

Conceptual Data Modelling using Entity-Relationship Model

Contents

- 예제 데이터베이스 응용(COMPANY)
- ER 모델의 개념
 - 엔티티와 애트리뷰트
 - 엔티티 타입, 값 집합, 키 애트리뷰트
 - 관계와 관계 타입
 - 약한 엔티티 타입
 - 역할과 관계 타입에서의 애트리뷰트
- ER 다이어그램 - 표기법
- COMPANY 스키마를 위한 ER 다이어그램
- 다른 표기법 - UML 클래스 다이어그램, 기타

개체(Entity)

개체란 단독으로 존재하는 객체를 의미하며, 동일한 개체는 존재하지 않습니다.
예) 학생 정보가 학번, 이름, 학년이 있을 때, 3개의 정보가 같은 학생이 오직 한 명이면 이를 개체라고 함.
학생 한 명이 개체가 되는 것.

이러한 개체들의 집합 Entity type ⇒ 네로로 표현

애트리뷰트, 속성(attribute)

• 개체가 갖는 속성을 의미함. 원으로 표시
Student 테이블 (학번), (이름), (학년) 같은 정보를 속성이라 한다.

관계 (relation)

Entity type 간의 관계를 의미합니다.
수강을 뜻하는 Takes는 학생과 강의의 "수강"이라는 관계를 갖는다.

5. DEFAULT

해당 필드의 기본값을 설정할 수 있게 해준
레코드 입력 할 때 해당 필드 값을 전달하지 않으면
자동으로 설정된 기본 값을 저장한다.

1. CASCADE

참조되는 테이블에서 데이터를 삽입하거나
수정하면 참조하는 테이블에서도 삽입과
수정이 같이 이루어짐

2. SET NULL

- 참조되는 테이블에서 데이터를 삭제하거나
수정하면 참조하는 테이블의 데이터는
NULL로 변경됨

3. NO ACTION

- " 참조하는 테이블의
데이터는 변경이 없음."

4. SET DEFAULT

- " 참조하는 테이블의 데이터는
필드의 기본값으로 설정"

5. RESTRICT

- 참조하는 테이블의 데이터가 남아있으면 참조되는 테이블에 대한 수정X

Phases of DB Design

제약 조건

1. NOT NULL 해당 필드는 NULL 값을 저장할 수 없다.
이 제약 조건에 설정된 필드는 무조건 데이터를
가지고 있어야 함.

2. UNIQUE 해당 필드는 서로 다른 값을 가짐야 함.
이 제약 조건에 설정된 필드는 중복된 값을 저장할 수
없다.

3. PRIMARY KEY 해당 필드는 Not Null 과 Unique 제약 조건의
특성을 모두 가진다.
= 기본키
이 제약 조건이 설정된 필드는 Null값 X, 중복값 X

4. FOREIGN KEY 외래키, 한 테이블을 다른 테이블과 연결해주는 역할
외래가 설정된 테이블에 레코드를 입력하면, 그에 따른 테이블의 내용을
참조해서 레코드가 입력됨.
즉, Foreign Key 제약 조건은 해외의 테이블을 다른 테이블에 의존하게
만든다.

FOREIGN KEY를 설정할 때, 참조되는 테이블의 필드는 반드시
UNIQUE OR PRIMARY KEY가 설정되어 있어야 한다.

CREATE TABLE Test2

```
(  ID INT,  
  parent_ID INT,  
  FOREIGN KEY (parent_ID)  
  REFERENCES Test1 (ID) ON UPDATE CASCADE );
```

1. ON DELETE

- 참조되는 테이블의 값이 삭제될 경우의 동작을 ON DELETE 구문 설정

2. ON UPDATE

- 참조되는 테이블의 값이 수정될 경우의 동작을 ON UPDATE 구문 설정

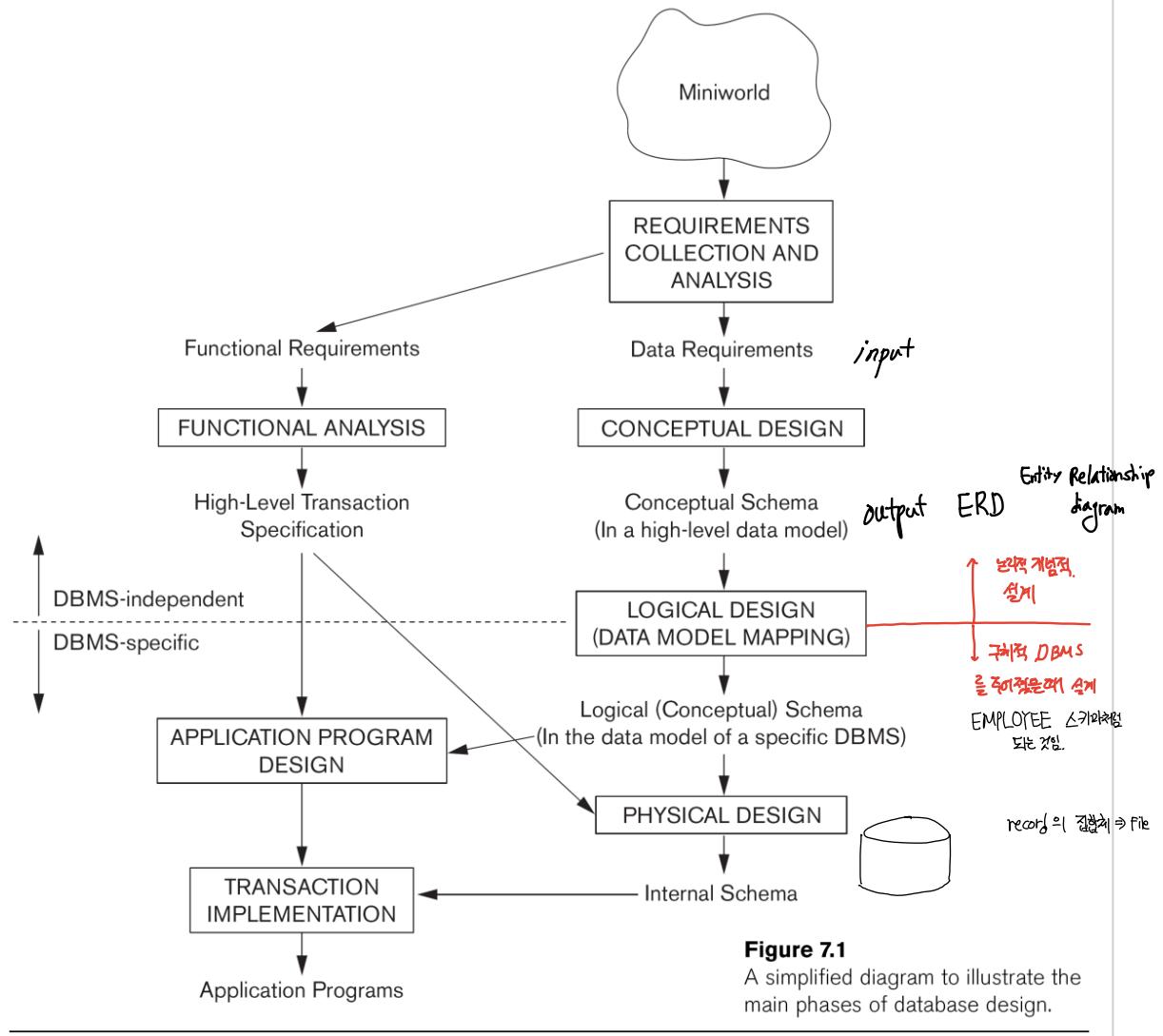


Figure 7.1
A simplified diagram to illustrate the main phases of database design.

Data Requirements: COMPANY DB

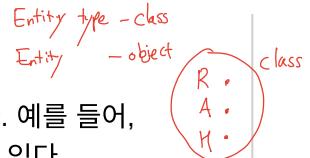
- 회사는 여러 부서(DEPARTMENT)들로 구성. 각 부서는 부서명(name), 번호(number), 부서장을 가진다. 부서장의 부임 날짜(start date)도 유지한다. **부서를 class로 본다면 부서명, 번호, 부서장이 attribute가 된다.**
- 한 부서는 여러 개의 프로젝트(PROJECT)들을 관리한다. 각 프로젝트는 이름(name)과 번호(number)를 가지며 한 곳(location)에 위치한다. **ER 모델에서는 entity type이라고 함.** 설명하는 변수들은 attribute라고 함.
- 각 사원(EMPLOYEE)의 주민등록번호(social security number), 주소(address), 월급(salary), 성별(sex), 생년월일(birthdate)을 저장한다.
- 각 사원은 한 부서에서 근무하며(works for) 여러 프로젝트에 관여한다(work on) .
- 각 사원이 각 프로젝트를 위해 주당 근무 시간을 저장한다.
- 또한, 각 직원의 직속 상사(direct supervisor)도 유지한다. 각 사원은 여러 명의 부양가족(DEPENDENT)들을 가진다. **단수, 복수를 구분하는 것도 중요. 한 사원은 여러명의 부양가족을 가질 수 있다.**
- 각 부양가족에 대해 이름(name), 성별(sex), 생년월일(birthdate), 직원과의 관계(relationship)를 저장한다.

The Concept of Entity Relationship Model

Entity and Attribute

모델링하려는 대상

- 엔티티는 데이터베이스 내에 표현된 작은 세계에 존재하는 객체 또는 실체이다. 예를 들어, EMPLOYEE John Smith, 연구 DEPARTMENT, ProductX PROJECT 등이 있다.
- 애트리뷰트들은 엔티티를 기술하기 위한 속성들이다. 예를 들어, EMPLOYEE 엔티티는 Name, SSN, Address, Sex, BirthDate를 가질 수 있다.
- 특정한 엔티티는 자신의 각 애트리뷰트에 대해 값을 가진다. 예를 들어, 한 특정한 사원 엔티티는 Name='John Smith', SSN='123456789', Address='731, Fondren, Houston, TX', Sex='M', BirthDate='09-JAN-55'를 값으로 가진다.
- 각 애트리뷰트는 정수, 스트링과 같은 자신에 연계된 값 집합(또는 데이터 타입)을 가진다.



Attribute Type

- 단순(원자) 애트리뷰트 (Simple, Atomic, Scalar)
 - 각 엔티티는 각 애트리뷰트에 대해 더 이상 나눌 수 없는 값을 가진다. 예를 들면, SSN 또는 Sex.
- 복합(Composite) 애트리뷰트
 - 애트리뷰트는 몇 개의 구성요소들로 이루어 질 수 있다. 예를 들면, Address(Apt#, House#, Street, City, State, ZipCode, Country) 또는 Name(FirstName, MiddleName, LastName). 구성은 그 구성요소가 다시 복합 애트리뷰트인 계층 구조를 형성할 수 있다.
- 다치(Multi-valued) 애트리뷰트
 - 각 엔티티는 애트리뷰트의 값으로 여러 값을 가질 수 있다. 예를 들면, {Color}로 표기되는 CAR의 색 또는 {PreviousDegrees}로 표기되는 STUDENT의 이전 학위가 다치 애트리뷰트이다.
- 일반적으로, 복합 및 다치 애트리뷰트는 몇 단계로 내포될 수 있지만 그런 경우는 흔하지 않다.
 - 예를 들면, STUDENT의 PreviousDegrees는 {PreviousDegrees (College, Year, Degree, Field)}로 표기되는 다치 애트리뷰트일 수 있다.

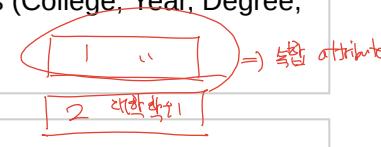


성과 이름을 냄는 것

나는 성을 고으한 값으로
보았다는 것

주제는 없는데 명의 연관으로 놓았지만,

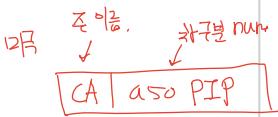
다치 attribute 예제??



(3 대학학위)

Entity Type and Key Attribute

- 엔티티 타입은 동일한 애트리뷰트들을 갖는 엔티티들의 집합으로 정의함
 - 예를 들면, EMPLOYEE 엔티티 타입 또는 PROJECT 엔티티 타입.
- 키 애트리뷰트
 - 한 엔티티 타입에서 각 엔티티가 유일한 값을 가지는 애트리뷰트를 그 엔티티 타입의 키 애트리뷰트라 한다. 예를 들면, EMPLOYEE의 SSN.
 - 키 애트리뷰트는 복합형일 수 있다. 예를 들어, VehicleTagNumber는 구성요소로 (Number, State)를 가지는 CAR 엔티티 타입의 키이다.
 - 엔티티 타입은 한 개 이상의 키를 가질 수 있다. 예를 들어, CAR 엔티티 타입은 다음의 두 키를 가진다.
 - VehicleIdentificationNumber (popularly called VIN)
 - VehicleTagNumber (Number, State), 또는 license_plate number.



복합 attribute

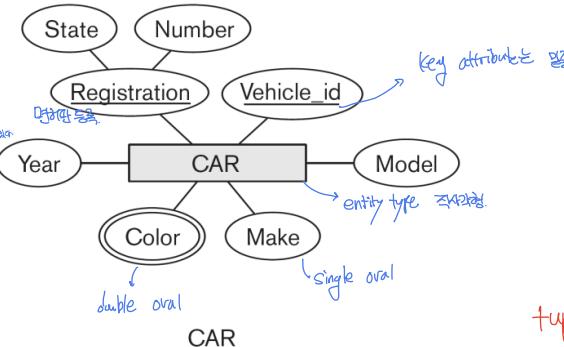
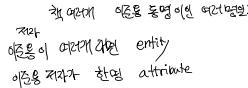


key
키
ck
회원키

Entity Type : CAR

Figure 7.7

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



(b) Key 는 책의 ISBN

책에서 출판사와 저자는

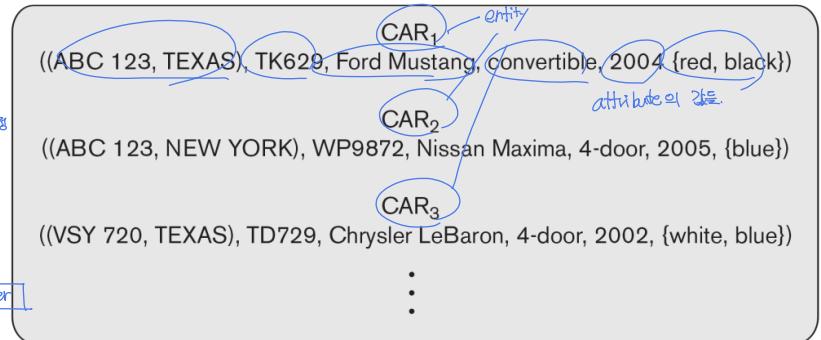
각각의 attribute

모델링하는데 도시문이 어떤
값으로 설정할지 entity로 설정
할지 중요.

(— — — — , 이준용)

