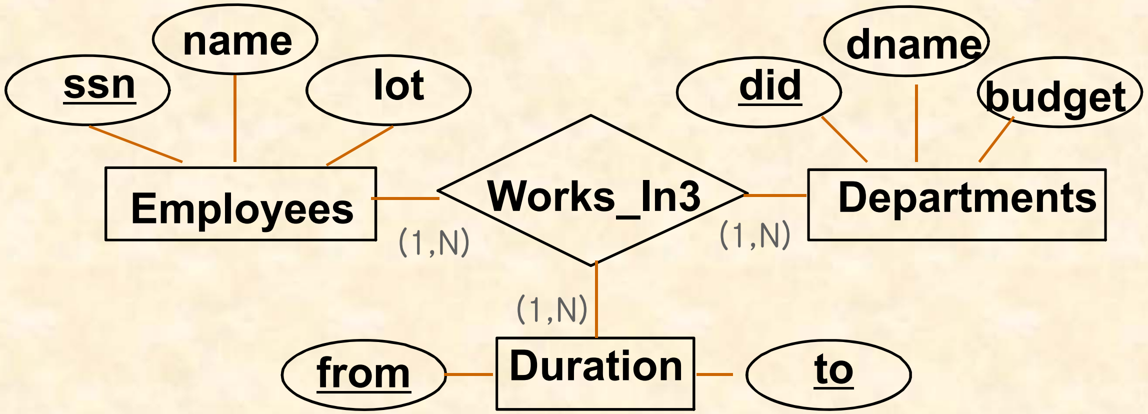
Entity vs. Attribute (Contd.)



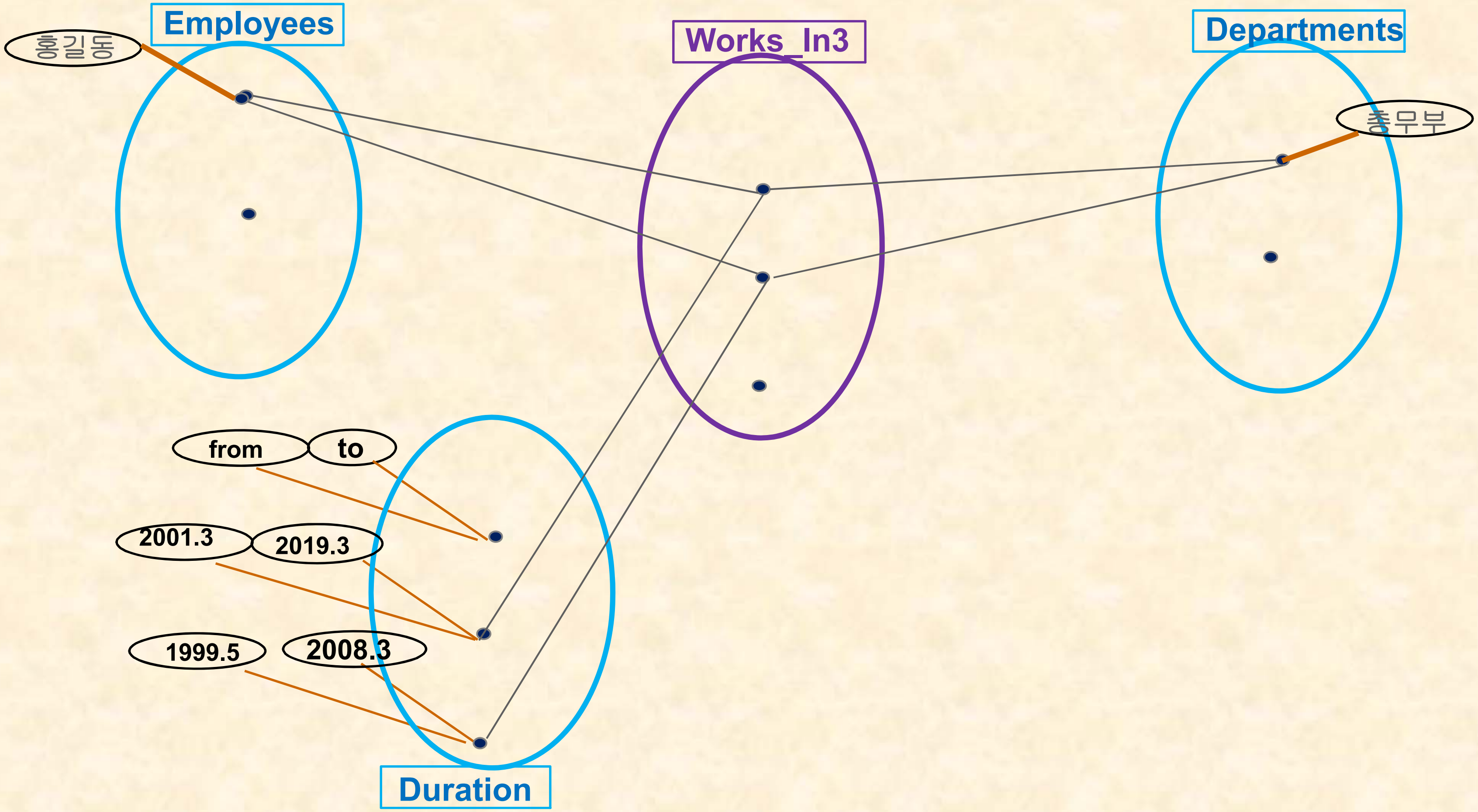


Entity vs. Attribute (Contd.)

- Similar to the problem of wanting to record several addresses for an employee: we want to record *several values of the descriptive attributes for each instance of this relationship.*

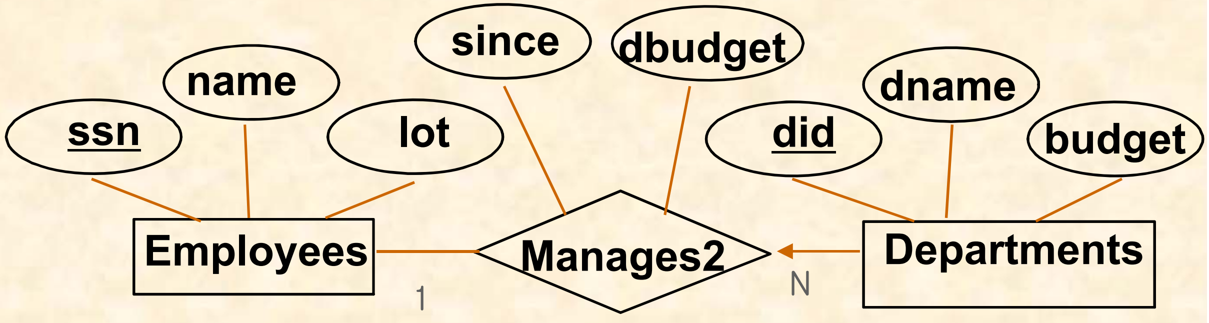


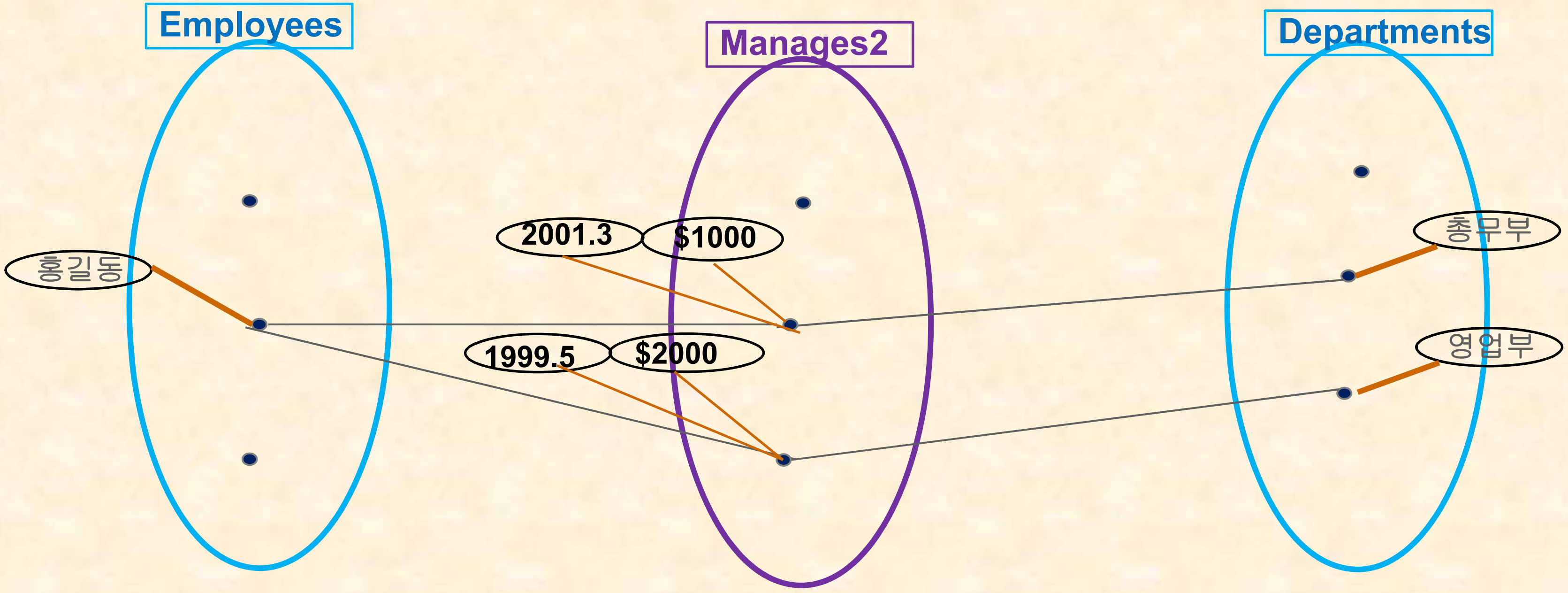
Entity vs. Attribute (Contd.)