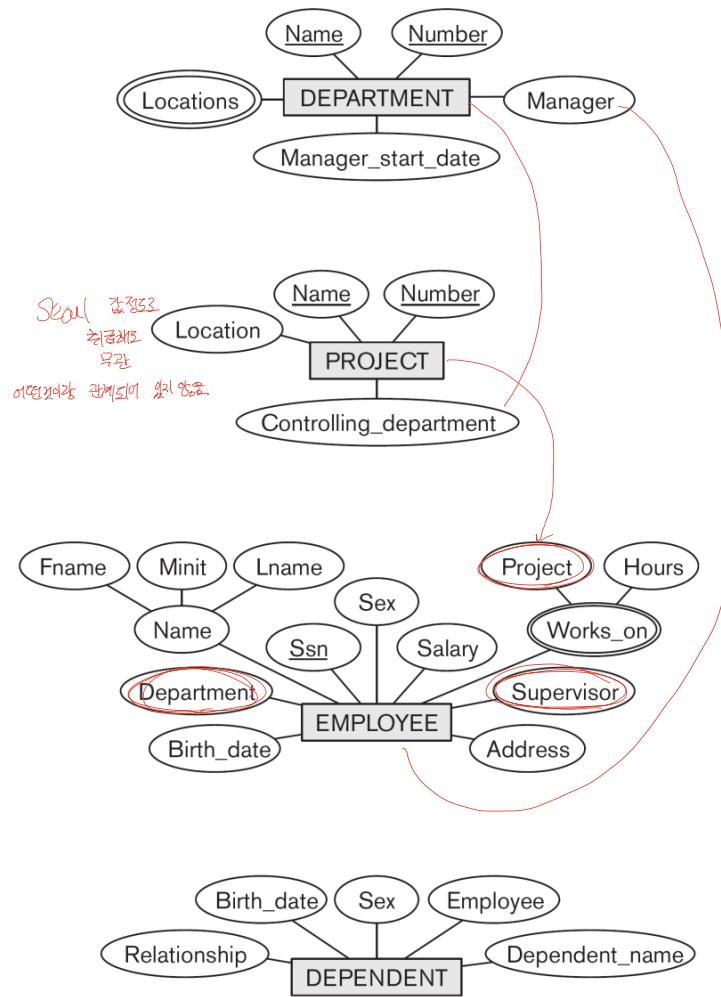


Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}



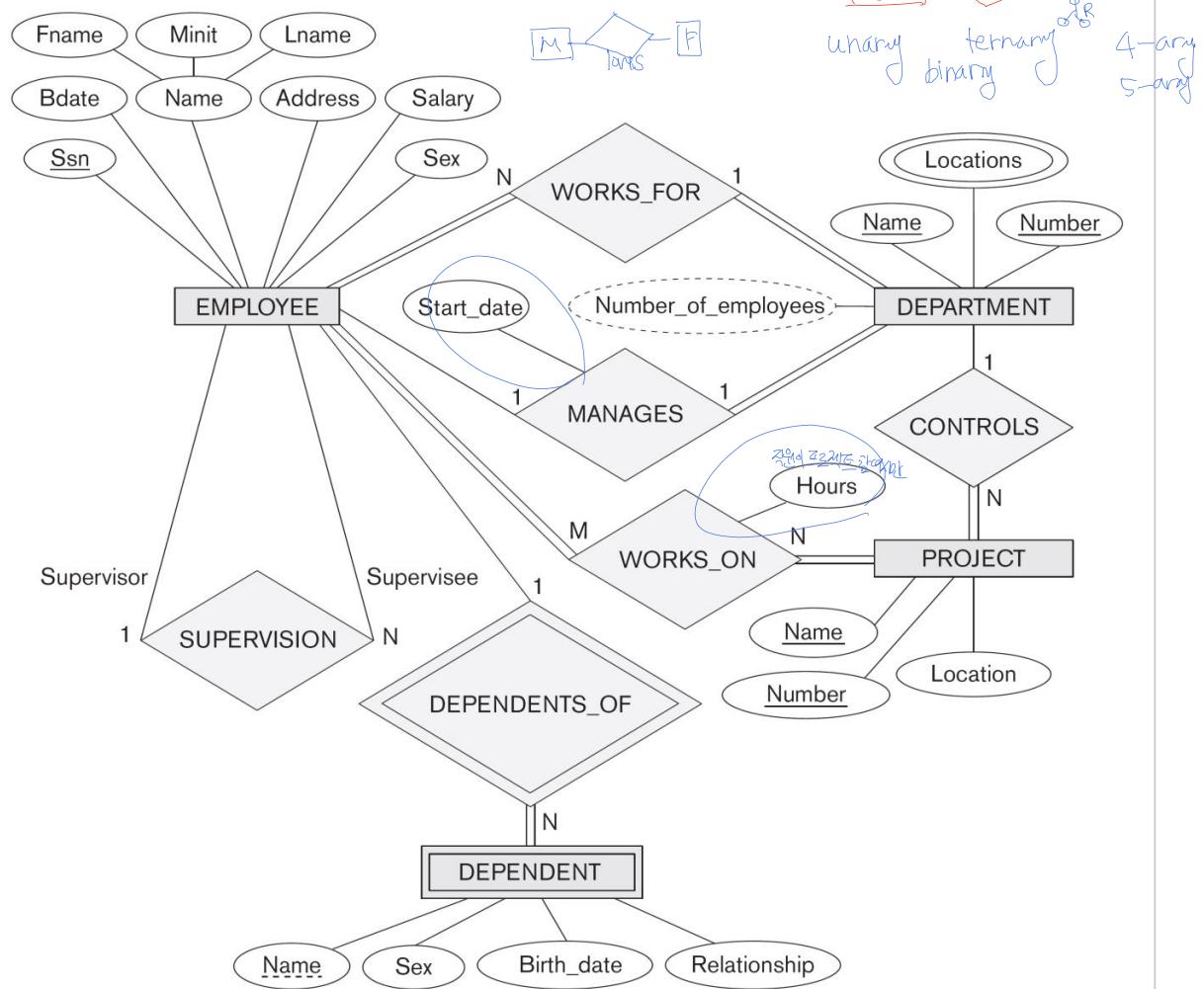
⁵We use a notation for ER diagrams that is close to the original proposed notation (Chen 1976). Many other notations are in use; we illustrate some of them later in this chapter when we present UML class diagrams and in Appendix A.

Preliminary Design for Company

**Figure 7.8**

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

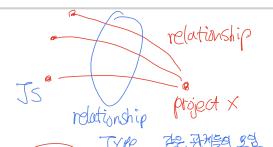
ER Diagram Entity Type: EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

**Figure 7.2**

An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 7.14.

Relationship and Relationship Type

- 관계는 두 개 또는 그 이상의 엔티티들을 특정한 의미로 연관 짓는 것임
 - 예를 들어, EMPLOYEE John Smith는 ProductX PROJECT를 위해 일하며, EMPLOYEE Franklin Wong은 Research DEPARTMENT를 관리한다.
- 같은 형의 관계들은 관계 타입으로 그룹화되어 형이 주어짐
 - 예를 들면, EMPLOYEE들과 PROJECT들이 참여하는 WORKS_ON 관계 타입, EMPLOYEE들과 DEPARTMENT들이 참여하는 MANAGES 관계 타입.
- 관계 타입의 차수는 참여하는 엔티티 타입의 개수임
 - MANAGES와 WORKS_ON은 모두 이진 관계이다.
- 두 엔티티 타입들에 대해 한 개 이상의 관계 타입이 있을 수 있다.
 - 예를 들면, EMPLOYEE와 DEPARTMENT 간의 MANAGES와 WORKS_FOR는 서로 다른 의미와 관계 인스턴스들을 가지는 두 관계들이다.



WORKS_FOR Relationship between EMPLOYEE and DEPARTMENT

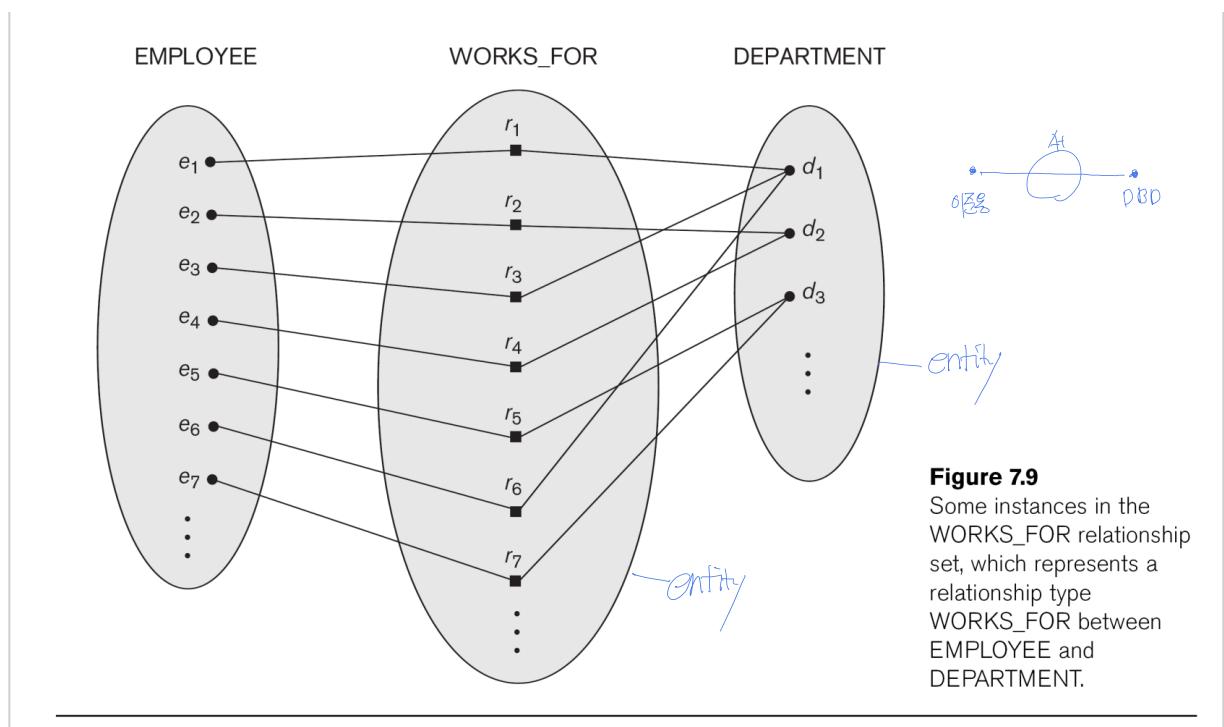
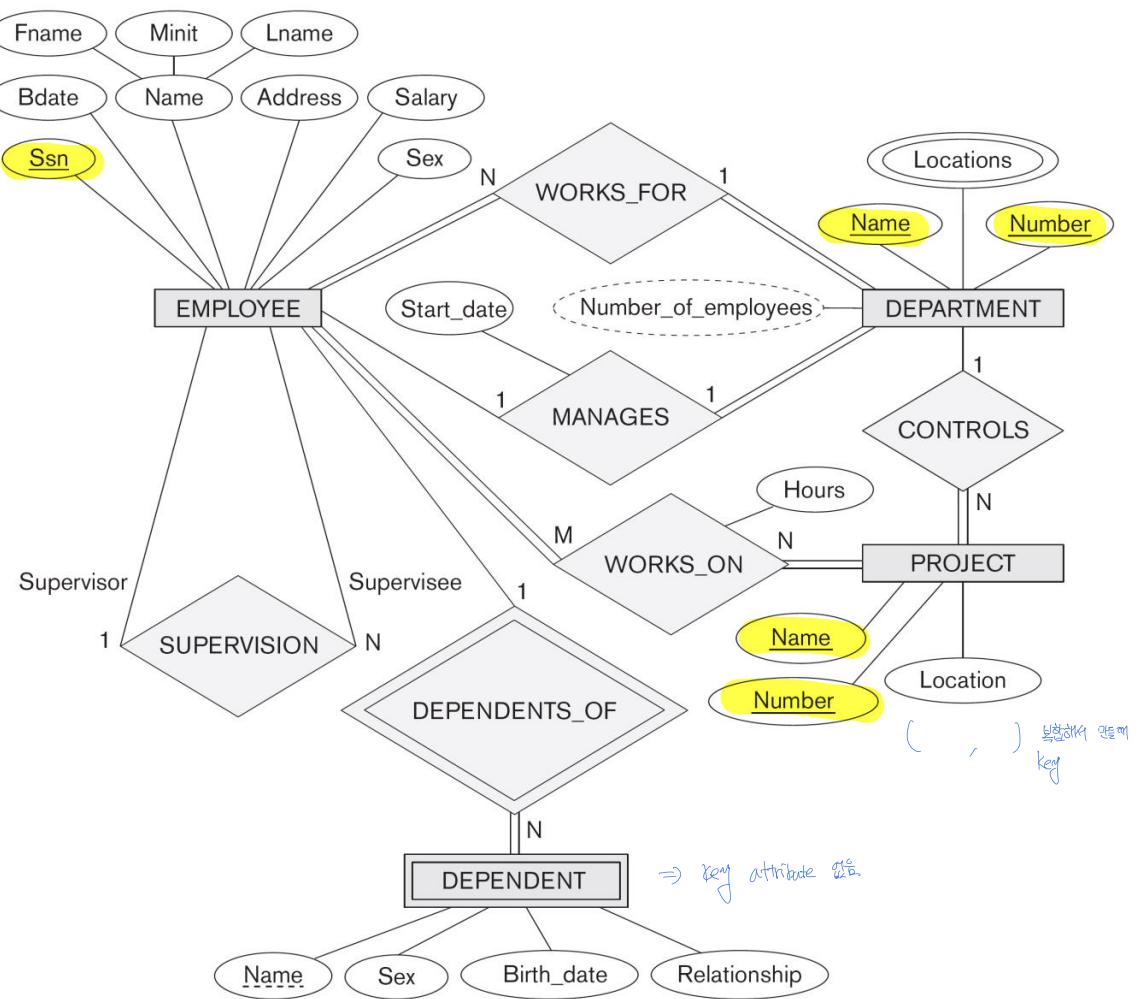


Figure 7.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

ER Diagram Relationship Type: WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF

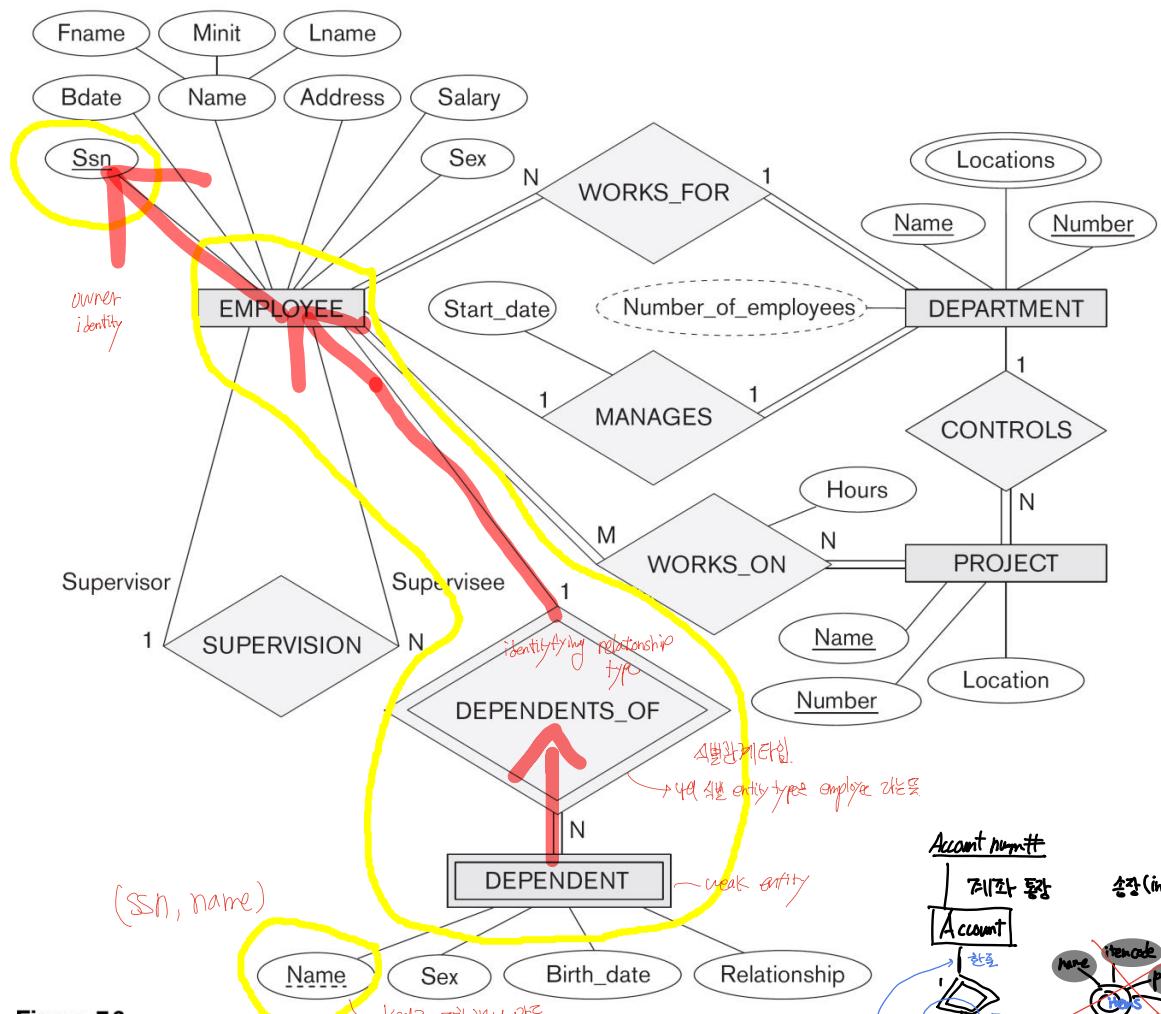
**Figure 7.2**

An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 7.14.

Weak Entity Type

- 키 애트리뷰트가 없는 엔티티
- 약한 엔티티는 소유 또는 식별 엔티티 타입과의 식별 관계 타입에 참여해야 함
- 엔티티들은 다음의 조합에 의해 식별됨
 - 약한 엔티티 타입의 부분 키
 - 식별 엔티티 타입과 연관된 특정 엔티티
 - 예: DEPENDENT 엔티티는 부양가족의 이름과 생년월일 및 그 부양가족과 연관되는 사원에 의해 식별된다고 가정하자. DEPENDENT는 EMPLOYEE를 식별 엔티티 타입으로 하고 DEPENDENTS_OF의 식별 관계 타입으로 연관되는 약한 엔티티 타입이다.

Weak Entity Type DEPENDENT and IdentifyingRelationship DEPENDENTS_OF

**Figure 7.2**

An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 7.14.

Constraints on Relationship

- 관계 탑입에 대한 제약조건:** 카디널리티 비율 제약조건과 참여 제약조건 두 가지가 있음

- 카디널리티 비율 제약조건 *Cardinality = length.size*

- 최대 카디널리티**

- 1:1, 1:N 또는 N:1, M:N

- 최소 카디널리티(참여 제약조건 또는 존재 종속 제약조건)**

- 0 (선택적 참여, 존재 종속이 아님) \Rightarrow 어떤 부속은 참여하고 있지 않고 있는 뜻
- 1 또는 그 이상 (의무적, 존재 종속) *mandatory*

- 참여제약 조건

- 관계에 참여하는 엔티티 탑입의 일부 엔티티만 관계에 참여함

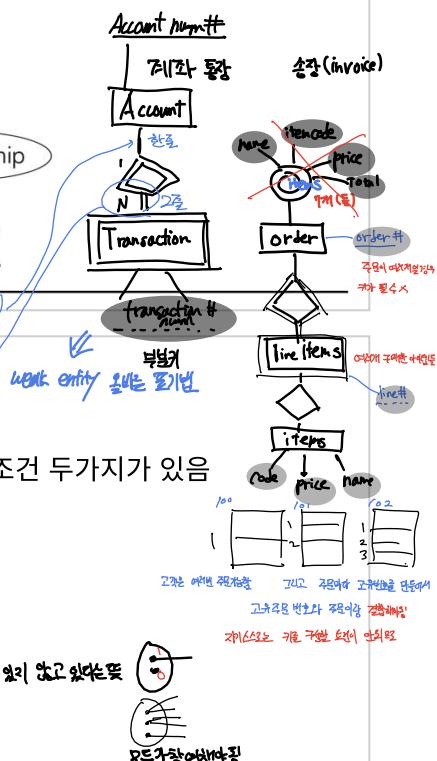
- 예를 들어, Employee 중에서 일부만 Department와의 manager 관계에 참여함

- 전체 (또는 존재 종속성(existential dependence), total, mandatory) 또는 부분(partial, optional) 참여

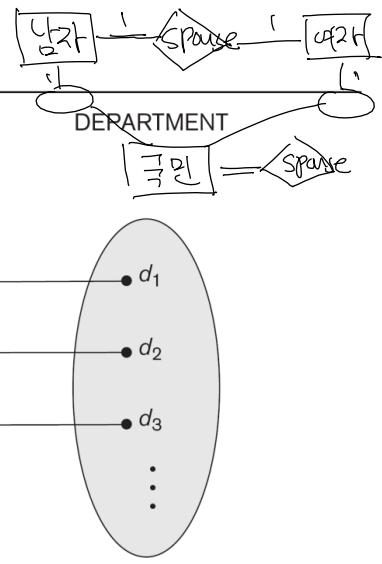
- 전체 참여의 경우 연결된 선을 이중선으로 표기함. EM

최대 카디널리티 \Rightarrow max card=1

최소 카디널리티 \Rightarrow min card=1



One-to-one (1:1) Relationship

**Figure 7.12**

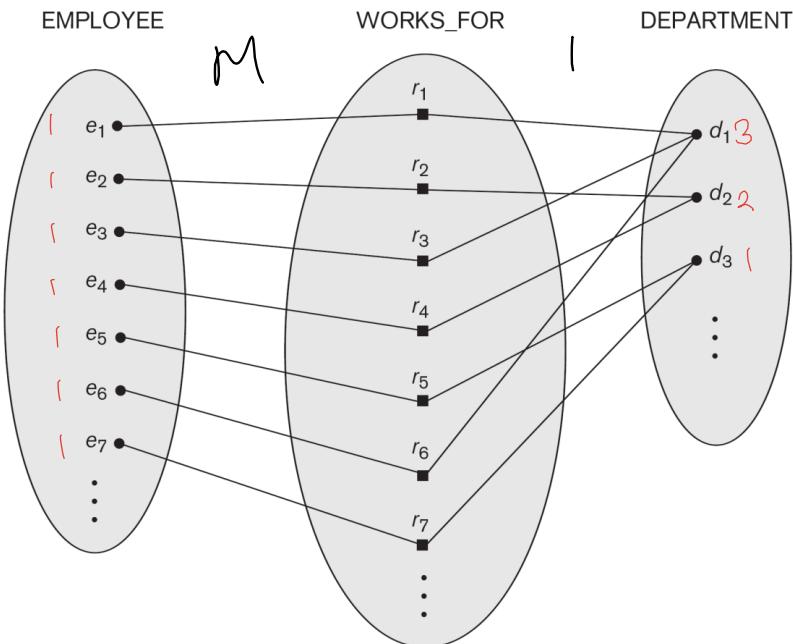
A 1:1 relationship,
MANAGES.

주인공은 최대 1개만 관리할 수가 있다 (o)
보이는 크기나 같아야 함.

⁹N stands for any number of related entities (zero or more).

One-to-many (1:N) Relationship

1:1 M:N (o)
1: Many M:M (x)

**Figure 7.9**

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

Many-to-many (M:N) Relationship

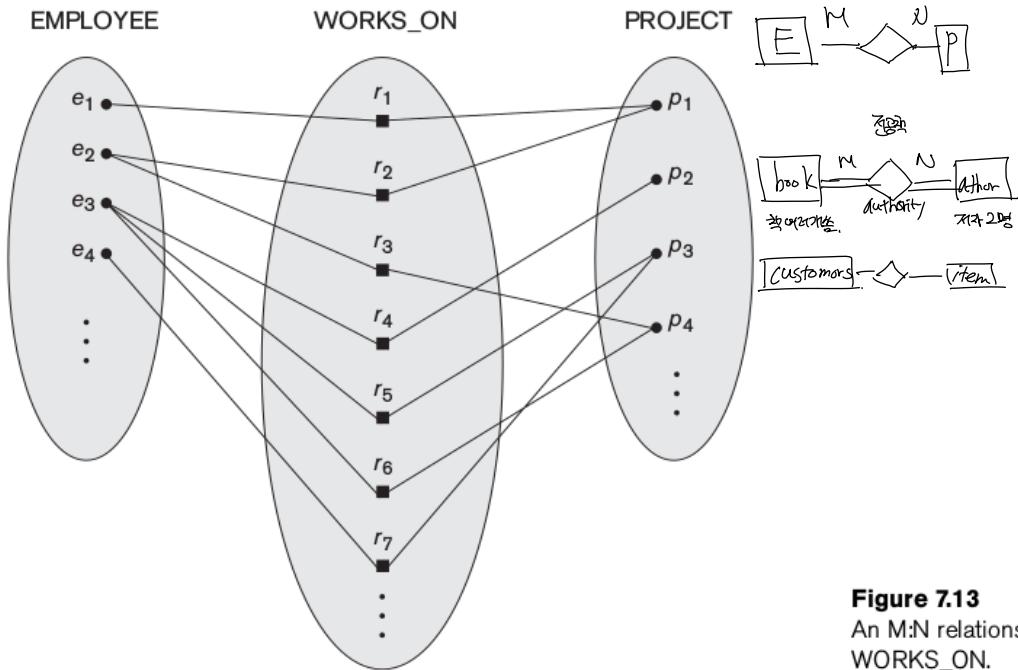
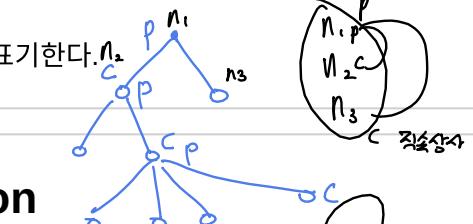
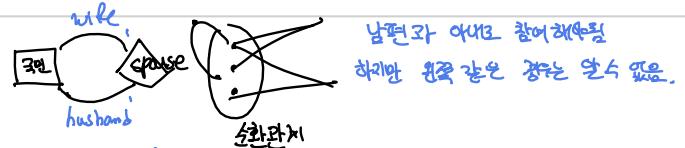


Figure 7.13
An M:N relationship,
WORKS_ON.

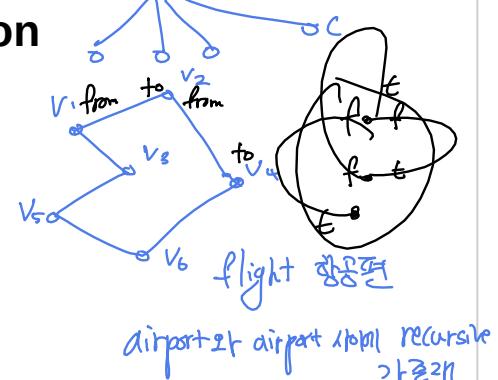
모든 사람이 결혼해야 되는가 아니고 한 줄

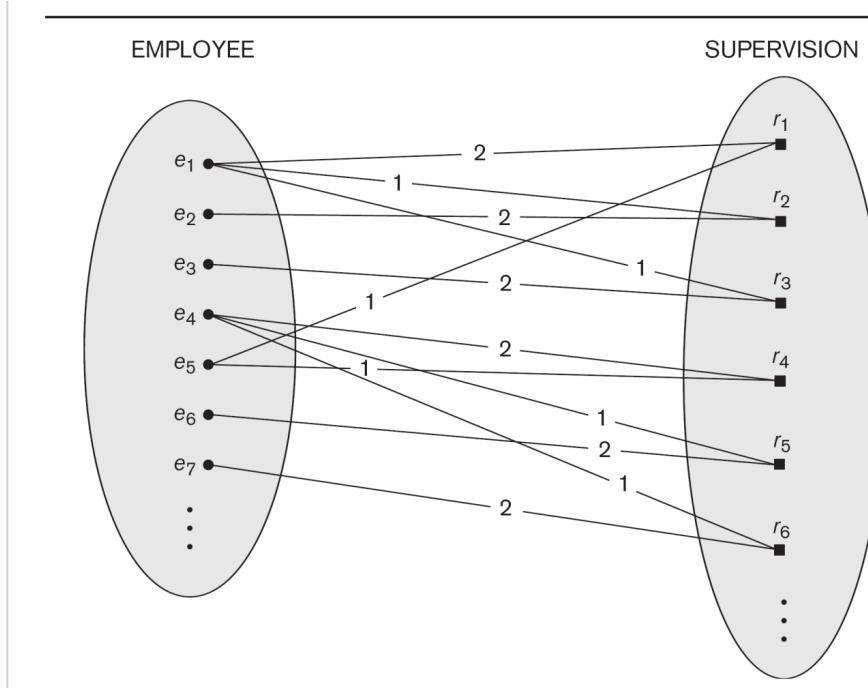
Recursive Relationship

- 순환 관계
- 한 개의 엔티티 타입이 어떤 관계 타입에 서로 다른 역할로 중복 참여하는 경우를 의미함
- 예를 들어, EMPLOYEE (감독자 또는 상사의 역할로)와 EMPLOYEE (부하 또는 근로자의 역할로) 간의 SUPERVISION 관계
- 다음 page 그림에서 1이라고 표기된 첫 번째 역할(감독자 또는 상사의 역할) 참여와 2라고 표기된 두 번째 역할 (부하 또는 근로자의 역할) 참여.
- ER 다이어그램에서는 참여를 구분하기 위해 역할 이름을 표기한다.



Recursive Relationship: Supervision



**Figure 7.11**

A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

Attributes of Relationship Type

- 관계 타입도 애트리뷰트들을 가질 수 있다.
- 예를 들면, 관계 타입 WORKS_ON의 Hours 속성의 값은 각 관계 인스턴스에 대해 EMPLOYEE 가 PROJECT를 위해 주당 일한 시간을 나타낸다.

Min-Max Notation on Constraints

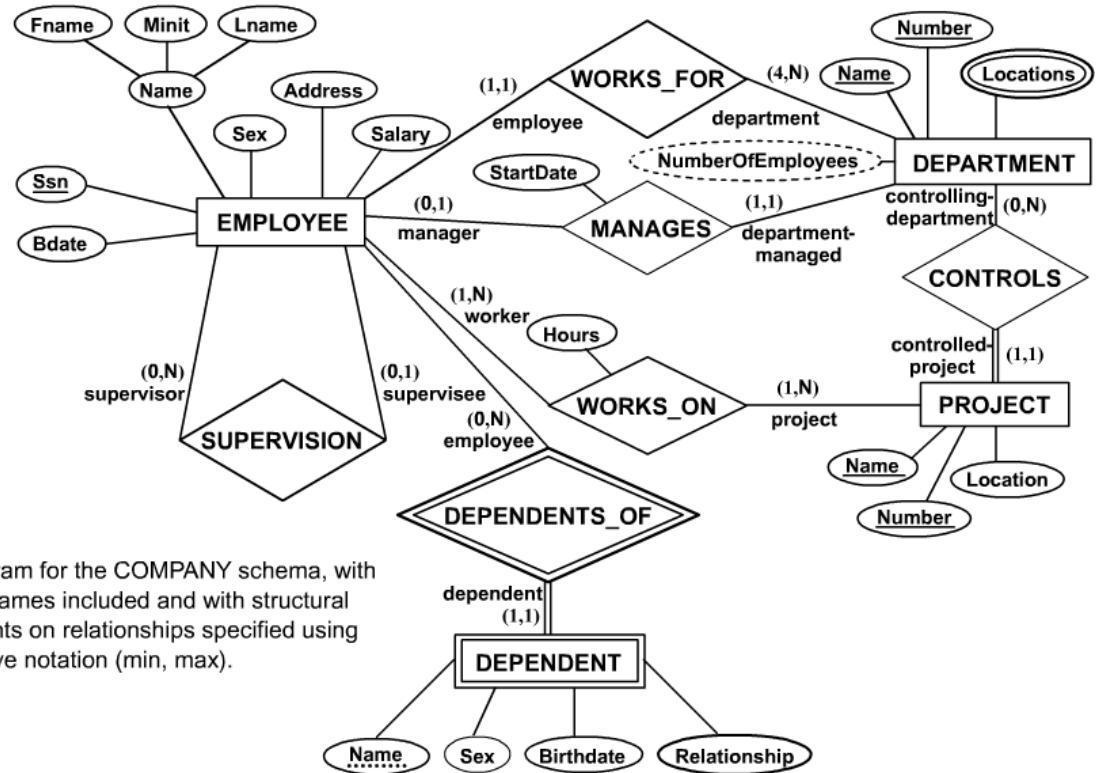
- 관계 타입 R에서 엔티티 타입 E의 각 참여를 명시
- E 내의 각 엔티티 e가 적어도 최소값 만큼 그리고 최대 최대값만큼 R 내의 관계 인스턴스들과 참여 함을 나타냄
- 디폴트(제약조건 무): 최소값=0, 최대값=n
- 반드시, 최소값 ≤ 최대값, 최소값 ≥ 0, 최대값 ≥ 1
- 작은-세계 제약조건으로 부터 유도됨

예:

- 부서는 정확히 한 명의 부서장을 가지며 사원은 최대 한 개의 부서를 관리할 수 있다.
 - MANAGES의 EMPLOYEE의 참여에 (0,1)로 표기
 - MANAGES의 DEPARTMENT의 참여에 (1,1)로 표기
- 사원은 정확히 한 부서에서 일 할 수 있으며 부서는 몇 명의 사원이라도 둘 수 있다.
 - WORKS_FOR의 EMPLOYEE의 참여에 (1,1)로 표기
 - WORKS_FOR의 DEPARTMENT의 참여에 (0,n)로 표기

Min-Max Notation of ER Diagram

Alternative ER Notations



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

N-ary Relationship

- 2진관계 (Binary)
 - 차수가 2인 관계 타입을 2진 관계라 한다.
- 3진(Ternary) 관계, n-ary 관계
 - 차수가 3인 관계 타입을 3진 관계라 하고, 차수가 n인 관계를 n-ary 관계라 한다.
- “n-ary 관계”와 “n개의 이진 관계들” 사이의 차이
 - 일반적으로, n-ary 관계는 n개의 이진 관계들과 같지 않다.
- **cardinality는 하나의 entity type과 다른 나머지 entity type과의 cardinality를 고려하면, 이진 관계의 cardinality와 같아진다.**

Ternary Relationship

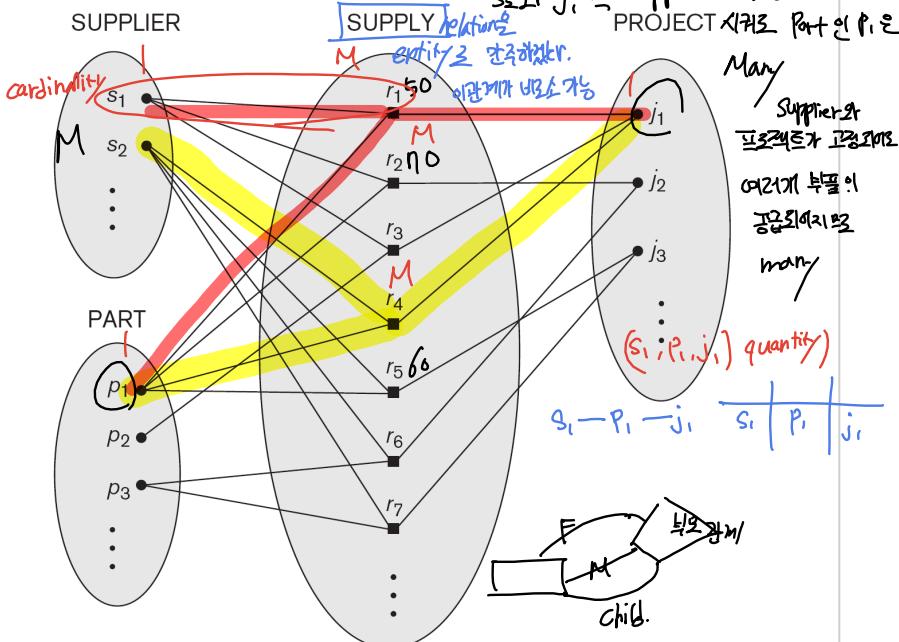
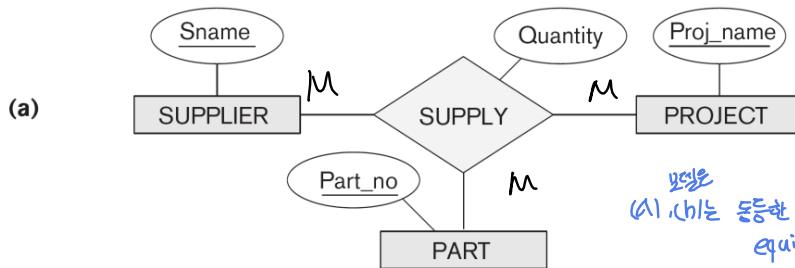
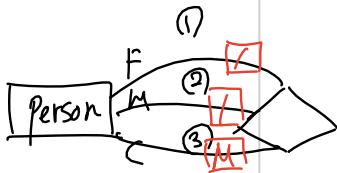
ex) P_1, j_1 은 고정시킬 때 S_1, j_1 즉 Supplier가 Project를 향해PROJECT 시점으로 Part인 P_1 을

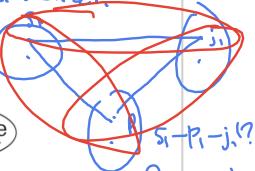
Figure 7.10
Some relationship instances in
the SUPPLY ternary relationship
set.



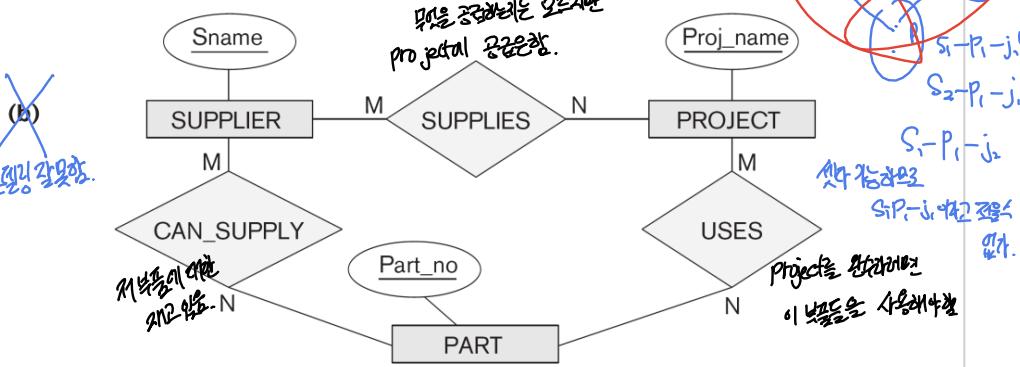
Ternary Relationship and Three Binary Relationships



모임은
(a), (b)는 동등한 능력을 가진다?
equivalent인가?



(b)
모델링 잘못함.