Entity vs. Relationship

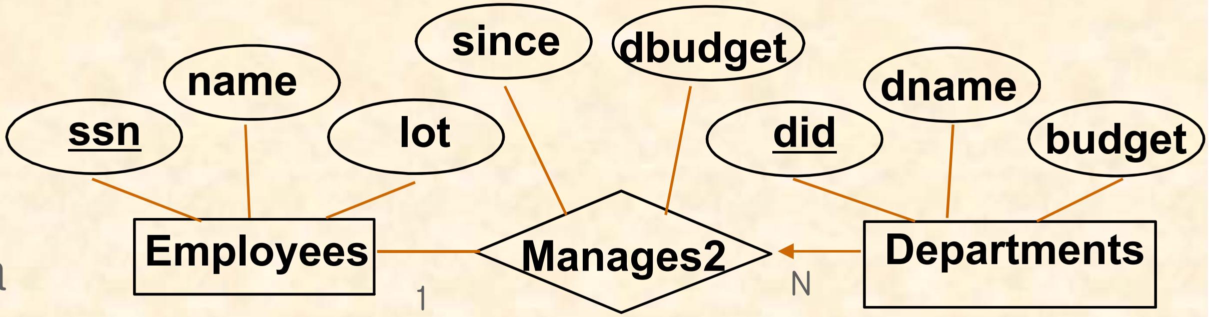
- First ER diagram OK if a manager gets a separate discretionary budget for each dept.

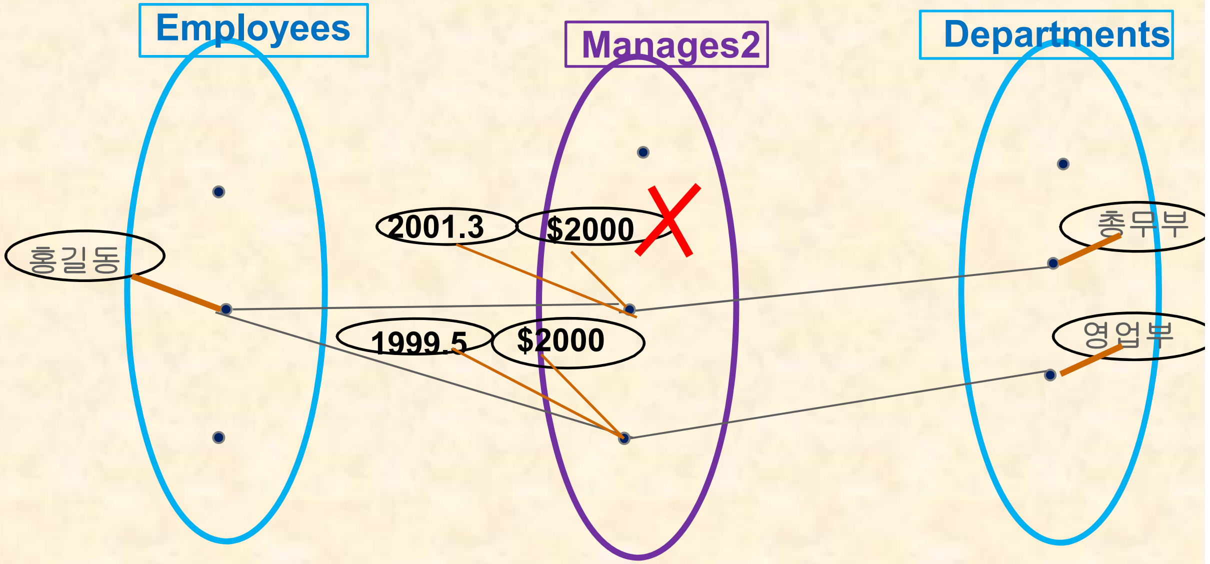




Entity vs. Relationship

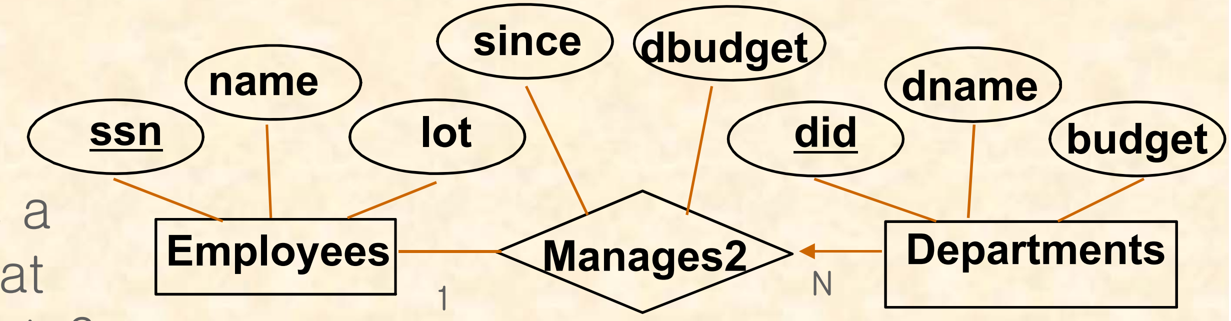
- First ER diagram OK if a manager gets a separate discretionary budget for each dept.
- What if a manager gets a discretionary budget that covers *all* managed depts?
  - Redundancy of *dbudget,* which is stored for each dept managed by the manager.

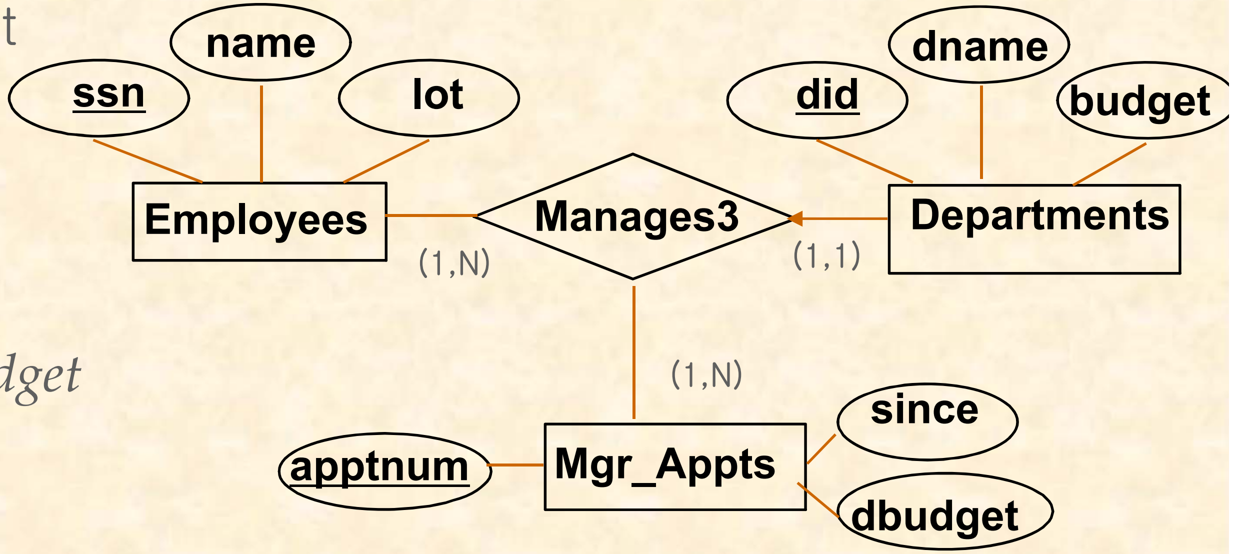




Entity vs. Relationship

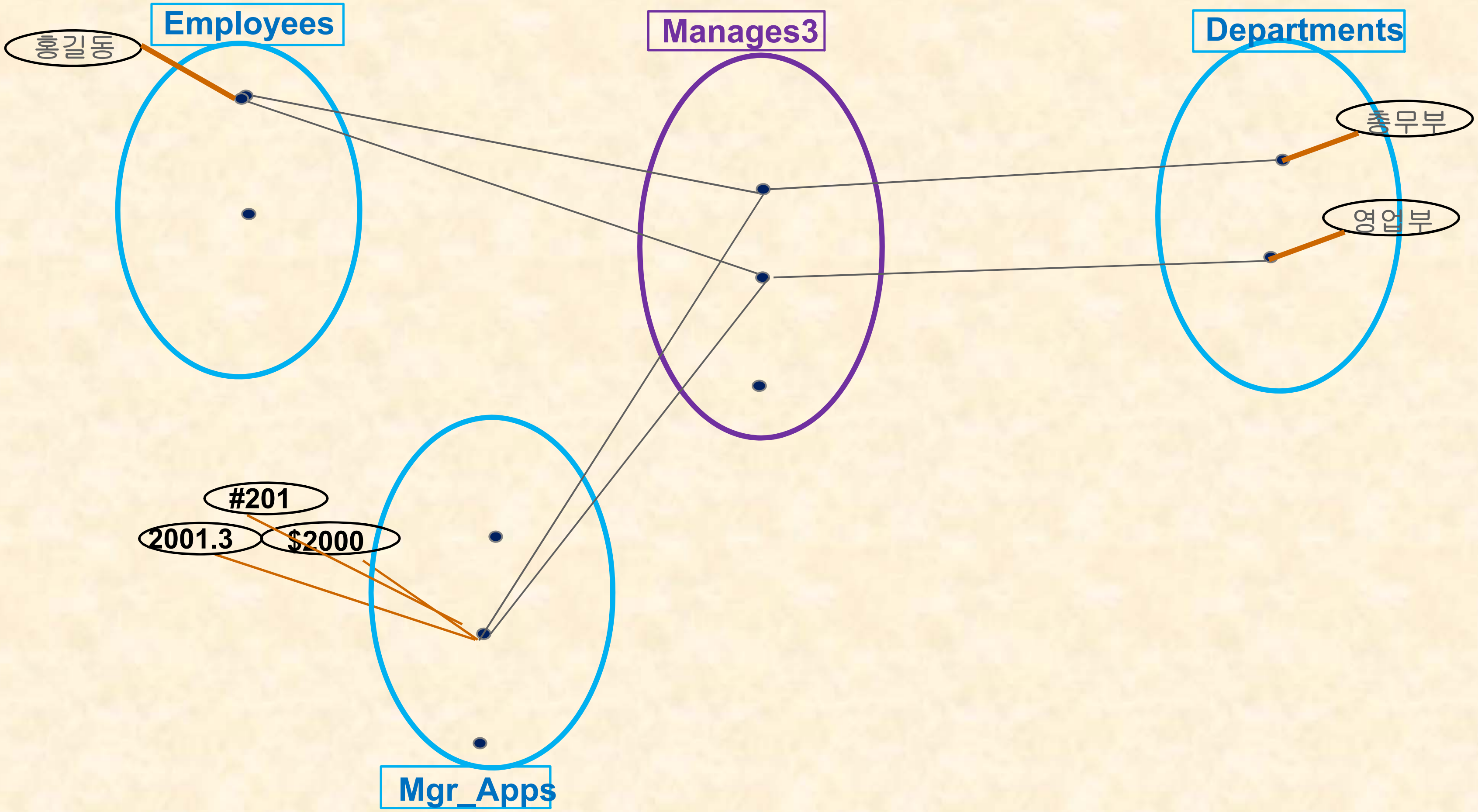
- What if a manager gets a discretionary budget that covers *all* managed depts?
  - Redundancy of *dbudget,* which is stored for each dept managed by the manager.





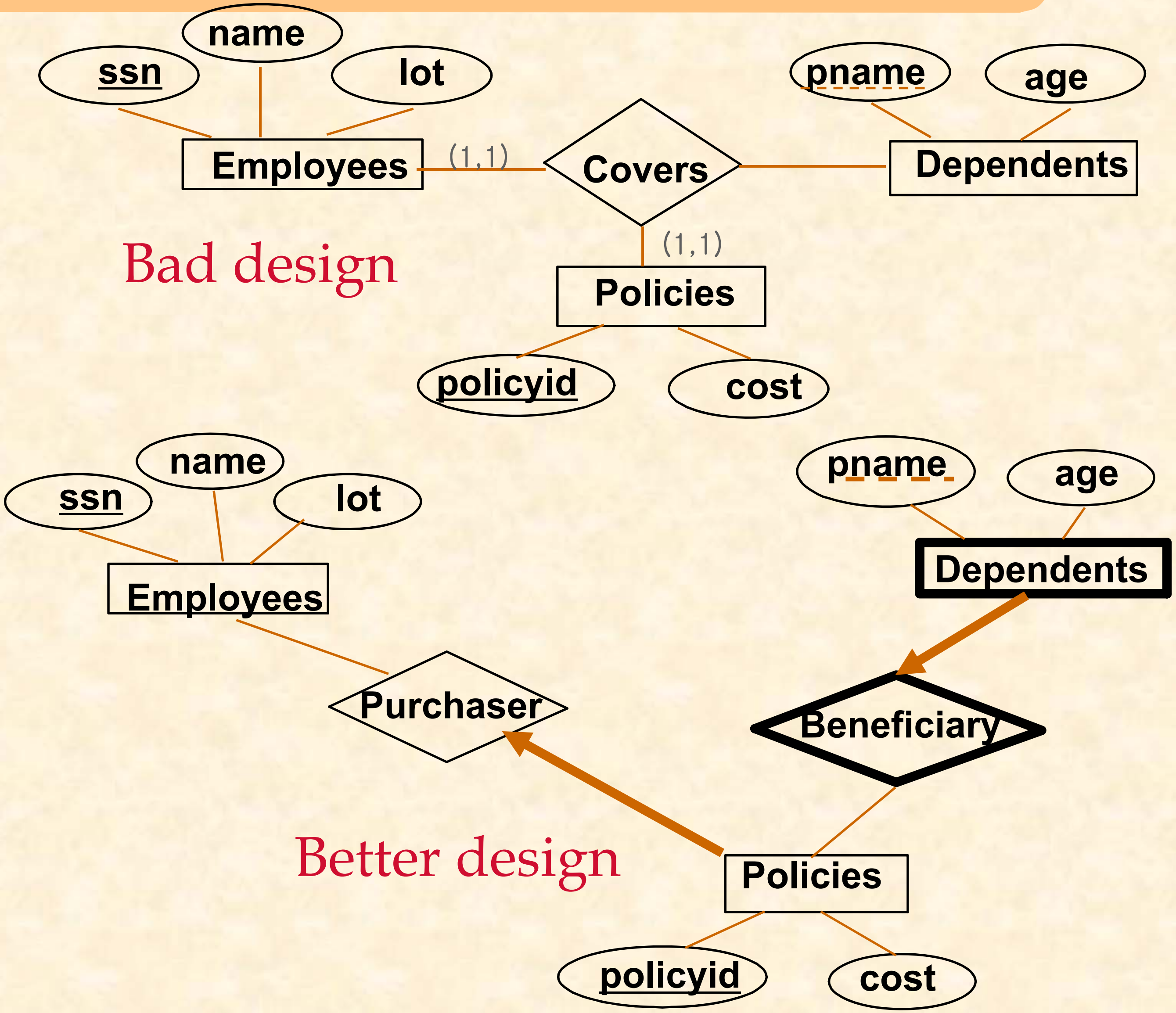
Misleading: suggests *dbudget* tied to managed dept.

Entity vs. Attribute (Contd.)

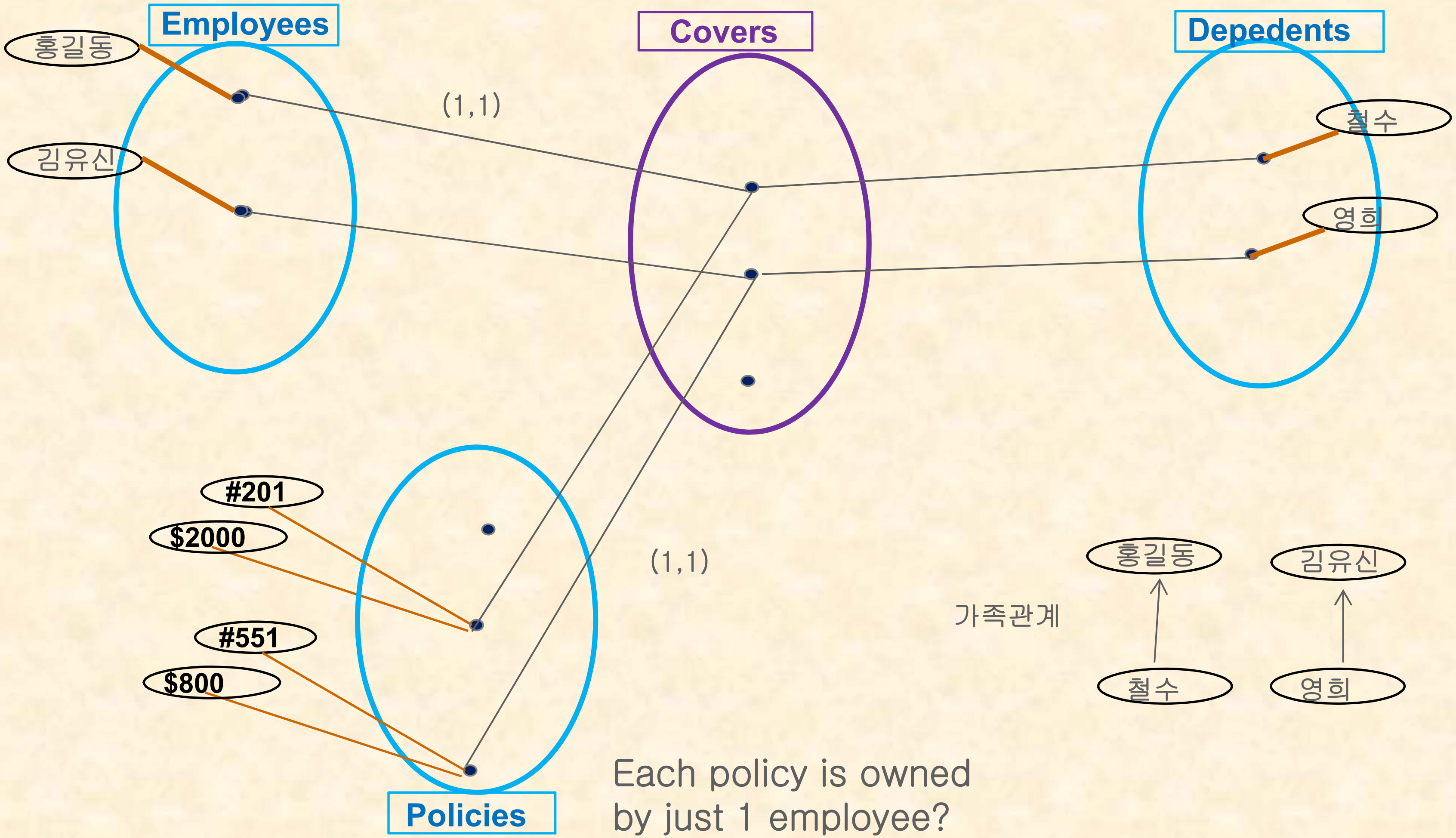


Binary vs. Ternary Relationships

- If each policy is owned by just 1 employee:
  - Key constraint on Policies would mean policy can only cover 1 dependent!
- What are the additional constraints in the 2nd diagram?