모임을 공급하는 모집과는
Project 공급처.
Supplier 공급처.
Project를 공급하는
이 봉수를 사용해야

Supplier 공급처
Project를 공급하는
이 봉수를 사용해야

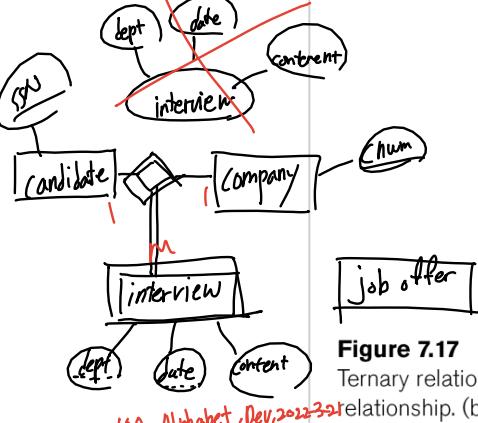
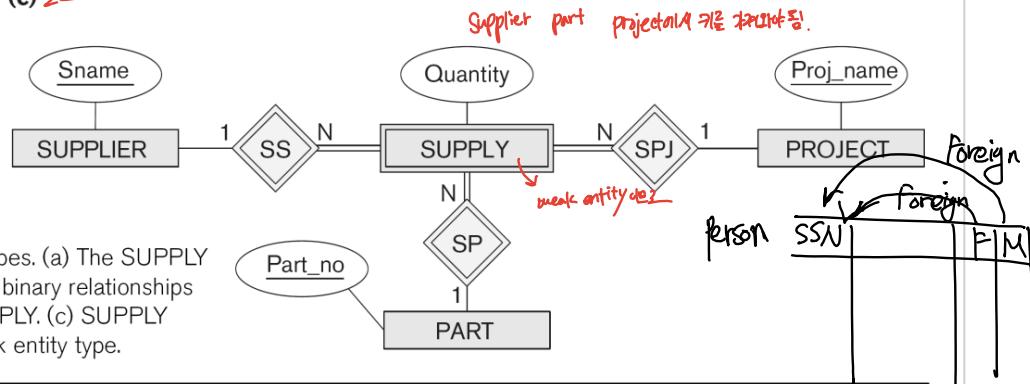


Figure 7.17

Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

(c) 결론



Supplier part project에서 키를 가져와야됨.

weak entity def

Limitations of ER Model

- 원래 ER 모델은 객체 모델링 기법에서 제공하는 추상화(특수화/일반화) 기능을 지원하지 못함
- 확장 엔티티-관계(EER, Enhanced ER) 모델
 - 집합-부분집합 관계를 포함.
 - 특수화/일반화 계층구조를 포함 .

ER Diagram for Bank Database

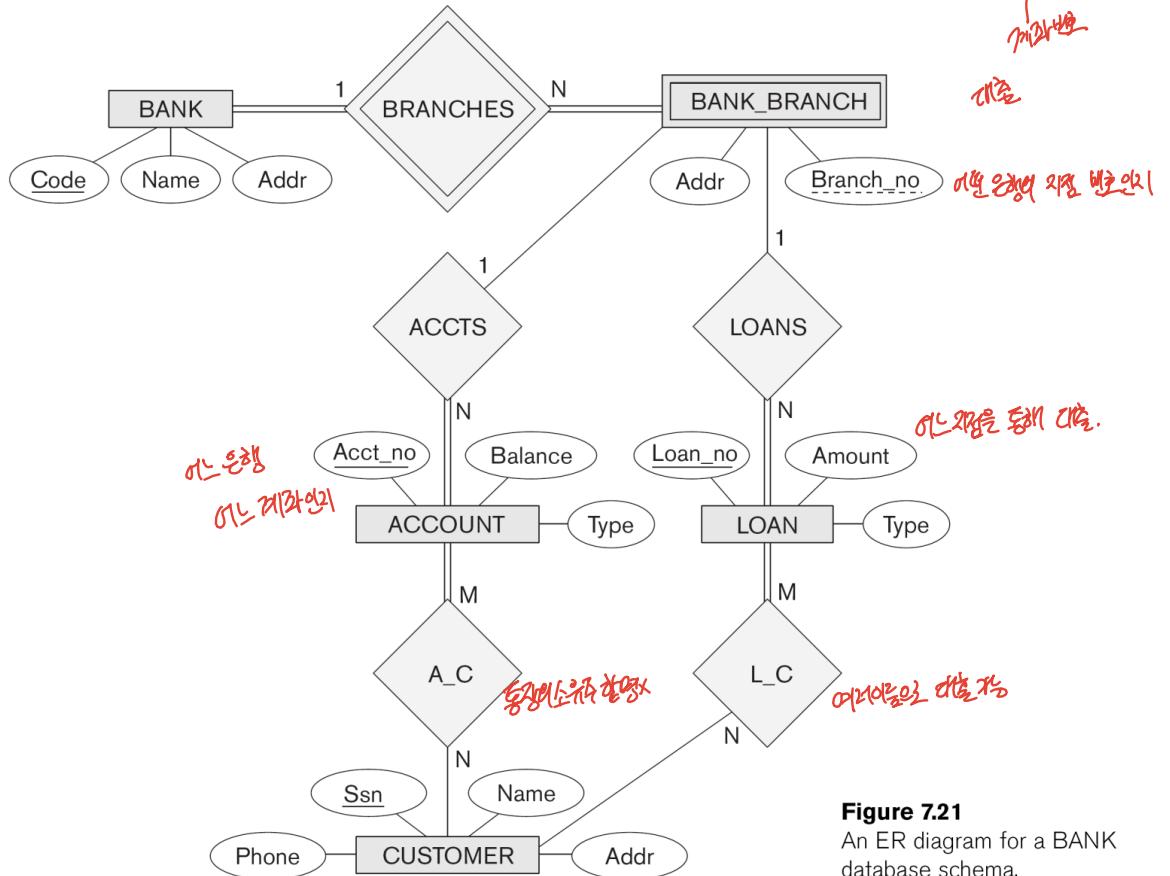


Figure 7.21
An ER diagram for a BANK database schema.

LAB: Requirements for University DB

- Draw ER Diagram

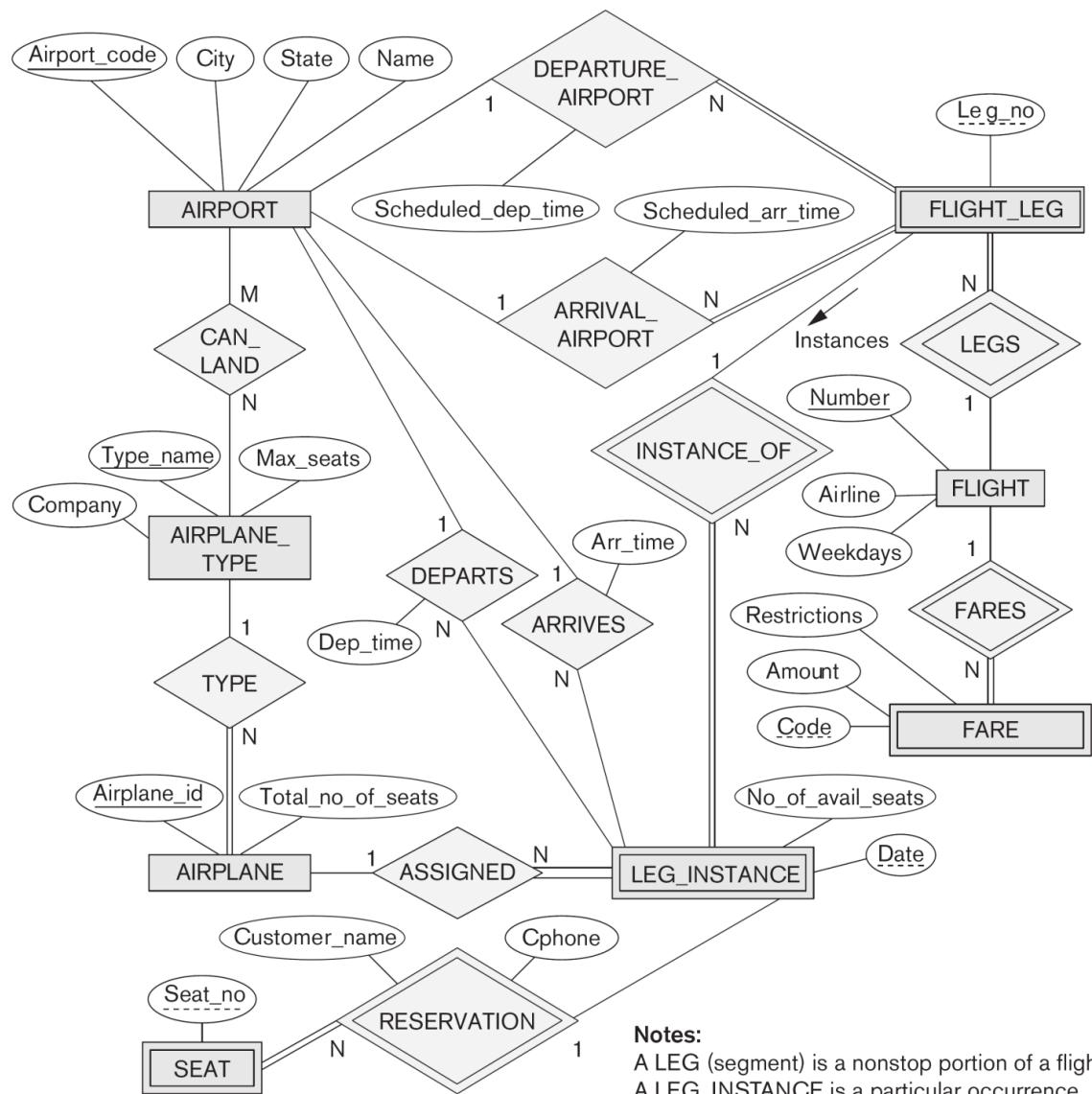
1221 03/14

- DBD 2022 | (1)
- DBD 2022 | (2)
- (a) The university keeps track of each student's name, student number, social security number, current address and phone, permanent address and phone, birthdate, sex, class (freshman, sophomore, ..., graduate), major department, minor department (if any), and degree program (B.A., B.S., ..., Ph.D.). Some user applications need to refer to the city, state, and zip of the student's permanent address, and to the student's last name. Both social security number and student number have unique values for each student.
 - (b) Each department is described by a name, department code, office number, office phone, and college. Both name and code have unique values for each department.
 - (c) Each course has a course name, description, course number, number of semester hours, level, and offering department. The value of course number is unique for each course.
 - (d) Each section has an instructor, semester, year, course, and section number. The section number distinguishes different sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ...; up to the number of sections taught during each semester.
 - (e) A grade report has a student, section, letter grade, and numeric grade (0, 1, 2, 3, 4 for F, D, C, B, A, respectively).

Example ER Diagram

Figure 7.20

An ER diagram for an AIRLINE database schema.

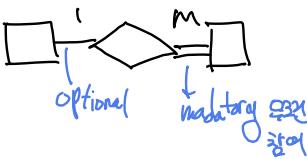
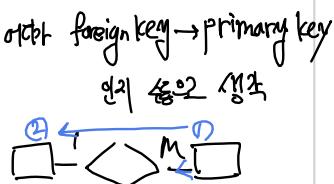


ER supplementary. Data Modeling using MySQL Workbench

Install MySQL Workbench from

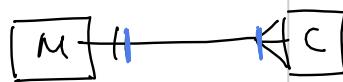
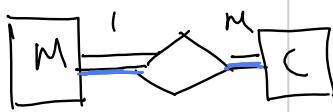
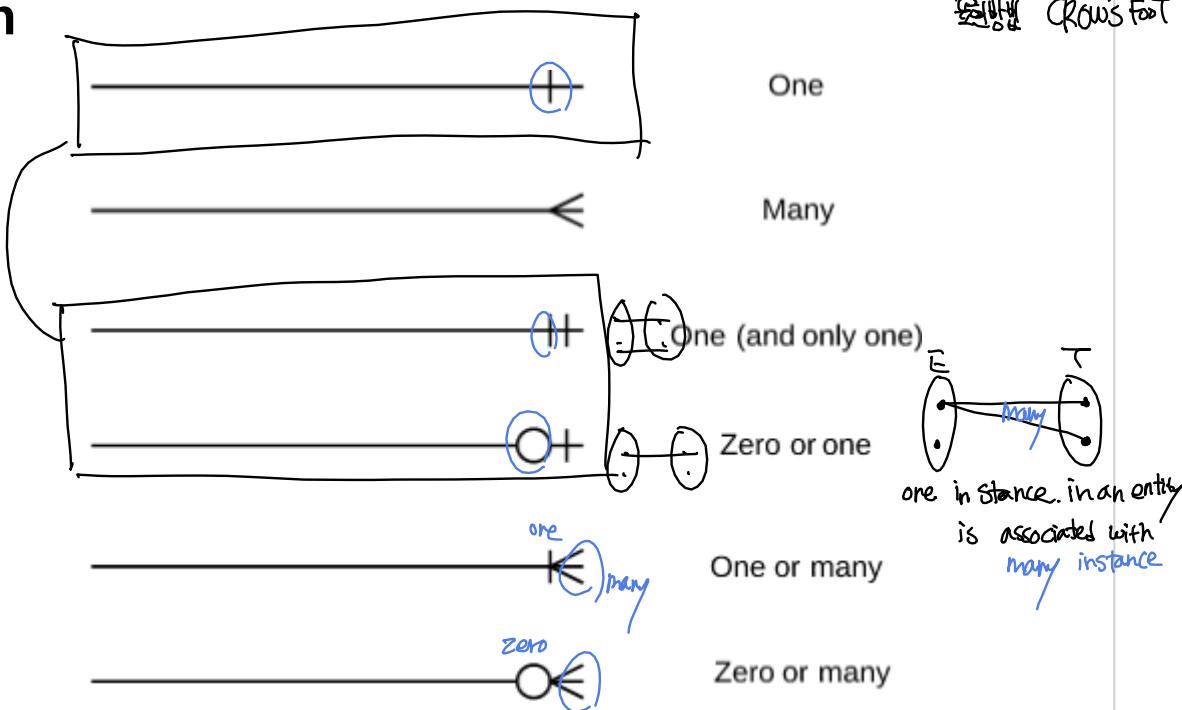
<https://dev.mysql.com/downloads/workbench/>
[\(https://dev.mysql.com/downloads/workbench/\)](https://dev.mysql.com/downloads/workbench/)

- IE Notation ER 모양 다시
- No composite or multi-valued attributes are allowed. You need to transform them manually.
- Weak entity is "identifying relationship" in IE notation
- To draw 1:1 relationship, first click 1-side (foreign key side) and then the other 1-side entity type (primary key side)
- To draw 1:M relationship, first click many-side (foreign key side) and then 1-side entity type
- M:N relationship will be automatically transformed into two 1:M relationships Many side 조건 one side 조건
- Ternary or more relationship is not supported. Designer need to transform them into binary relationships manually 트리티지 4개
- Superclass and Union type is not supported. Designer need to transform them manually. 제작자
- Relationship attributes are not supported. Move them into foreign key side.
- All these manual transformations are covered in the "Mapping" chapter.
- Cardinality and Optional/Mandatory participation notation of binary relationship is as follows:



One instance in an entity type is associated with

한 개의 행에 Crow's Foot



instances in the other entity type.

Easy way to remember:

Consider mother 1..1 <-----> 1..m children relationship

- one mother has ONE or MANY children
- one child has ONE and ONLY ONE mother
- cardinality 값은 가까운 곳에 붙는다고 생각하면 기억하기 쉽다.



Chen's Notation으로 쓰여진 CompanyDB ER Diagram을 IE(Crow's Foot) Notation을 작성하여 보자.