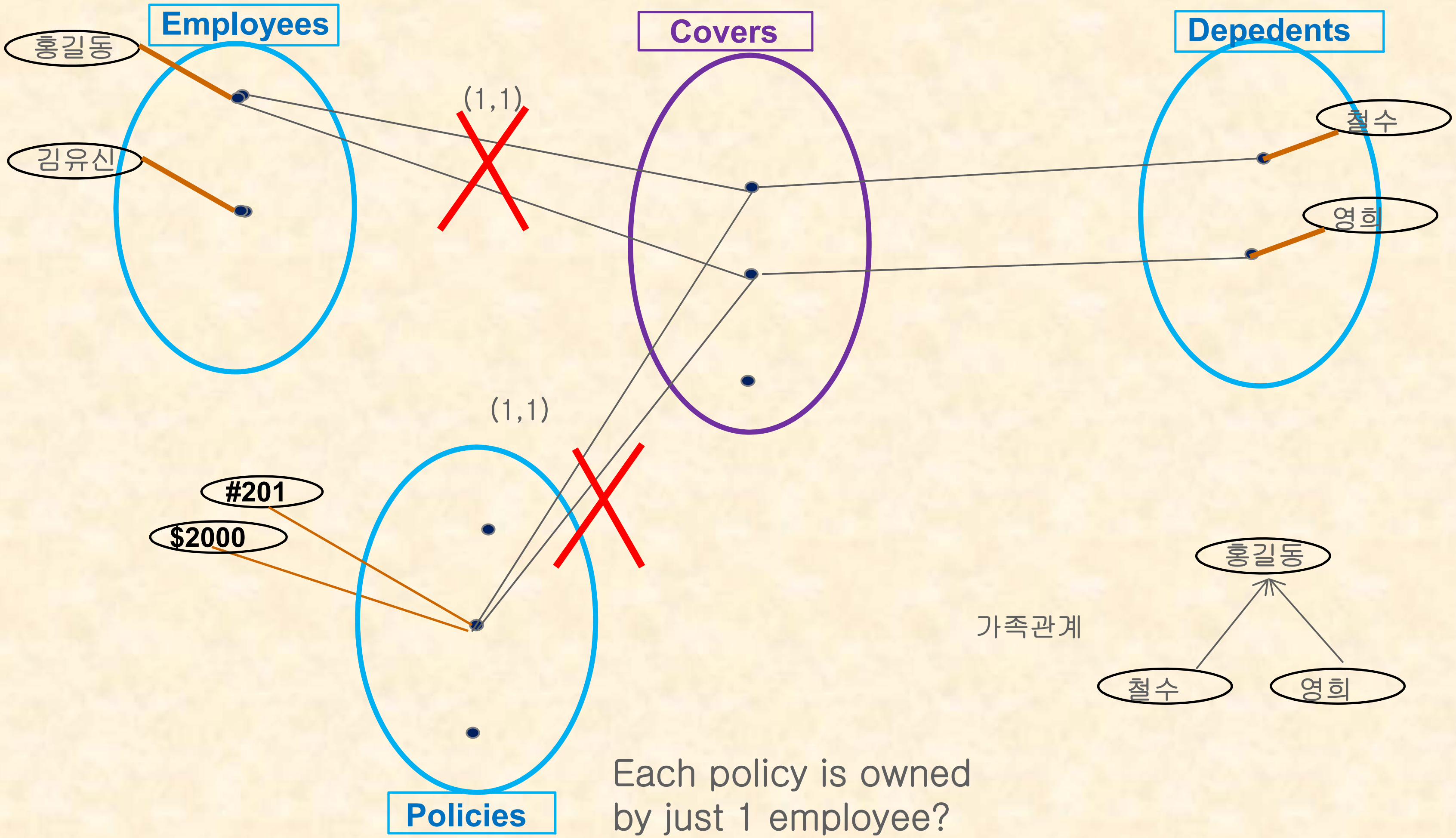


Entity vs. Attribute (Contd.)



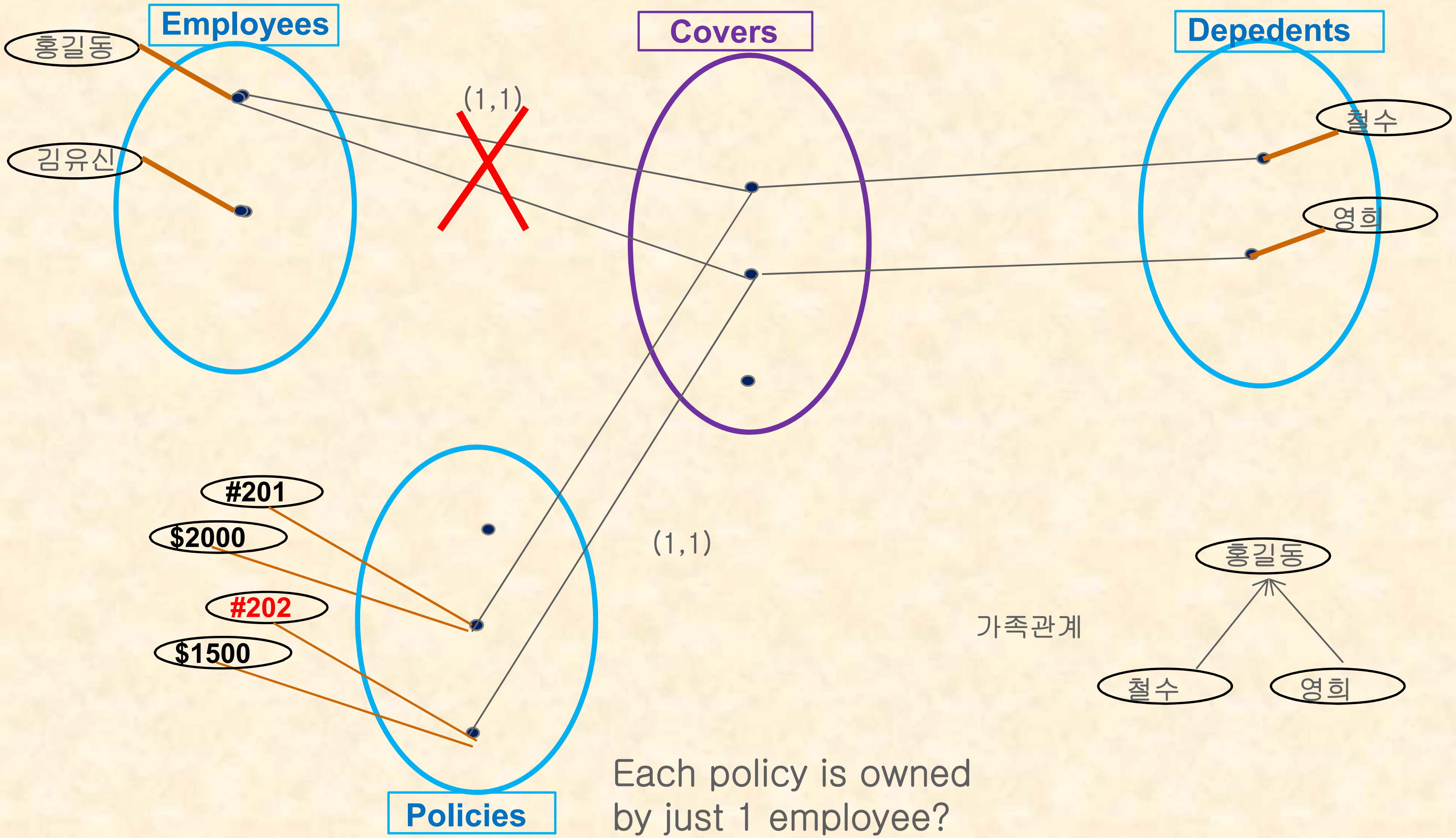
Each policy is owned by just 1 employee?

Entity vs. Attribute (Contd.)



Each policy is owned
by just 1 employee?

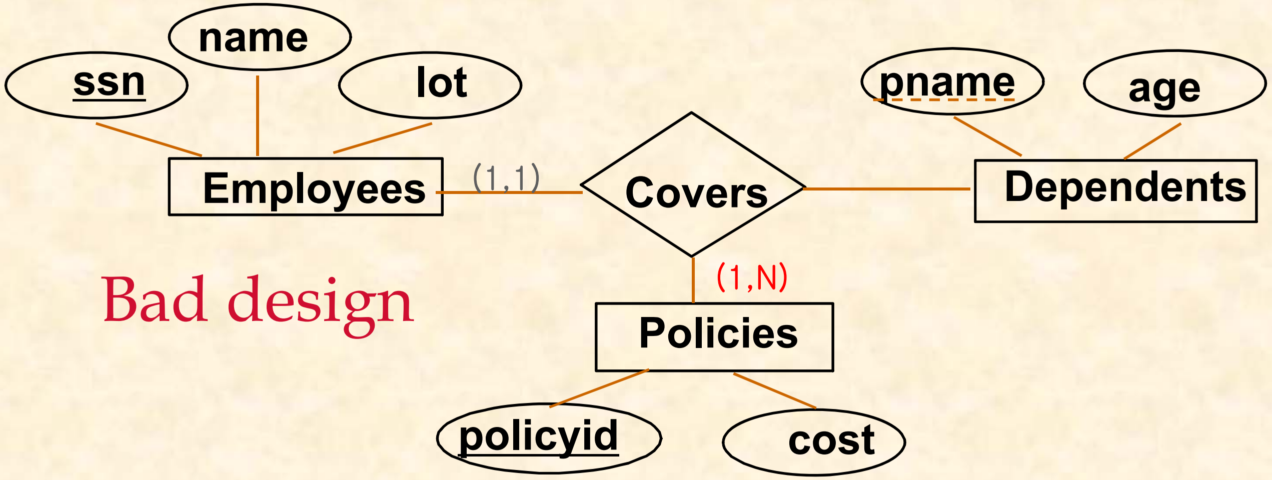
Entity vs. Attribute (Contd.)



Each policy is owned by just 1 employee?

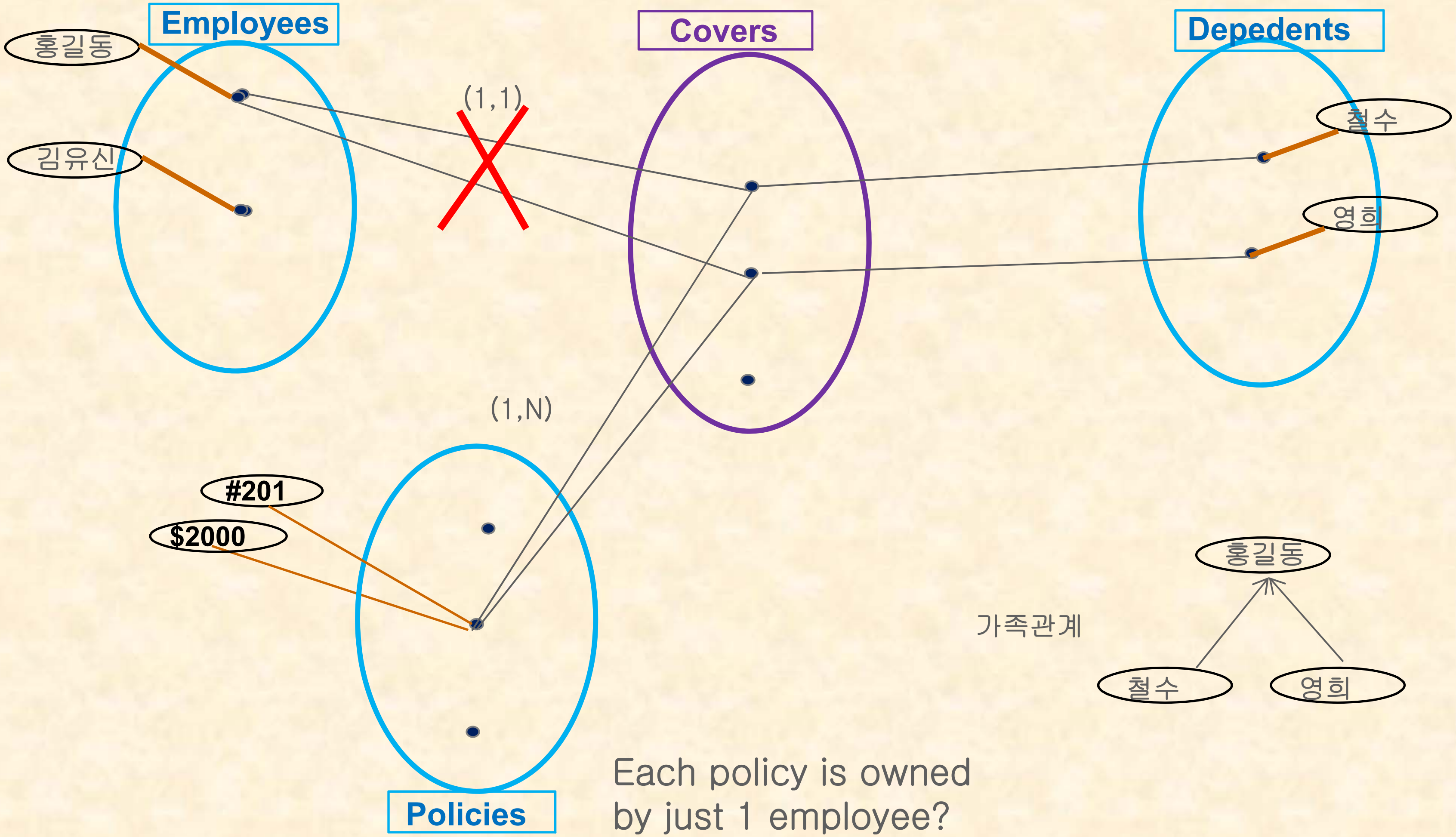
Binary vs. Ternary Relationships

- If each policy is owned by just 1 employee:
  - Key constraint on Policies would mean policy can only cover 1 dependent!



Bad design

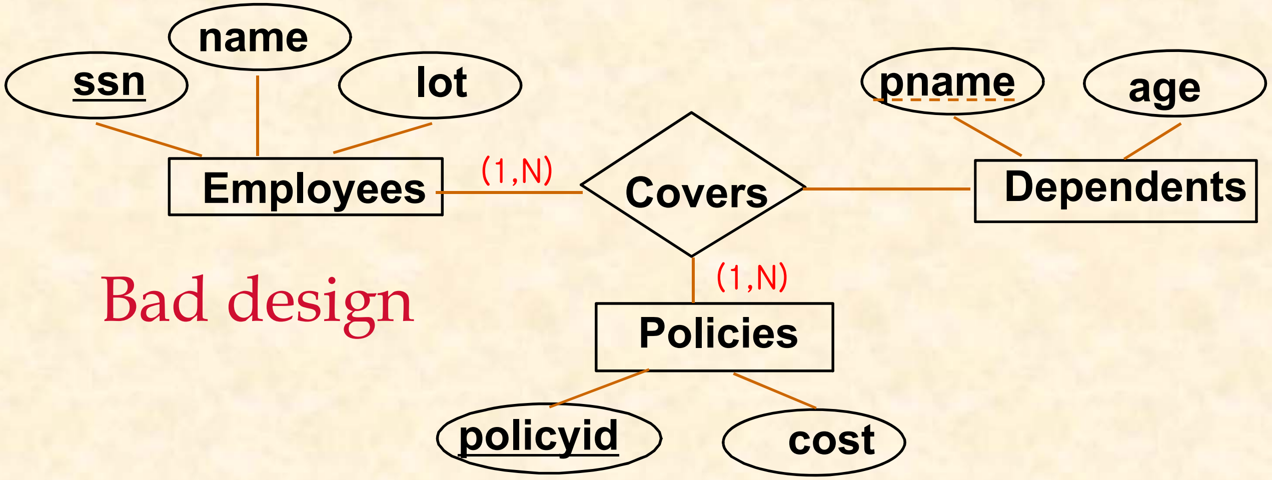
Entity vs. Attribute (Contd.)



Each policy is owned by just 1 employee?

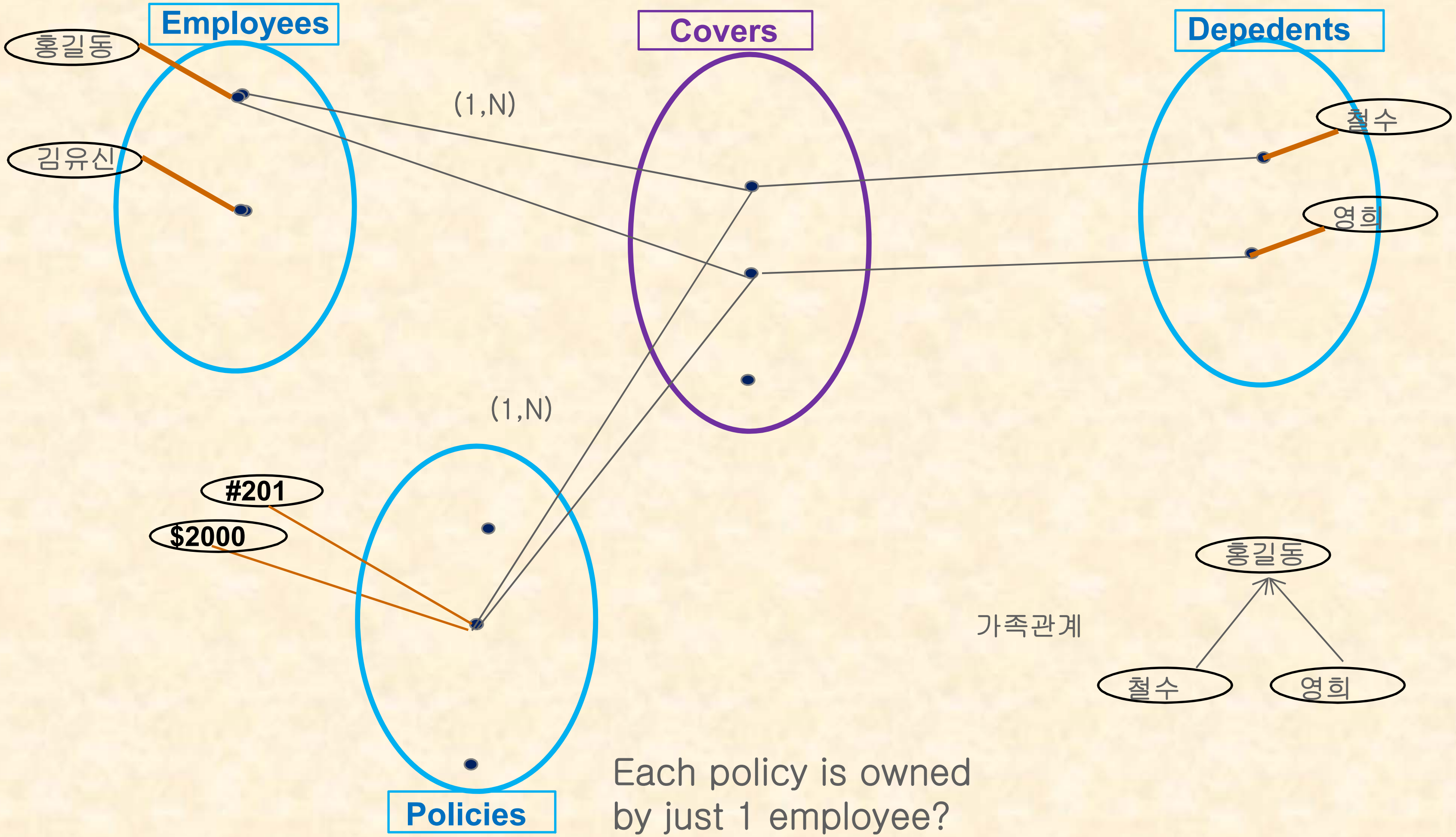
Binary vs. Ternary Relationships

- If each policy is owned by just 1 employee:
  - Key constraint on Policies would mean policy can only cover 1 dependent!