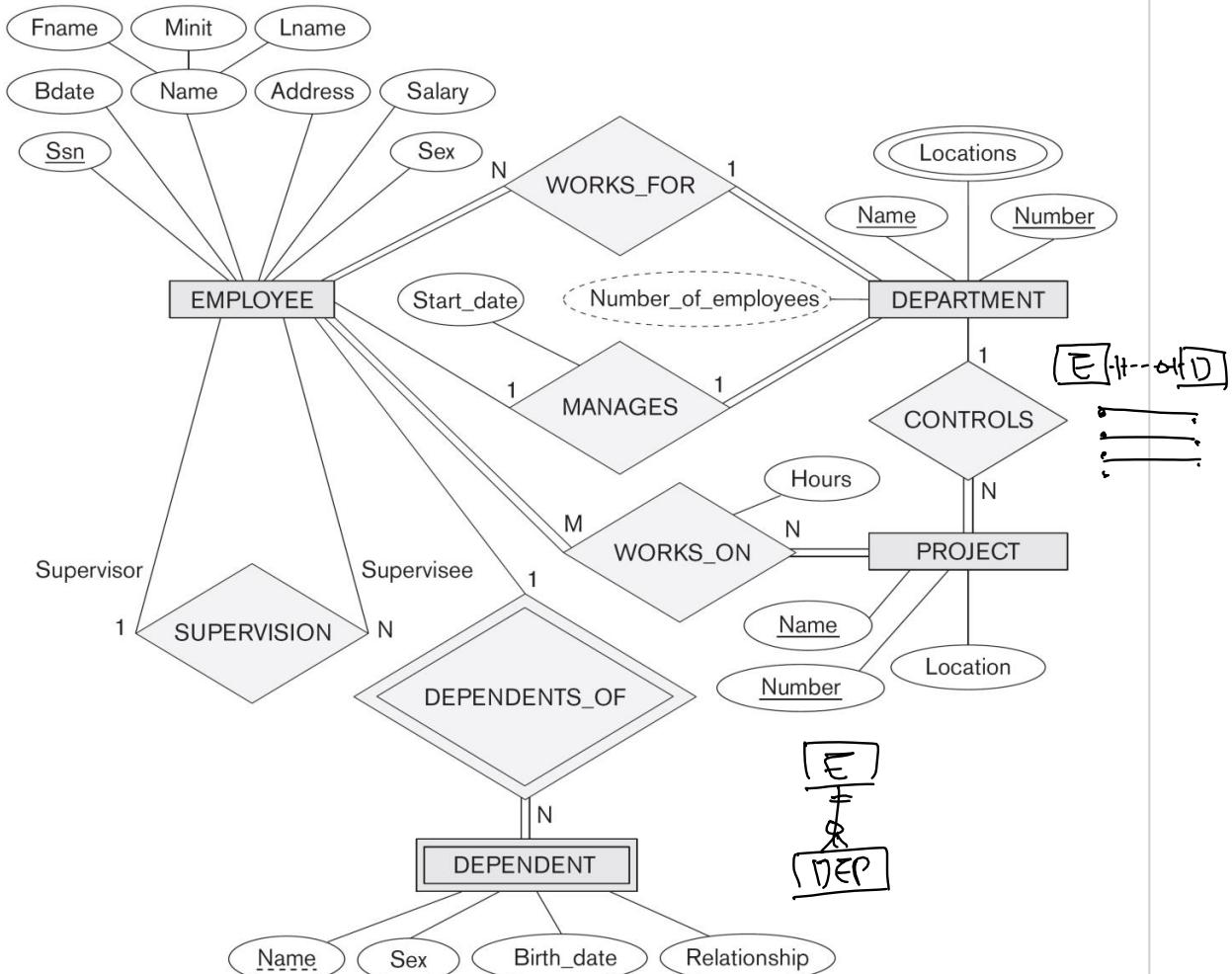


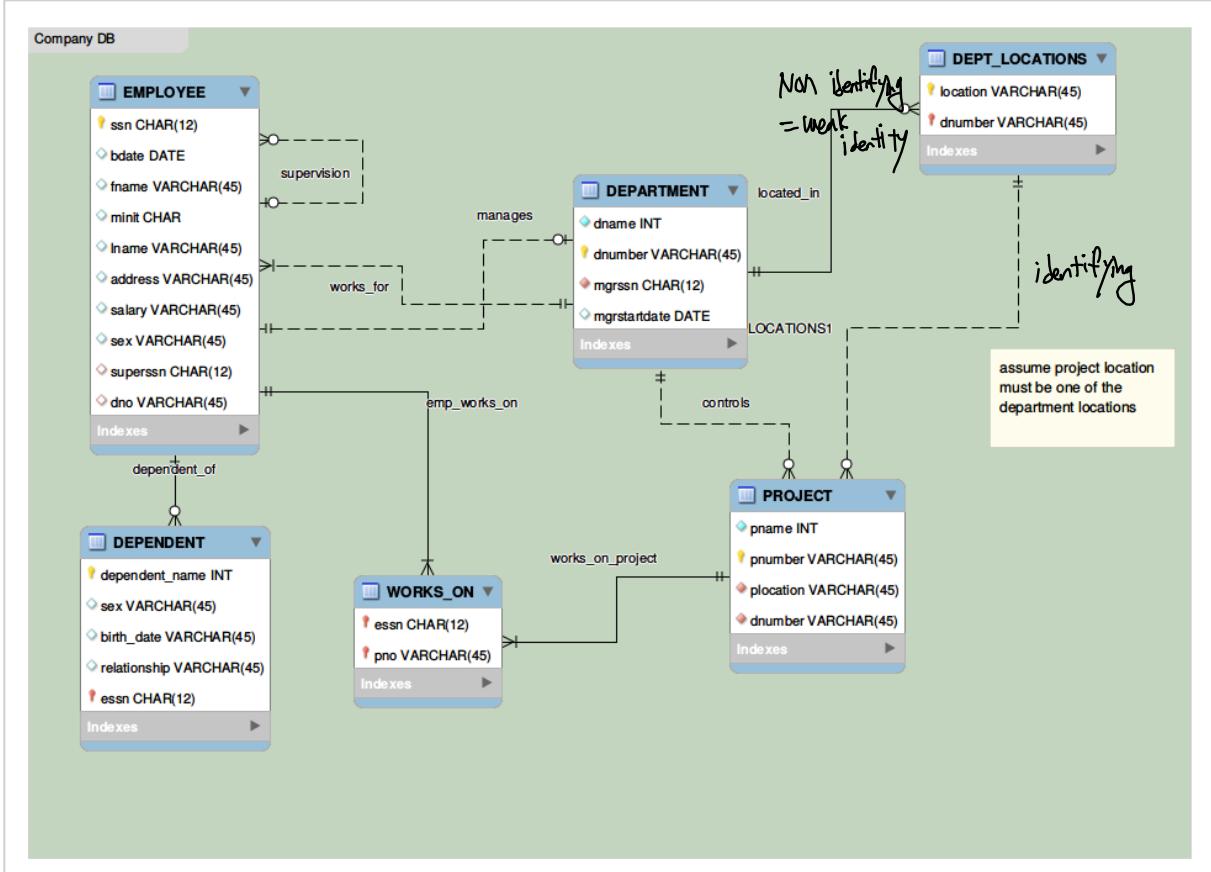
Figure 7.2

An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 7.14.

- Relationship name을 표시하려면, Model->Model Option->Diagram->Show Captions에서 show caption을 check해야 한다.

각 테이블의 Foreign Keys 탭에서 On Update/On Delete Foreign Keys Options에서 적절한 Option 을 선택한다.

- RESTRICTED
- SET NULL
- CASCADE
- (NO ACTION은 MySQL에서는 RESTRICTED)
- mandatory option may be overridden by taking 'set null'



Forward Engineering

```
-- Table `mydb`.`DEPARTMENT`  
-----  
CREATE TABLE IF NOT EXISTS `mydb`.`DEPARTMENT` (  
  `dname` INT NOT NULL,  
  `dnumber` VARCHAR(45) NOT NULL,  
  `mgrssn` CHAR(12) NOT NULL,  
  `mgrstartdate` DATE NULL,  
  UNIQUE INDEX `name_UNIQUE` (`dname` ASC),  
  PRIMARY KEY (`dnumber`),  
  INDEX `fk_DEPARTMENT_EMPLOYEE1_idx` (`mgrssn` ASC),  
  CONSTRAINT `fk_DEPARTMENT_EMPLOYEE1`  
    FOREIGN KEY (`mgrssn`)
```

from DBSON1188

Show create table department\G

```
REFERENCES `mydb`.`EMPLOYEE`(`ssn`)
ON DELETE SET NULL
ON UPDATE CASCADE)
ENGINE = InnoDB;
```

...

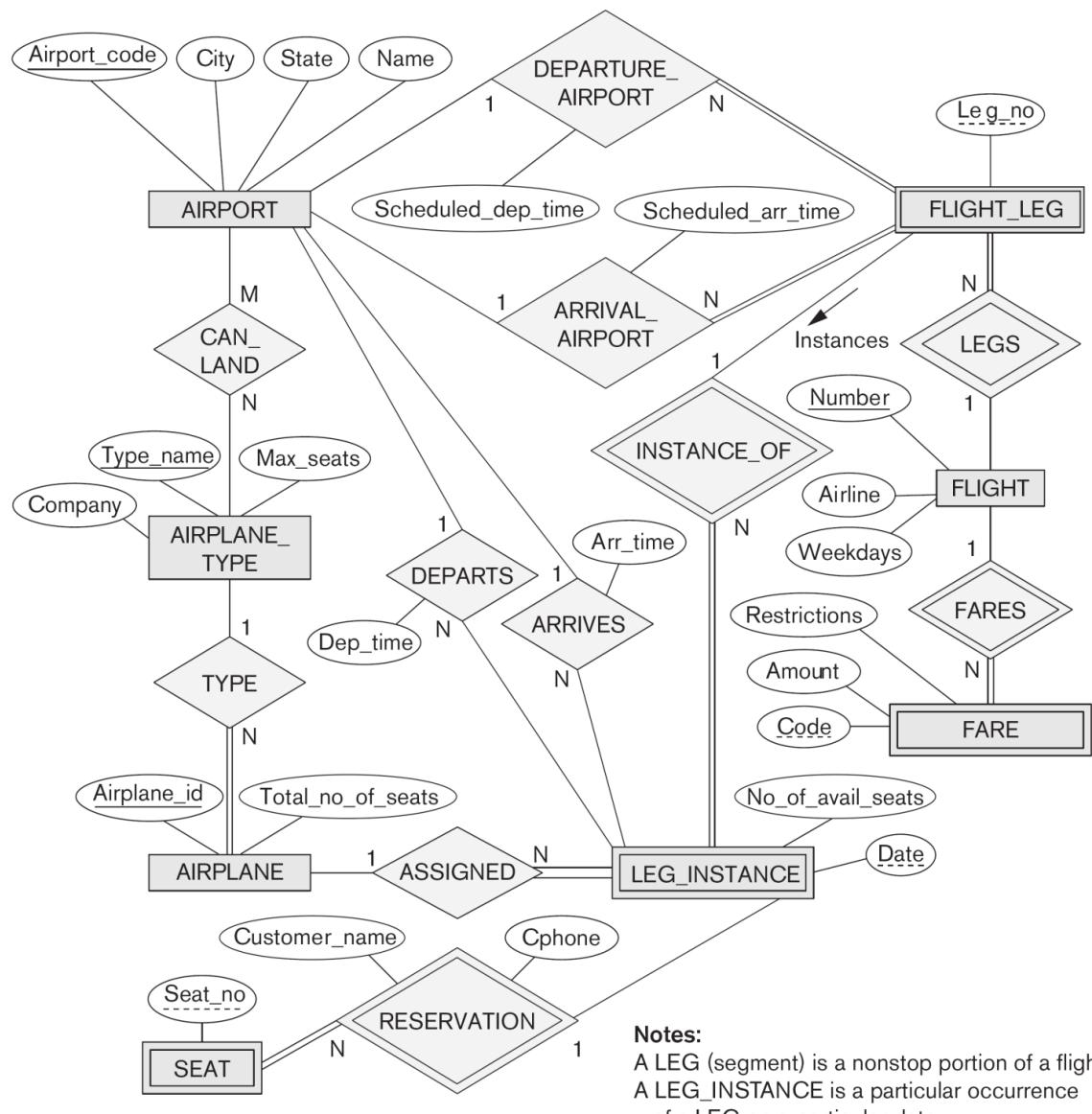
If project location must be one of the dept_locations,

- Pick "Relationship using existing columns"
- First, click plocation in PROJECT
- Then, location in DEPT_LOCATIONS

Homework

Figure 7.20

An ER diagram for an AIRLINE database schema.



- MySQL Workbench EER Modeling을 사용하여, AIRLINE DB에 대한 Crow's Notation으로 된 ER Diagram을 작성하시오. 데이터 탑입도 어느 정도는 적절하게 선택되어야 한다.
- Forward Engineering Tool을 이용하여, MySQL Server에 schema를 생성하시오. SQL create table을 출력하시오.

- MySQL workbench ER Model is almost relational database schema.
- ERWin is more powerful, professional ER modeling tool (supports many-many, ternary, supertype-subtype relationship, and many more) (You can download trial version at <http://go.erwin.com/erwin-data-modeler-free-trial> (<http://go.erwin.com/erwin-data-modeler-free-trial>)).

4th chapter

Set theory 집합론

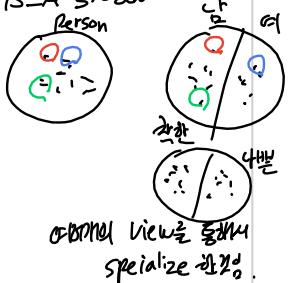
각각의 개별은 집합을 이루나요

Super class Sub class

모든 (C) 은 멤버임.

obj Is-A obj

person is-A student



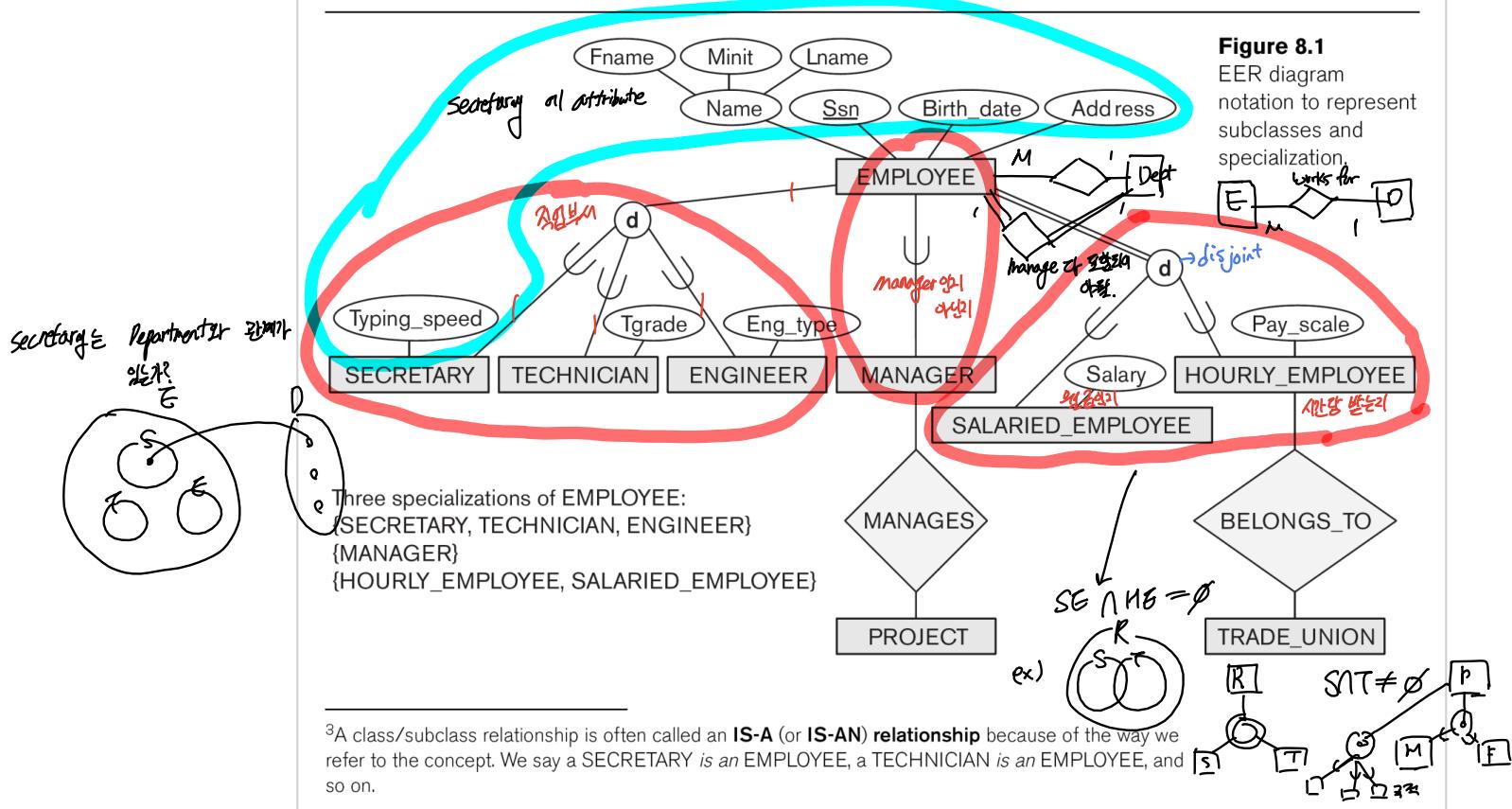
1.1 Contents

- EER stands for Enhanced ER or Extended ER
- EER Model Concepts
 - Includes all modeling concepts of basic ER
 - Additional concepts:
 - subclasses/superclasses
 - specialization/generalization
 - categories (UNION types)
 - These are fundamental to conceptual modeling
- The additional EER concepts are used to model applications more completely and more accurately
 - EER includes some object-oriented concepts, such as inheritance

1.2 Subclasses and Superclasses

- An entity type may have additional meaningful subgroupings of its entities
 - Example: EMPLOYEE may be further grouped into:
 - SECRETARY, ENGINEER, TECHNICIAN, ...
 - Based on the EMPLOYEE's Job
 - MANAGER
 - EMPLOYEES who are managers
 - SALARIED_EMPLOYEE, HOURLY_EMPLOYEE
 - Based on the EMPLOYEE's method of pay
- EER diagrams extend ER diagrams to represent these additional subgroupings, called subclasses or subtypes

1.3 Subclasses and Superclasses



1.4 Subclasses and Superclasses

- Each of these subgroupings is a subset of EMPLOYEE entities
- Each is called a subclass of EMPLOYEE
- EMPLOYEE is the superclass for each of these subclasses
- These are called superclass/subclass relationships:
 - EMPLOYEE/SECRETARY
 - EMPLOYEE/TECHNICIAN
 - EMPLOYEE/MANAGER
 - ...

$$\text{Salary}_E \cup \text{Man}_E = \text{Employee}$$

$$\text{Manager} \subseteq \text{Employee}$$

$$\text{SUVE} \subseteq E$$

1.5 Subclasses and Superclasses

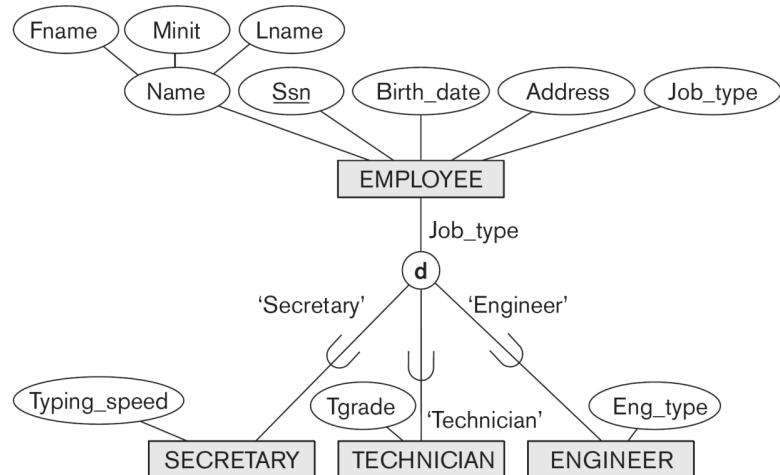
- These are also called **IS-A** relationships
 - SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE,
- Note: An entity that is member of a subclass represents the same real-world entity as some member of the superclass:
 - The subclass member is the same entity in a distinct specific role
 - An entity cannot exist in the database merely by being a member of a subclass;** it must also be a member of the superclass
 - A member of the superclass can be optionally included as a member of any number of its subclasses

1.6 Subclasses and Superclasses

- Examples:
 - A salaried employee who is also an engineer belongs to the two subclasses:
 - ENGINEER, and
 - SALARIED_EMPLOYEE
 - A salaried employee who is also an engineering manager belongs to the three subclasses:
 - MANAGER,
 - ENGINEER, and
 - SALARIED_EMPLOYEE
- It is not necessary that every entity in a superclass be a member of some subclass

1.7 Representing Specialization in EER Diagrams

Figure 8.4
EER diagram notation for an attribute-defined specialization on Job_type.