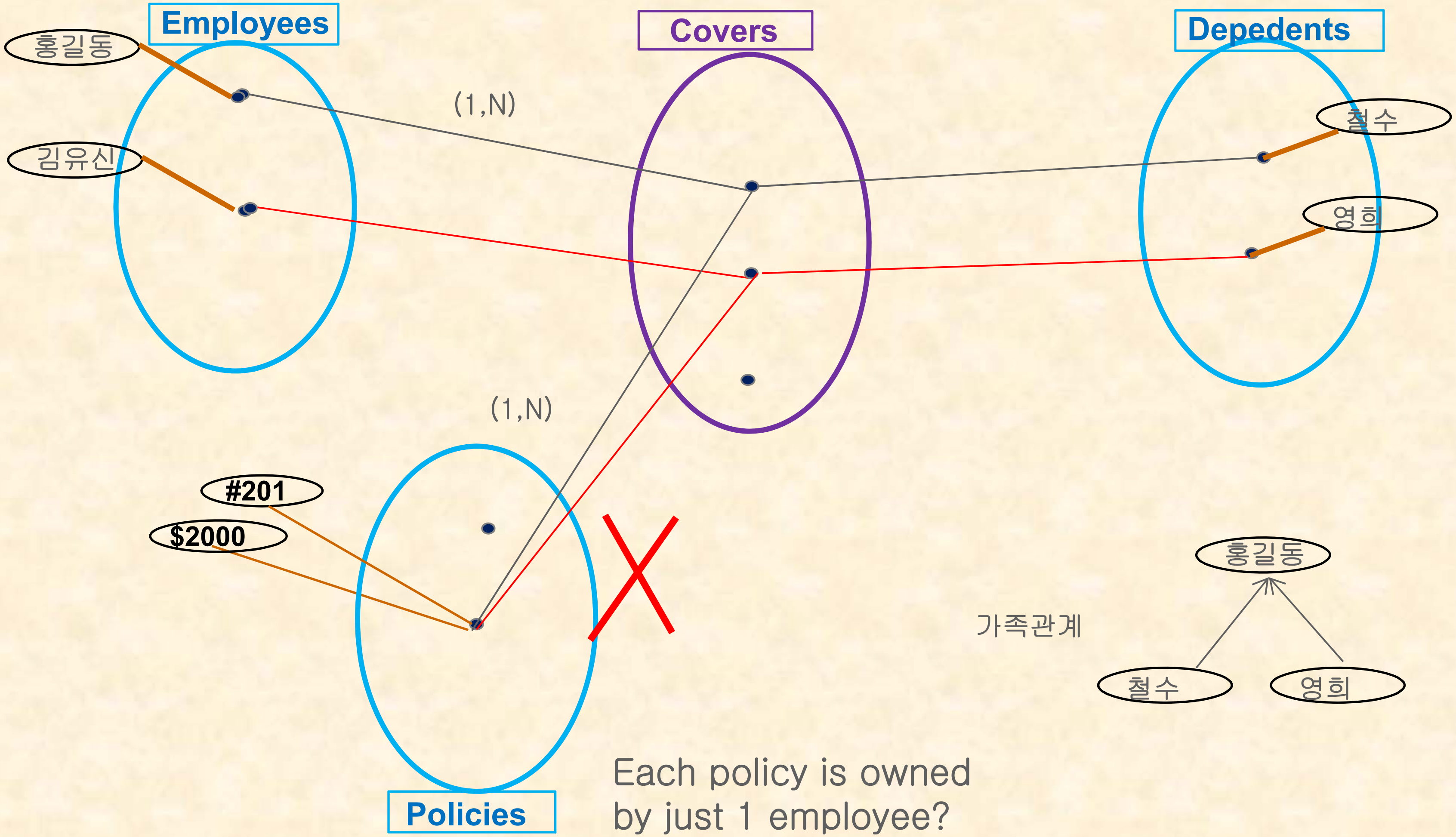
Bad design

Entity vs. Attribute (Contd.)



Each policy is owned by just 1 employee?

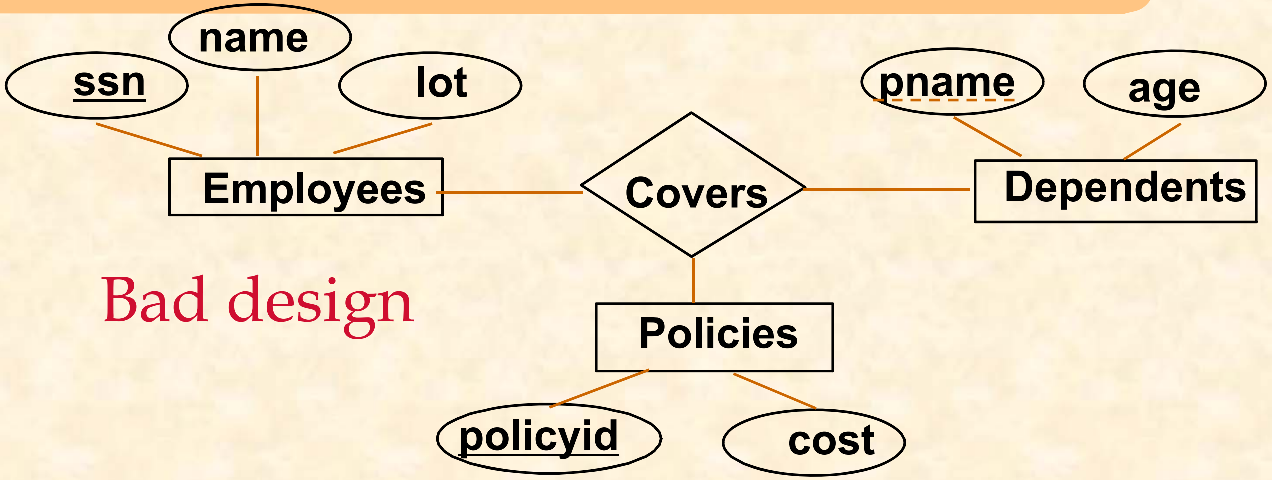
Entity vs. Attribute (Contd.)

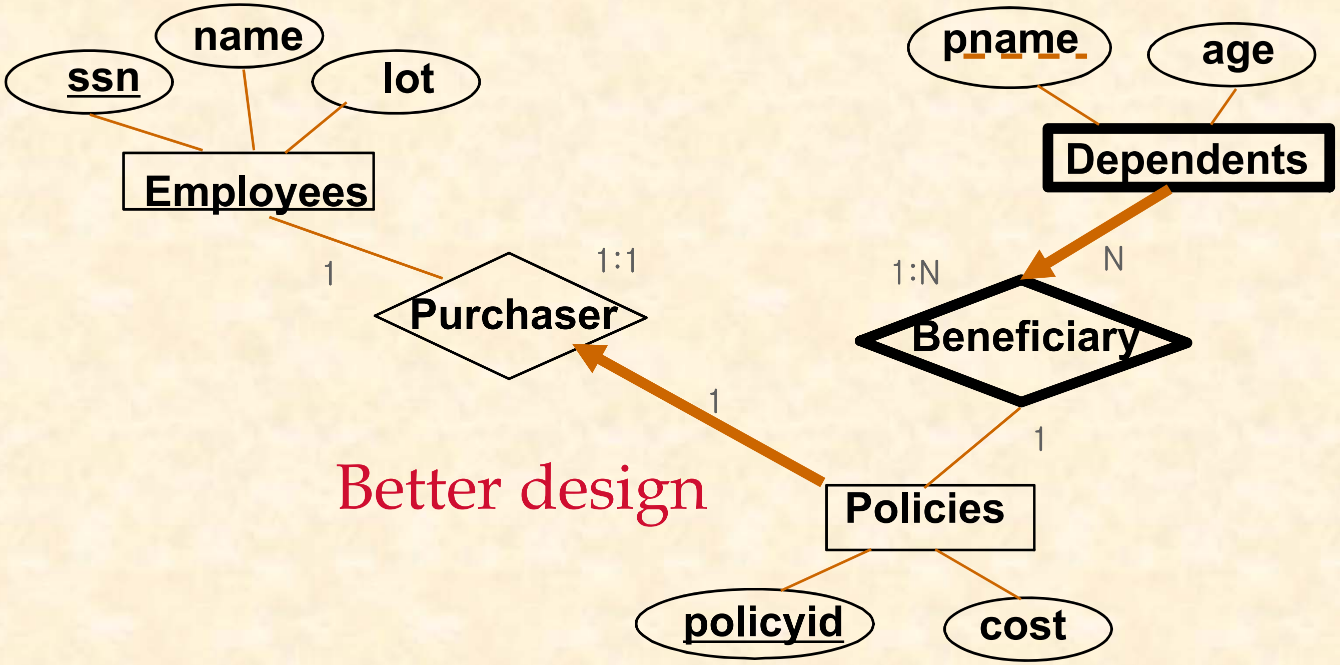


Each policy is owned by just 1 employee?