⁶Such an attribute is called a *discriminator* in UML terminology.

1.8 Attribute Inheritance in Superclass / Subclass Relationships

- An entity that is member of a subclass inherits
 - All attributes of the entity as a member of the superclass
 - **All relationships of the entity as a member of the superclass**
- Example:
 - In the previous slide, SECRETARY (as well as TECHNICIAN and ENGINEER) inherit the attributes Name, SSN, ..., from EMPLOYEE
 - Every SECRETARY entity will have values for the inherited attributes

1.9 Specialization

- Specialization is the process of defining a set of subclasses of a superclass

- The set of subclasses is based upon some distinguishing characteristics of the entities in the superclass
 - Example: {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of EMPLOYEE based upon job type.
 - May have several specializations of the same superclass

1.10 Instance of a Specialization

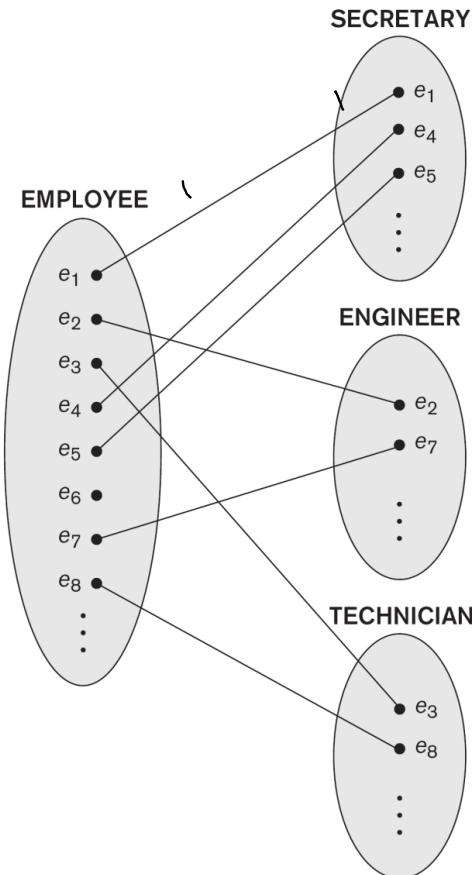


Figure 8.2
Instances of a specialization.

1.11 Specialization

- Example: Another specialization of EMPLOYEE based on method of pay is {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE}.
 - Superclass/subclass relationships and specialization can be diagrammatically represented in EER diagrams
 - Attributes of a subclass are called specific or local attributes.
 - For example, the attribute TypingSpeed of SECRETARY
 - The subclass can also participate in specific relationship types.
 - For example, a relationship BELONGS_TO of HOURLY_EMPLOYEE

1.12 Specialization

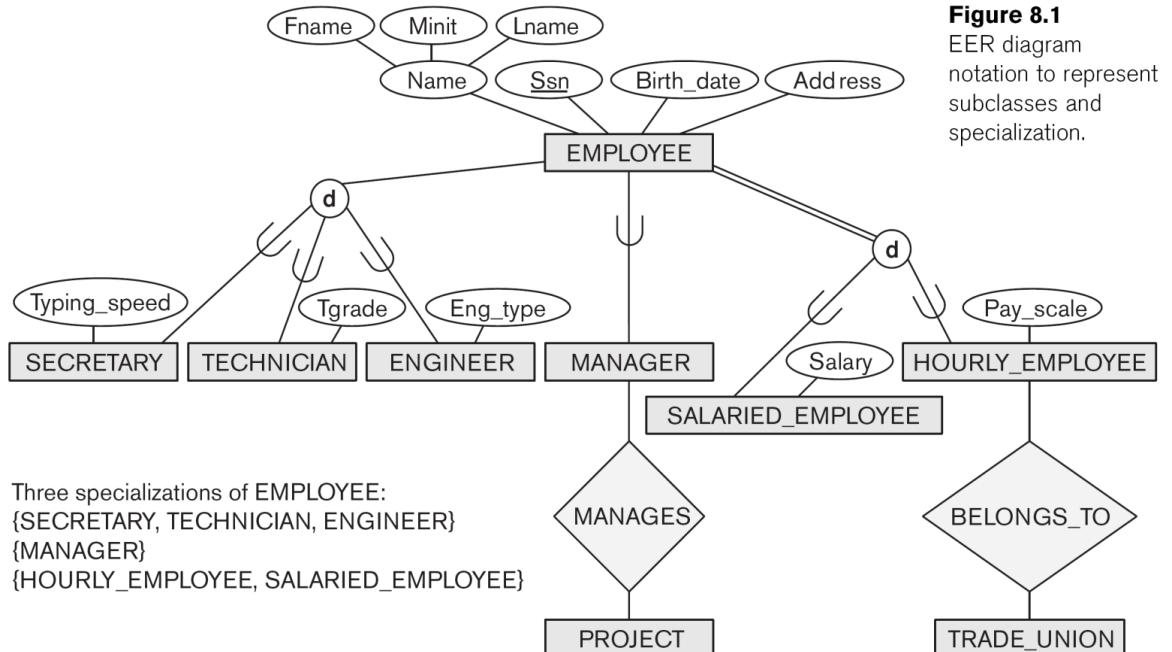


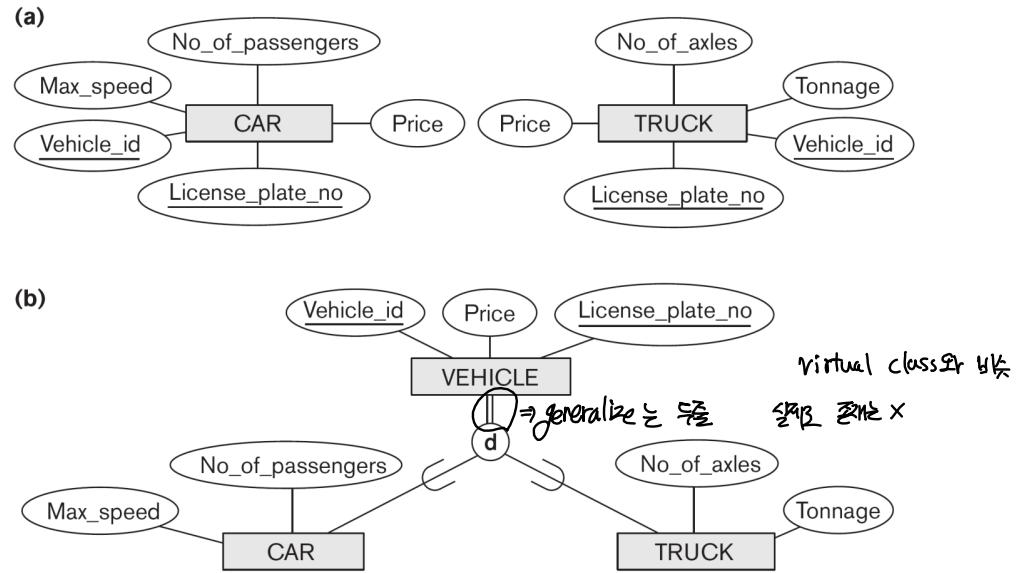
Figure 8.1
 EER diagram notation to represent subclasses and specialization.

³A class/subclass relationship is often called an **IS-A** (or **IS-AN**) **relationship** because of the way we refer to the concept. We say a SECRETARY *is an* EMPLOYEE, a TECHNICIAN *is an* EMPLOYEE, and so on.

1.13 Generalization

- Generalization is the reverse of the specialization process
- Several classes with common features are generalized into a superclass;
 - original classes become its subclasses
- Example: CAR, TRUCK generalized into VEHICLE;
 - both CAR, TRUCK become subclasses of the superclass VEHICLE.
 - We can view {CAR, TRUCK} as a specialization of VEHICLE
 - Alternatively, we can view VEHICLE as a generalization of CAR and TRUCK

1.14 Generalization

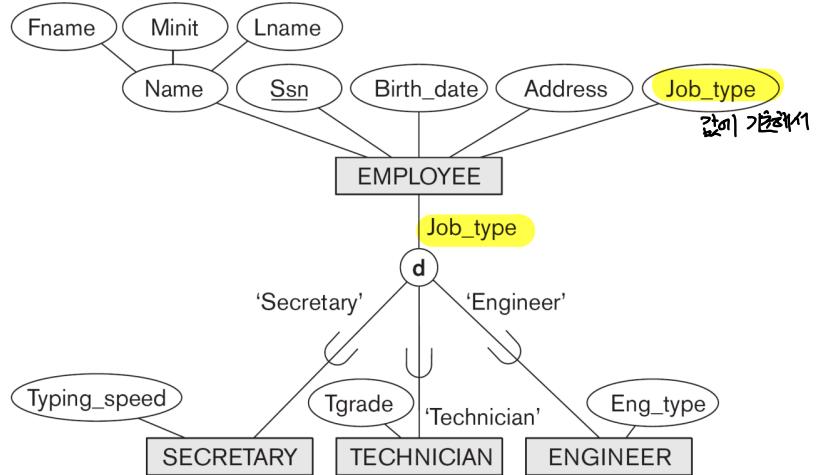
**Figure 8.3**

Generalization. (a) Two entity types, CAR and TRUCK. (b)
Generalizing CAR and TRUCK into the superclass VEHICLE.

1.15 Example of disjoint partial Specialization

Figure 8.4

EER diagram notation for an attribute-defined specialization on Job_type.



⁶Such an attribute is called a *discriminator* in UML terminology.

1.16 Example of overlapping total Specialization

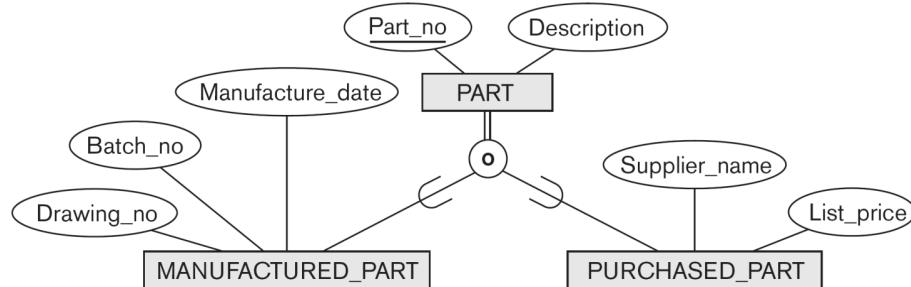


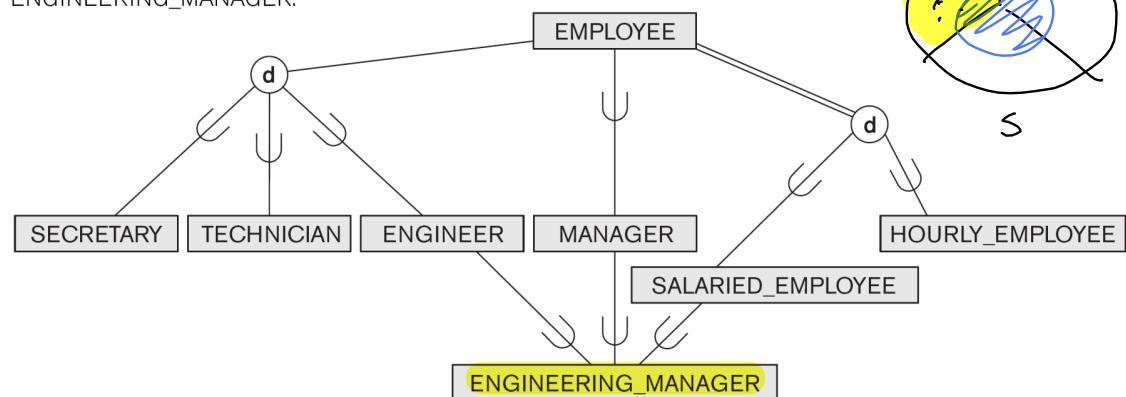
Figure 8.5
EER diagram notation for an overlapping (nondisjoint) specialization.

⁷The notation of using single or double lines is similar to that for partial or total participation of an entity type in a relationship type, as described in Chapter 7.

1.17 Multiple Inheritance: Shared Subclass “Engineering_Manager” \Rightarrow lattice

Figure 8.6

A specialization lattice with shared subclass **ENGINEERING_MANAGER**.



1.18 Specialization / Generalization Lattice Example (UNIVERSITY)

Partial order

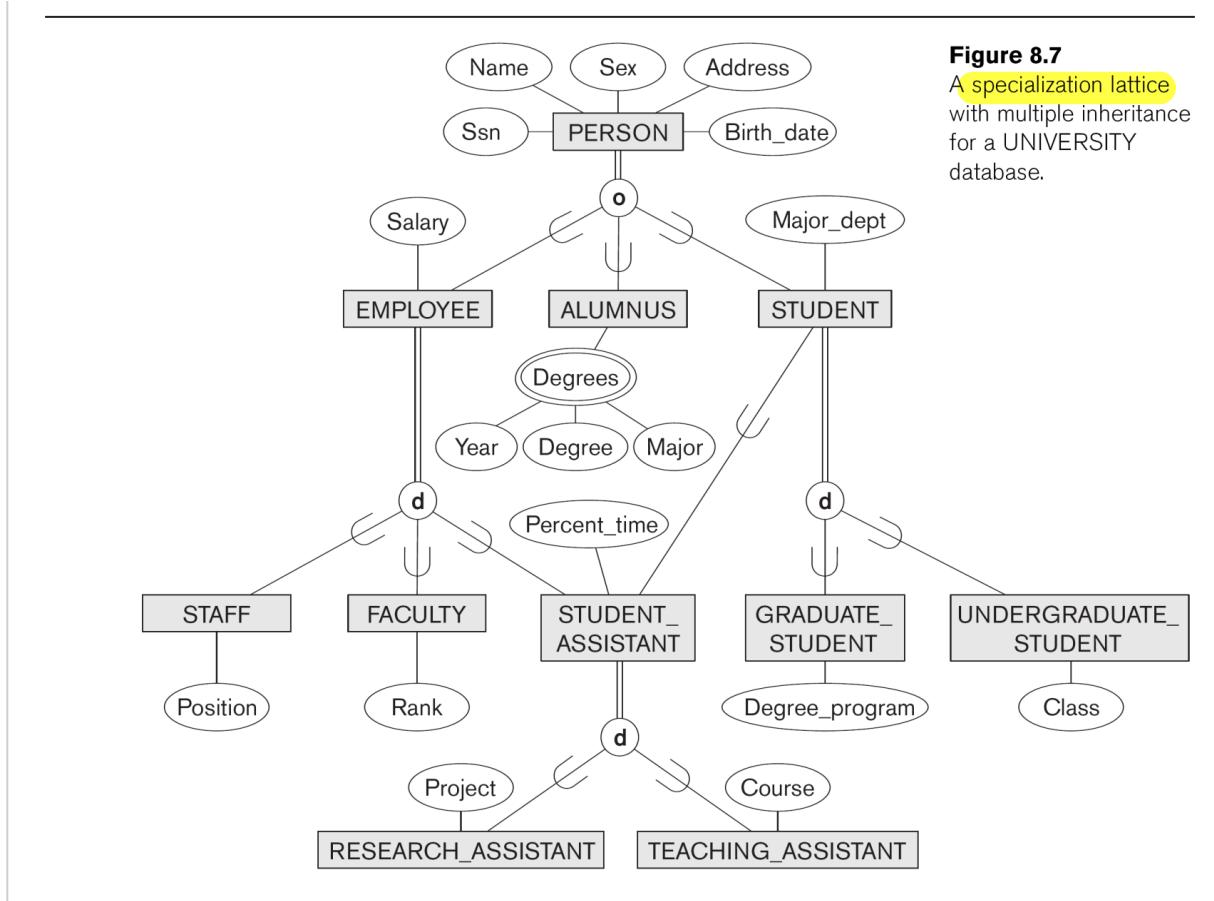


Figure 8.7
A specialization lattice with multiple inheritance for a UNIVERSITY database.

1.19 Categories (UNION TYPES)

- All of the superclass/subclass relationships we have seen thus far have a single superclass
- A shared subclass is a subclass in:
 - more than one distinct superclass/subclass relationships
 - each relationship has a single superclass
 - shared subclass leads to multiple inheritance
- In some cases, we need to model a single superclass/subclass relationship with more than one superclass
- Superclasses can represent different entity types
- Such a subclass is called a category or UNION TYPE

1.20 Categories (UNION TYPES)

- Example: In a database for vehicle registration, a vehicle owner can be a PERSON, a BANK (holding a lien on a vehicle) or a COMPANY.
 - A category (UNION type) called OWNER is created to represent a subset of the union of the three superclasses COMPANY, BANK, and PERSON
 - A category member must exist in at least one of its superclasses
- Difference from shared subclass, which is a:
 - subset of the intersection of its superclasses
 - shared subclass member must exist in all of its superclasses

1.21 Two categories (UNION types): OWNER, REGISTERED_VEHICLE

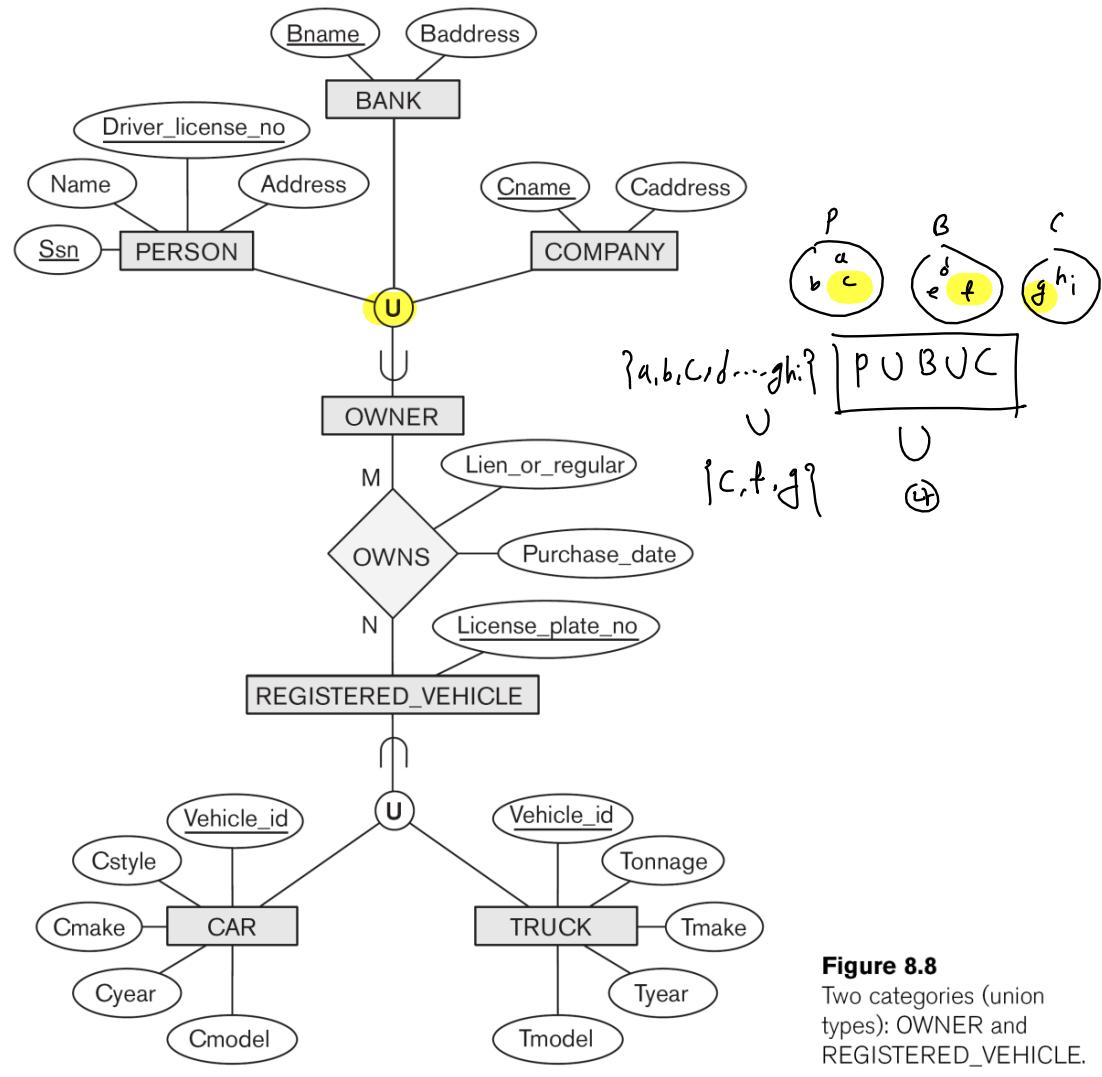


Figure 8.8
Two categories (union types): OWNER and REGISTERED_VEHICLE.

1.22 LAB

Consider the entity sets and attributes shown in the table below. Place a checkmark in one column in each row to indicate the relationship between the far left and right columns.

1. The left side has a relationship with the right side.
2. The right side is an attribute of the left side.
3. The left side is a specialization of the right side.
4. The left side is a generalization of the right side.

Entity Set	(a) Has a Relationship with	(b) Has an Attribute that is	(c) Is a Specialization of	(d) Is a Generalization of	Entity Set or Attribute
1. MOTHER					PERSON
2. DAUGHTER					MOTHER
3. STUDENT					PERSON
4. STUDENT					Student_id
5. SCHOOL					STUDENT
6. SCHOOL					CLASS_ROOM
7. ANIMAL					HORSE
8. HORSE					Breed
9. HORSE					Age
10. EMPLOYEE					SSN
11. FURNITURE					CHAIR
12. CHAIR					Weight
13. HUMAN					WOMAN
14. SOLDIER					PERSON
15. ENEMY_COMBATANT					PERSON

1.23 LAB

- Consider the BANK ER schema, and suppose that it is necessary to keep track of different types of ACCOUNTS (SAVINGS_ACCTS, CHECKING_ACCTS, TRUST,...) and LOANS (CAR_LOANS, HOME_LOANS, PERSONAL, ...).
- Suppose that it is also desirable to keep track of each account's TRANSACTIONS (deposits, withdrawals, checks,...) and each loan's PAYMENTS;
- both of these include the amount, date, time,
- Modify the BANK schema, using EER concepts of specialization and generalization.

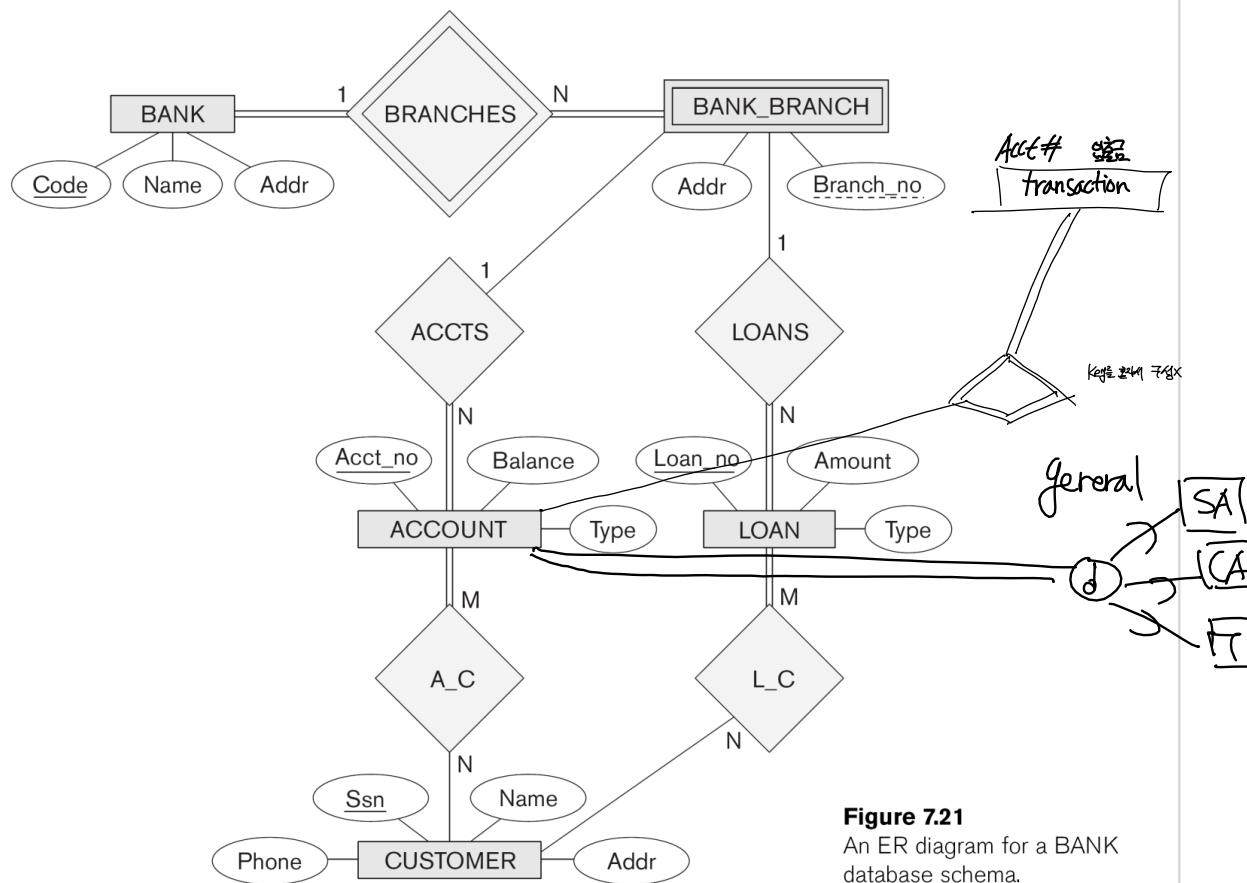


Figure 7.21

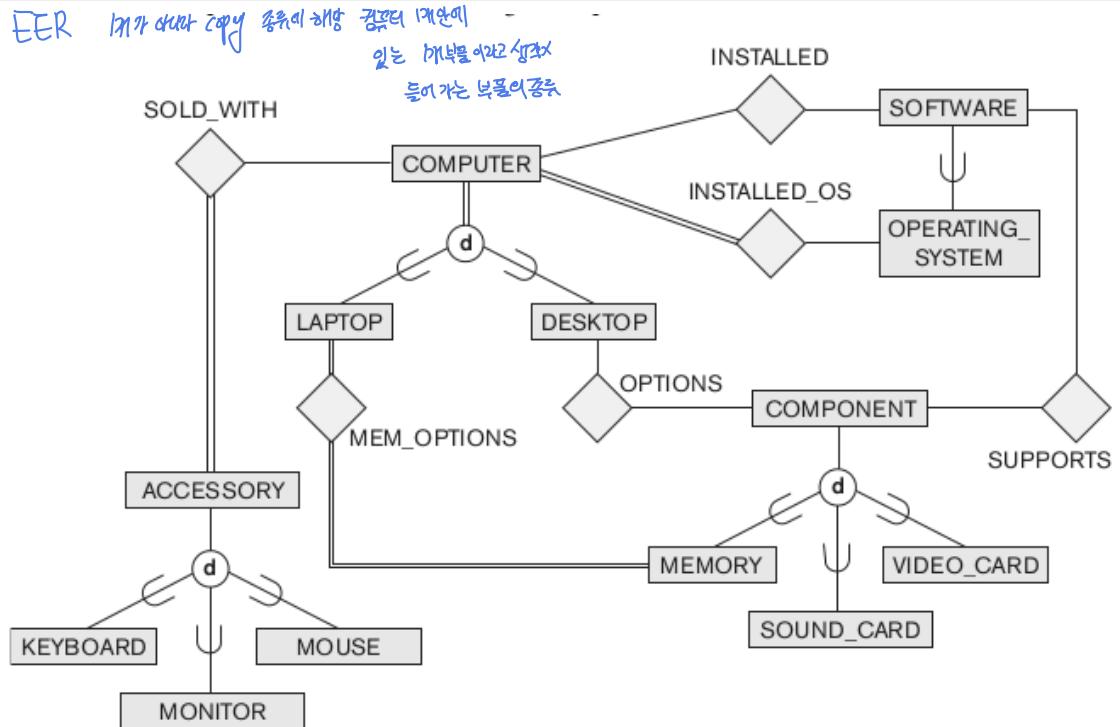
An ER diagram for a BANK database schema.

1.24 LAB

Design an EER diagram that describes computer systems at a company.

Hints

- Nouns: Software, Operating System, Laptop, Desktop, Memory, Sound Card, Video Card, Keyboard, Mouse, Monitor \Rightarrow Accessory
- Verbs: Options, Sold with, Installed



1.25 LAB: University Database

Nouns:

- Faculty,
- Student,
- Grad_student,
- Department,
- Section,
- Current_section,
- Course,
- Grant, *연금비 (외부 자금원을 가져온다)*
- College *공과대학*

*undergraduate 학부생
graduate 대학원생*

책임인자 (principal Investigator)



Verbs

- Faculty and Grad_student can teach section *교수하는 일*.
- Faculty advises Grad_student *지도하는 일*

- Faculty members are a thesis committee of a grad_student 교수님은 학생을
- Faculty and Grad_student are supported by grant 연구원과 학생을 150만원 정도 지원합니다.
- Faculty is a PI of grant PI principal Investigator
- student has a relationship with a current section
- student has a relationship with a section
- student has a relationship with a department
- Faculty has a relationship with a department
- College has a relationship with a department

1.26 University Database Example

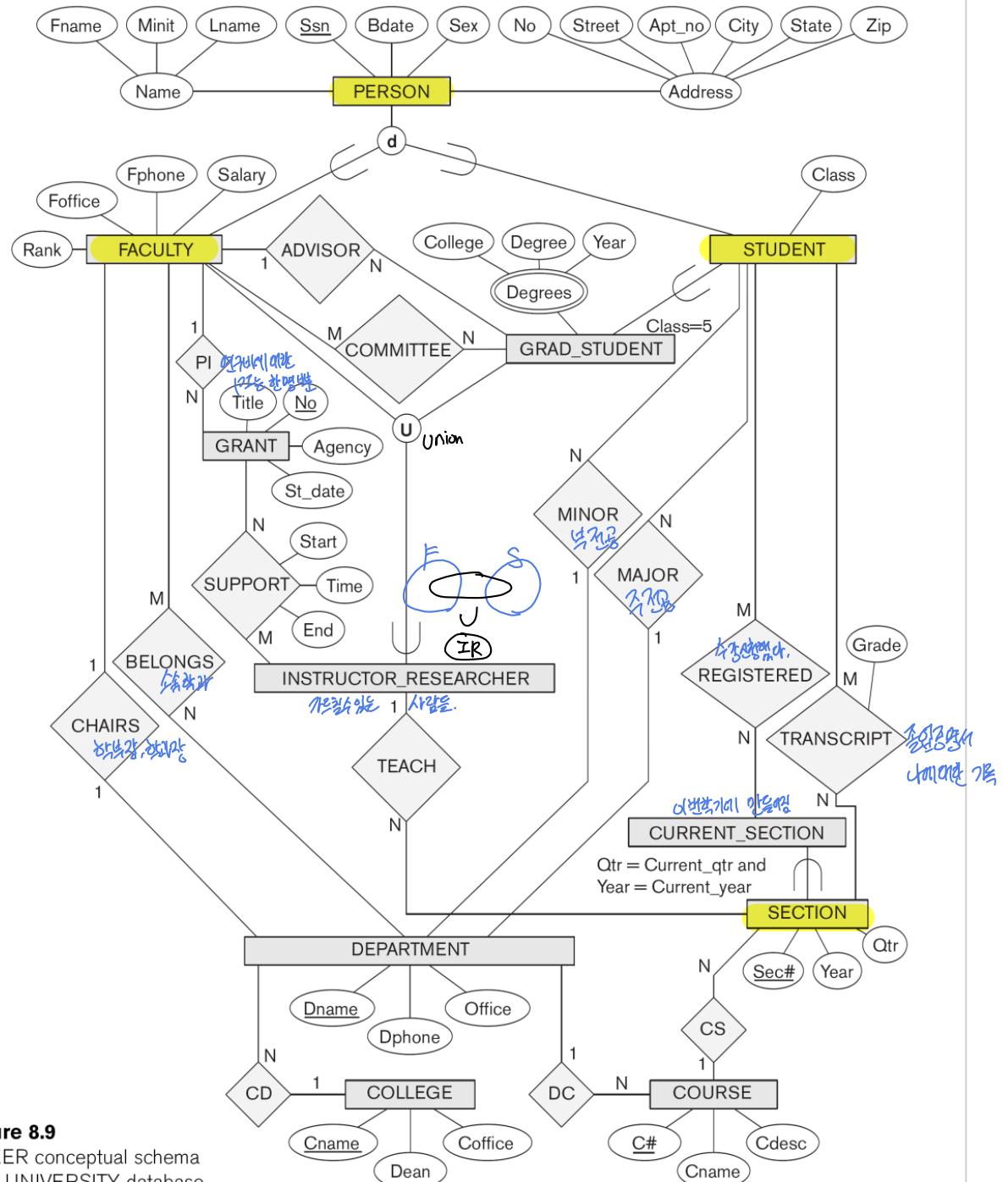
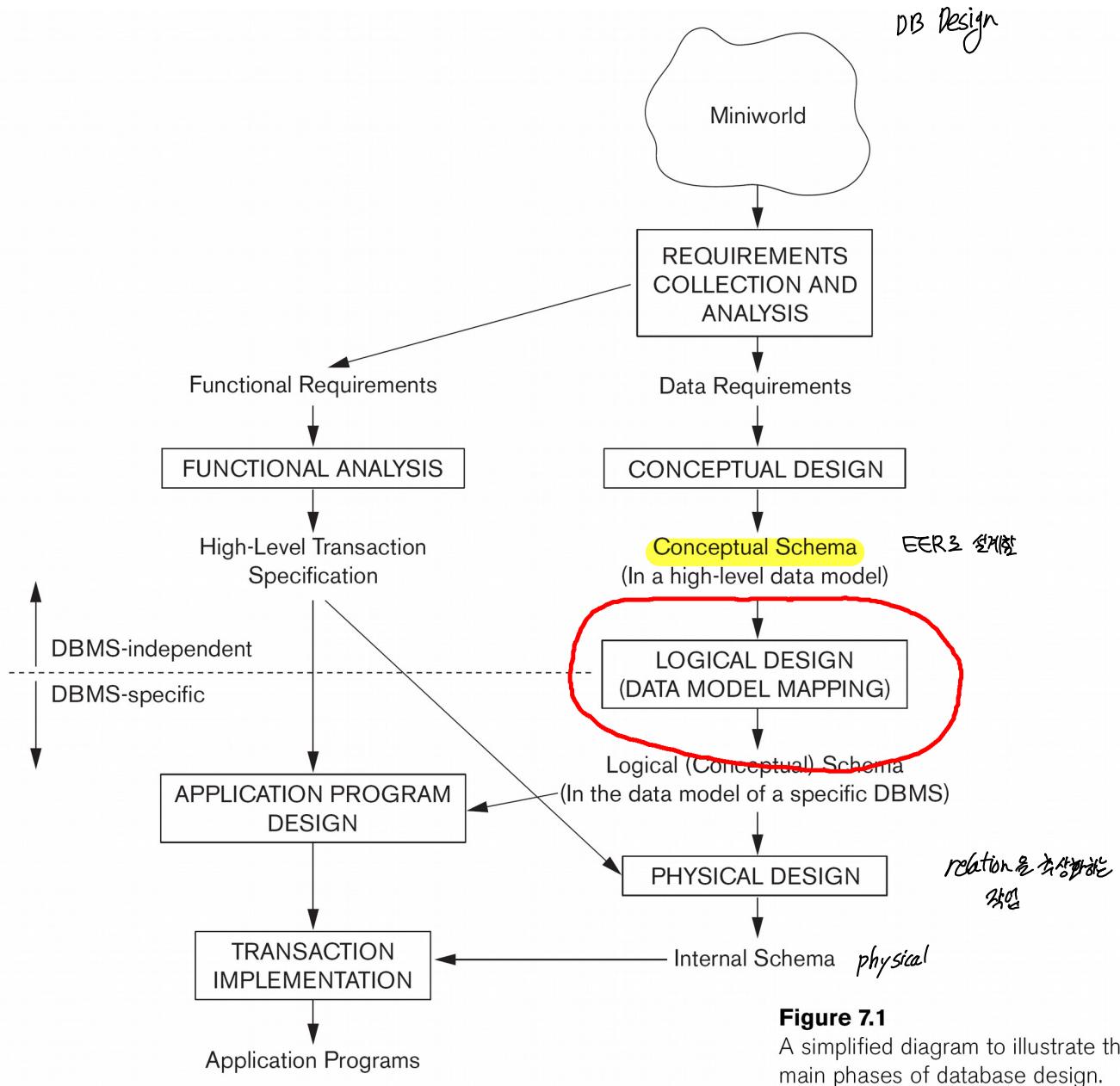


Figure 8.9
An EER conceptual schema
for a UNIVERSITY database.