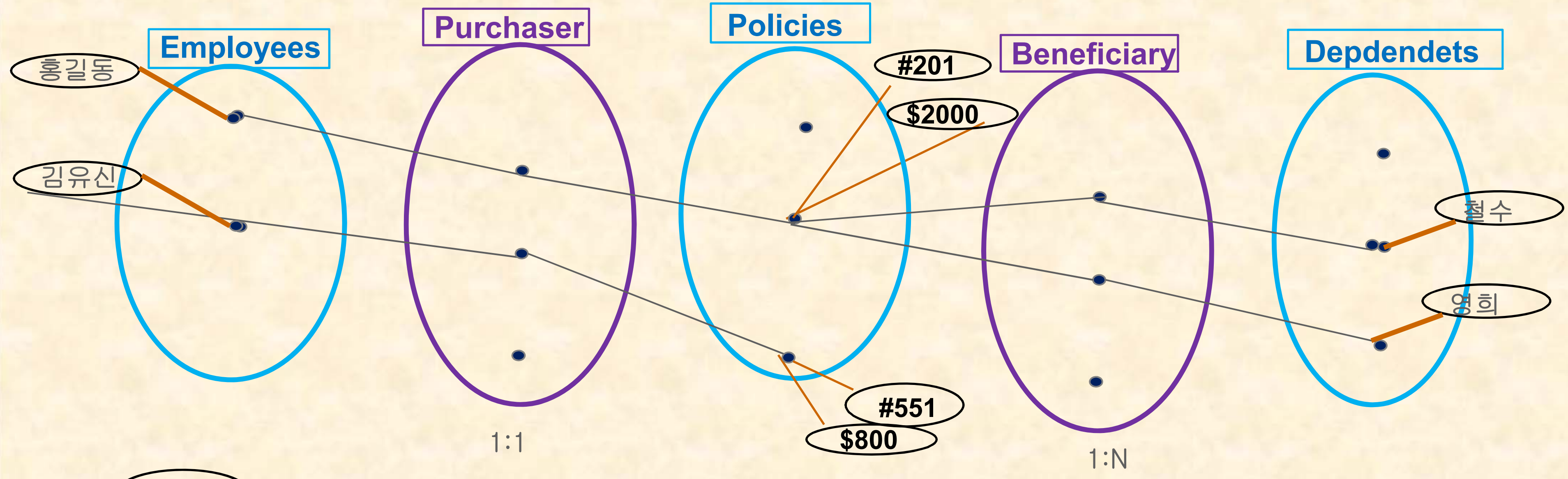
Binary vs. Ternary Relationships

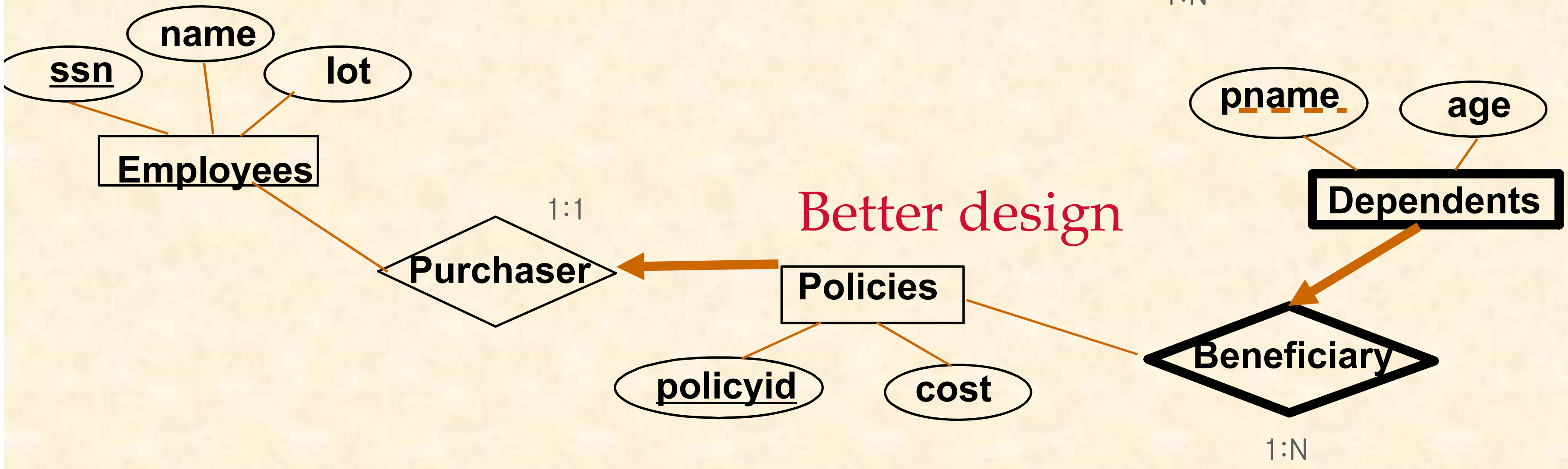
- If each policy is owned by just 1 employee:
  - Key constraint on Policies would mean policy can only cover 1 dependent!
- What are the additional constraints in the 2nd diagram?





Binary vs. Ternary Relationships





Better design

Binary vs. Ternary Relationships (Contd.)

- Previous example illustrated a case when two binary relationships were better than one ternary relationship.
- An example in the other direction: a ternary relation Contracts relates entity sets Parts, Departments and Suppliers, and has descriptive attribute *qty*. No combination of binary relationships is an adequate substitute:
  - S supplies P, D needs P, and D deals-with S does not imply that D has agreed to buy P from S.
  - How do we record *qty*?

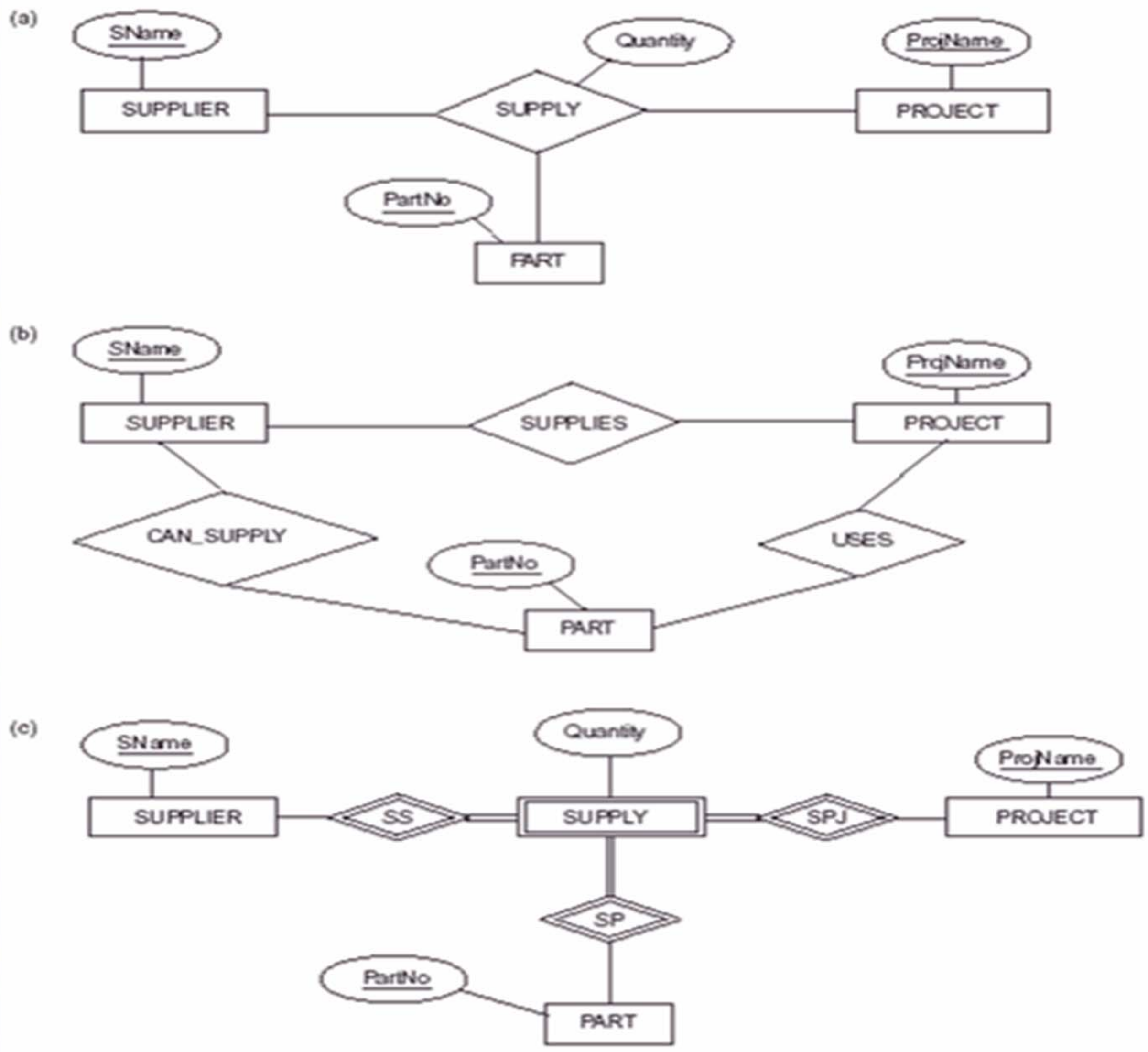

(a)
SName
Quantity
ProjName
SUPPLIER
SUPPLY
PROJECT
PartNo
PART
(b)
SName
ProjName
SUPPLIER
SUPPLIES
PROJECT
CAN_SUPPLY
USES
PartNo
PART
(c)
SName
Quantity
ProjName
SUPPLIER
SS
SUPPLY
SPJ
PROJECT
SP
PartNo
PART

Figure 4.14 Another example of ternary versus binary relationship types.

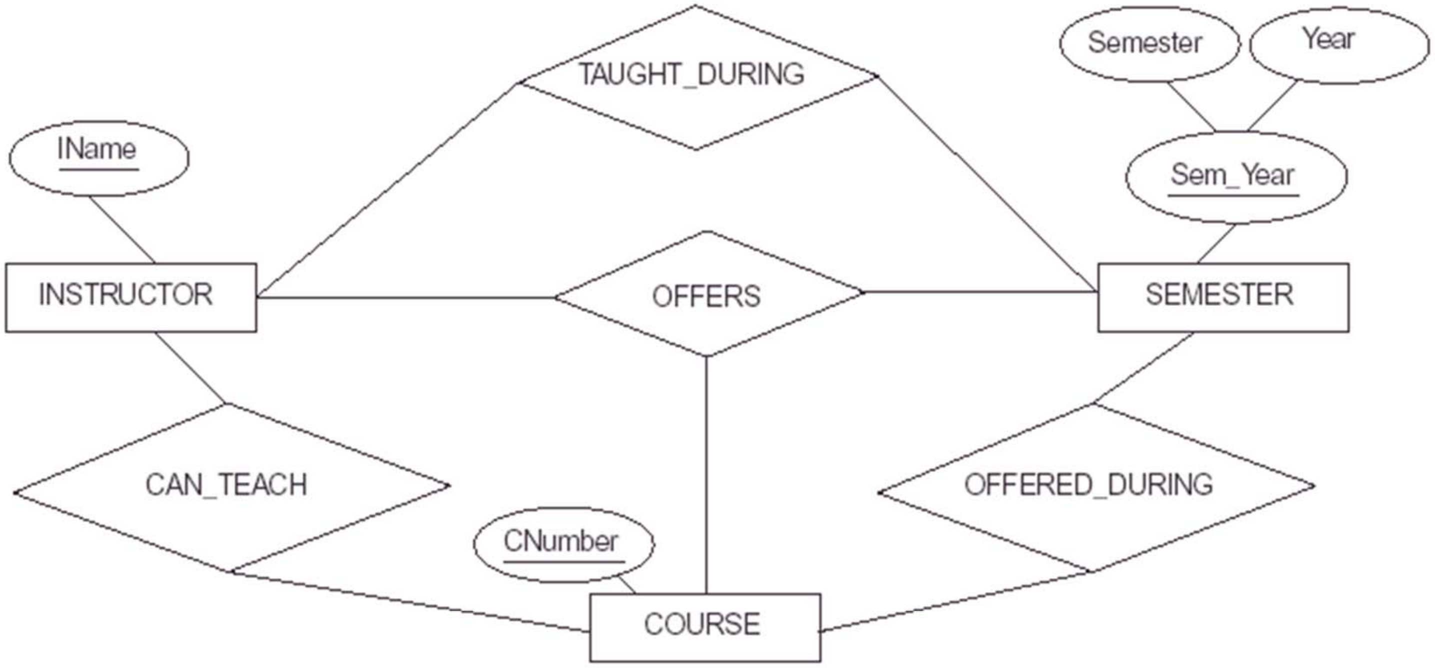


Figure 4.15 A weak entity type INTERVIEW, with a ternary identifying relationship type.

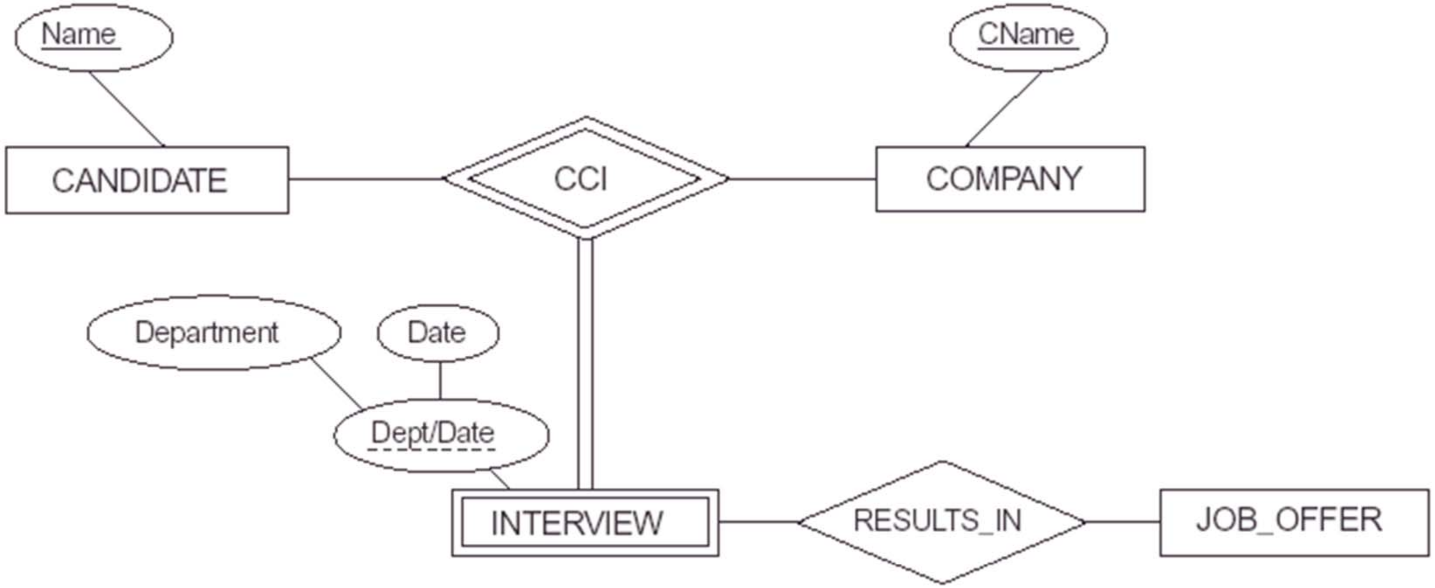
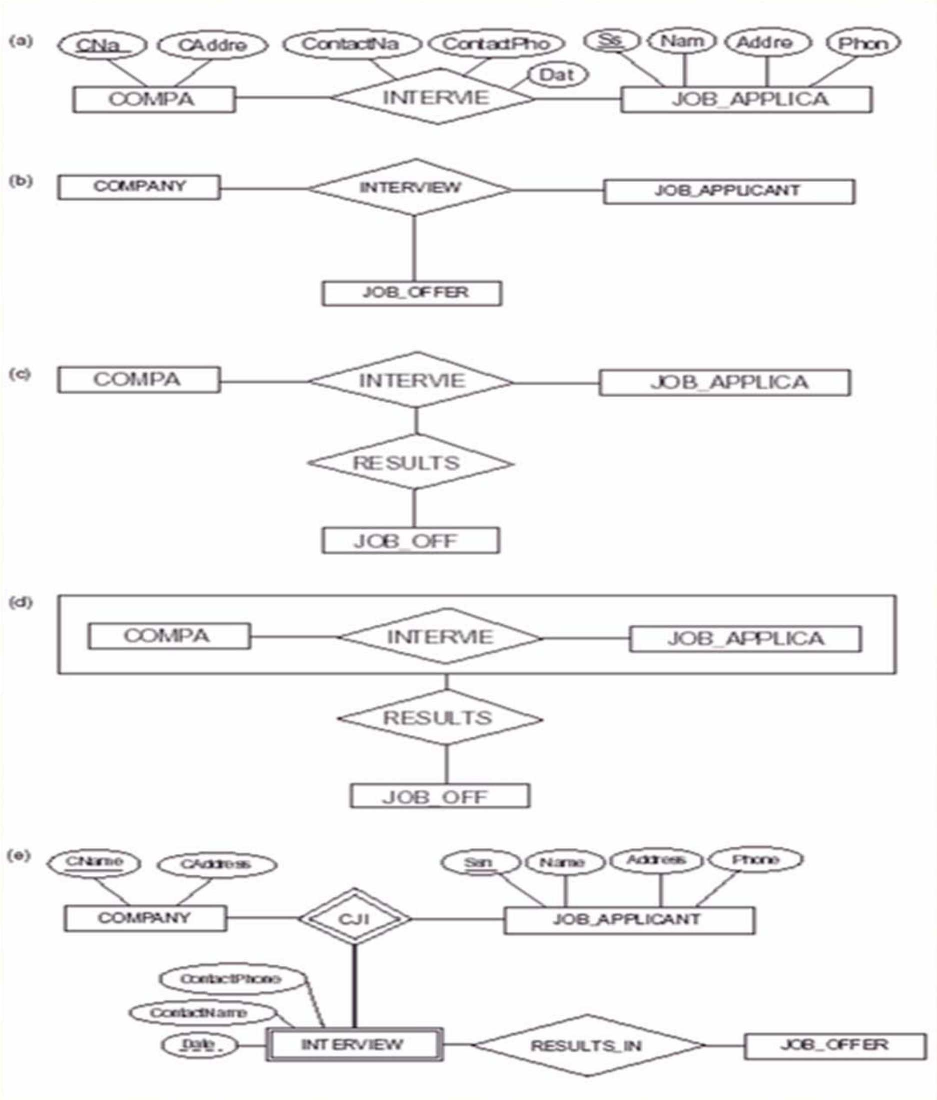


Figure 4.16 An illustration of aggregation. (a) The INTERVIEW relationship type. (b) Including JOB_OFFER in a ternary relationship type (incorrect). (c) Including JOB_OFFER by having a relationship in which another relationship participates (generally not allowed in ER). (d) Using aggregation and a composite (molecular) object (generally not allowed in ER). (e) Correct representation in ER.



Summary of Conceptual Design

- *Conceptual design* follows *requirements analysis*,
  - Yields a high-level description of data to be stored
- ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
- Basic constructs: *entities*, *relationships*, and *attributes* (of entities and relationships).
- Some additional constructs: *weak entities*, *ISA hierarchies*, and *aggregation*.
- Note: There are many variations on ER model.

Summary of ER (Contd.)

- Several kinds of integrity constraints can be expressed in the ER model: *key constraints*, *participation constraints*. Some *foreign key constraints* are also implicit in the definition of a relationship set.
  - Some constraints (notably, *functional dependencies*) cannot be expressed in the ER model.
  - Constraints play an important role in determining the best database design for an enterprise.

Summary of ER (Contd.)

- ER design is *subjective*. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship

- Ensuring good database design: resulting relational schema should be analyzed and refined further. FD information and normalization techniques are especially useful.