Ch 8. Relational Database Design by ER- and EER-Relation Mapping

Where are we?



Relational Database Design by ER-Relation Mapping

- STEP 1
 - 엔티티 타입은 릴레이션으로 매핑한다. 엔티티 타입의 키 중에서 하나를 릴레이션의 기본 키로 지정한다.
- STEP 2
 - 약한 엔티티 타입도 릴레이션으로 매핑하되 소유 릴레이션 (owner relation)의 키 속성을 포함시킨다. 생성된 릴레이션의 기본 키는 소유 릴레이션의 키와 약한 엔티티 타입의 부분키를 합쳐서 만든다.
- STEP 3

- 1:1 이진 관계는 관계에 참여하는 두 릴레이션 중에서 어느 하나의 (전체 참여 엔티티가 있으면 그 엔티티를 선택) 외래키 속성으로 매핑한다. 두 엔티티 탑입이 모두 완전하게 참여하는 경우, 두 테이블을 하나의 테이블로 병합할 수 있다.
- STEP 4
 - 1:N 이진 관계는 N-side 릴레이션의 외래키 속성으로 매핑하며, 1-side의 주 키를 참조하도록 한다.
- STEP 5
 - N:M 이진 관계는 별도의 릴레이션 (이를 관계 릴레이션이라고 부름)으로 생성하고, 관계에 참여하는 두 릴레이션의 기본 키를 각각 참조하는 외래키로 애트리뷰트를 구성한다. 이 때 두 외래키가 관계 릴레이션의 기본키를 형성한다.
- STEP 6
 - 다중값 애트리뷰트는 키를 포함하는 릴레이션으로 매핑된다.
- STEP 7
 - n 차 관계는 관계에 참여하는 n 개의 릴레이션의 키들로 구성되는 관계 릴레이션으로 매핑된다. 관계 릴레이션의 애트리뷰트들은 참여 릴레이션의 주 키를 참조하는 외래키들과 관계 속성(들)으로 구성된다.

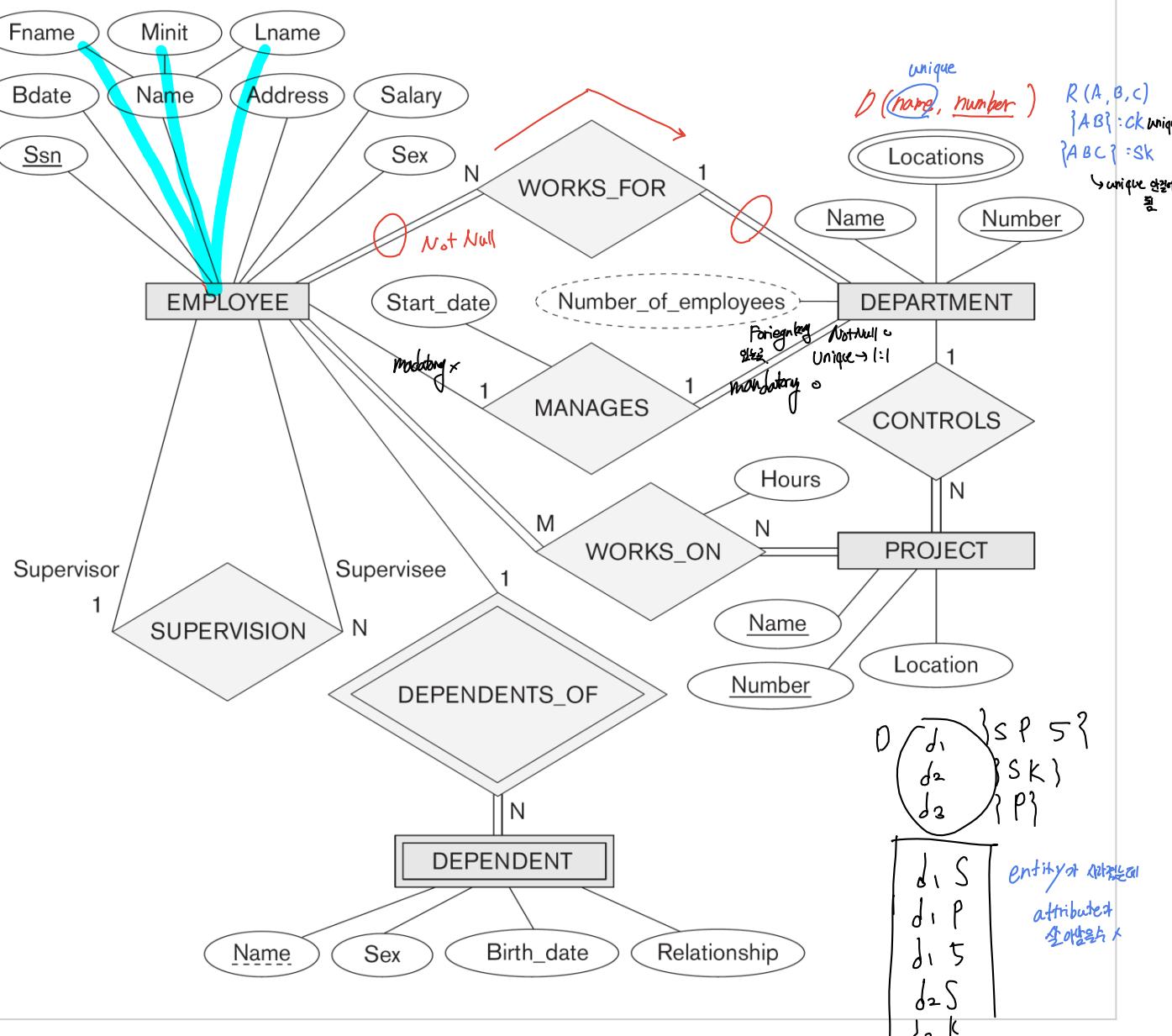


ER Model : Company

Figure 9.1

The ER conceptual schema diagram for the COMPANY database.

관계형 DB에서는 오직 Relation 밖에 없음.



STEP 1: Mapping of Regular Entity Types

- ER 스키마의 각 정규(강한) 엔티티 타입 E에 대해 E의 모든 simple attribute(또는 composite attribute의 단순 구성 요소)을 포함하는 관계 R을 작성한다.
 - E의 key attribute 중 하나를 R의 primary key 키로 선택한다. 다른 key attribute는 unique(candidate key)로 선언하여야 한다.
 - Figure 9.1에서 STEP 1을 적용해 볼 것!
 - Figure 9.3(a)에서 확인
- 연관 key가 Composite인 경우, 선택한 key의 simple attributes의 set이 Rel primary key를 구성.

Figure 9.3

Illustration of some mapping steps.
a. Entity relations after step 1.

b. Additional weak entity relation after step 2.

c. Relationship relation after step 5.

d. Relation representing multivalued attribute after step 6.

(a) EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary
-------	-------	-------	-----	-------	---------	-----	--------

DEPARTMENT

Dname	Dnumber
-------	---------

PROJECT

Pname	Pnumber	Plocation
-------	---------	-----------

(b) DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
------	----------------	-----	-------	--------------

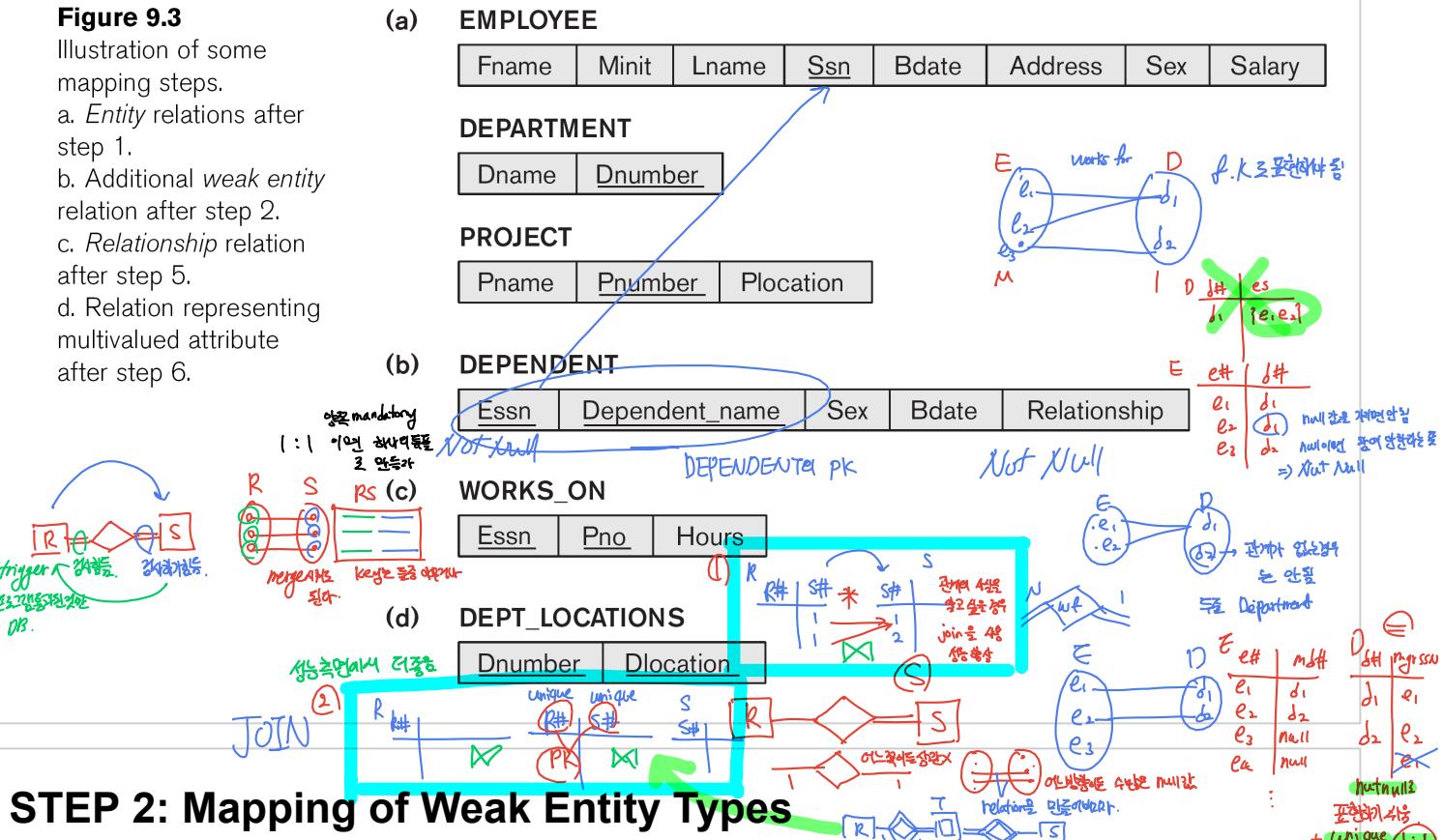
DEPENDENT PK

WORKS_ON

Essn	Pno	Hours
------	-----	-------

(d) DEPT_LOCATIONS

Dnumber	Dlocation
---------	-----------



STEP 2: Mapping of Weak Entity Types

- owner entity type E와 연관된 weak entity type W에 대해, W의 모든 simple attribute를 포함하는 관계 R을 만든다.
- 모든 owner entity type의 primary key attribute를 R의 foreign key로 포함한다.
- R의 primary key는 owner entity type의 primary key와 weak entity type W의 partial key(있다면)를 포함한다.
- Figure 9.1에서 STEP 2를 적용해 볼 것!
- Figure 9.3(b)에서 확인
- Weak entity는 Strong Entity에 존재 종속적(existence dependency)이기 때문에 Foreign Key Option에서 (ON UPDATE, ON DELETE) CASCADE option을 선택하는 것이 일반적이다.

STEP 3: Mapping of Binary 1:1 Relationship Types

- ER 스키마에서 각 이진 1:1 관계 타입 R에 대해, R에 참여하는 엔티티 타입에 해당하는 relation S와 T를 찾는다.
- Three possible approaches:
 - Foreign Key approach: relation S를 선택하고 S의 foreign key로 T의 primary key를 포함한다. (S와 T가 전체 참여(total participation)하는 entity type을 S로 선택하는 것이 좋다. Why?)
 - Example (see Figure 9.2): DEPARTMENT는 1:1 relationship MANAGES에 전체 참여이다. (Fig. 9.1)

다른 relation T의 PK를 S의 FK로 포함

- DEPARTMENT의 Mgr_SSN이 EMPLOYEE를 참조하는 foreign key가 된다. (NULL에 대한 Option은? Allowed?) EMPLOYEE의 PK인 Ssn을 DEPARTMENT의 fk인 Mgr_ssn으로 가진다 (Mgr_ssn이 NULL을 허용)
 - MANAGES의 애트리뷰트는 DEPARTMENT의 애트리뷰트가 된다.
 - Merged relation option: S와 T를 하나의 relation R로 병합한다. (S와 T의 참여가 모두 전체 참여(total participation)인 경우에 가능한 선택사항이다. Why?) entities가 total participation인 경우 적절
 - R의 primary key, candidate key?
 - Relationship relation option: 1:1 relationship을 나타내는 새로운 relation R을 만들 수 있다.
 - extra JOIN 비용이 단점이다.
 - R의 primary key, candidate key?
 - Figure 9.1에서 STEP 3을 적용해 볼 것!
 - Figure 9.2에서 확인
- relation S & T의 PKs를 cross-referencing 하기 위해 새로운 relation U를 생성
 U의 PK는 S & T로 부터의 PKs 중 하나
 또 다른 PK는 U의 unique key
 예전 join 필요. → 추가장을 얻기 위해 수정
 EMPLOYEE Frame, Mint Lane Ssn
 DEPARTMENT Name, Manager Mgr_ssn Mgr_start_date
 T

STEP 4: Mapping of Binary 1:N Relationship Types

- 1:N relationship type에서 N-side entity type에 해당하는 relation S를 찾는다.
 - 1:N relationship에 참여하는 1-side entity type에 해당하는 relation T의 primary key를 S의 foreign key로 포함한다.
 - 1:N relationship type의 simple attribute를 S의 attribute로 포함시킨다.
 - Examples (Figures 9.1): 1:N relationship types - WORKS_FOR, CONTROLS, and SUPERVISION에 대해 STEP 4를 적용해 볼 것!
 - Figure 9.2에서 확인해 볼 것!
 - N-side entity type이 전체 참여인 경우, 취해야 할 행동은?
 - 1-side entity type이 전체 참여인 경우, 취해야 할 행동은?
 - ON DELETE시에 취해야 할 Foreign Key Option은?
 - Relationship relation option: 1:N relationship을 나타내는 새로운 relation R을 만들 수 있다.
 - extra JOIN 비용이 단점이다.
 - R의 primary key, candidate key?
- Relation S: The EMPLOYEE entity type (N-side)
 Relation T: The DEPARTMENT entity type (1-side)
 EMPLOYEE가 DEPARTMENT의 Number(PK)를 참조하기 위해
 DeptNo(FK)를 가짐

STEP 5: Mapping of Binary M:N Relationship Types

- M:N relationship type R에 해당하는 새로운 relation S를 만든다.
 - R에 참여하는 두 entity type에 해당하는 두 relation의 primary key들을 S에 foreign key로 포함한다.
 - S의 primary key는 두 foreign key의 조합으로 구성된다.
 - M:N relationship type의 simple attribute를 S의 attribute로 포함시킨다.
 - Relationship instance는 참여하는 Entity에 존재 종속적(existence dependency)이기 때문에 Foreign Key Option에서 (ON UPDATE, ON DELETE) CASCADE option을 선택하는 것이 일반적이다.
 - Example: M:N relationship type WORKS_ON (Figure 9.1)에 대해 STEP 5를 적용해 보자.
 - Figure 9.2에서 확인
 - S의 Foreign Key는 NULL 값이 허용되는가?
- 양쪽의 foreign key 양쪽
 그대로 일관성 자체를
 entity 보존.
- Key들 간으로써 구성된것으로
 NULL값 허용 X
- $\text{FKs}(\text{Esn and Pno}) \rightarrow \text{EMPLOYEE \& PROJECT relations}$
 으로 부터 가져온.
- Pk: ?Esn, Pno?
 One attribute : Hours (WORKS_ON의 Hours attribute)

STEP 6: Mapping of Multivalued attributes

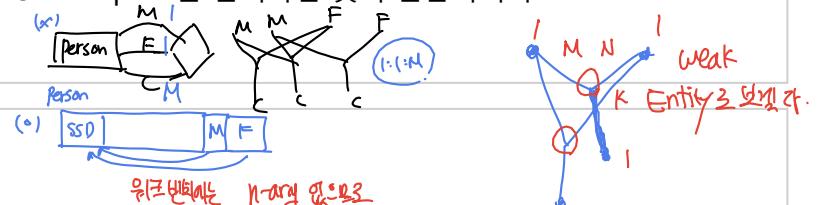
- relation S(primary key: K)의 multivalued attribute A에 대해 새로운 relation R을 만든다.
 - R에 K를 포함한다. R의 K는 S를 참조하는 foreign key가 된다.
 - K와 A의 조합이 R의 primary key가 된다.
 - Attribute는 소속된 Entity에 존재 종속적(existence dependency)이기 때문에 Foreign Key Option에서 (ON UPDATE, ON DELETE) CASCADE option을 선택하는 것이 일반적이다.
 - Example (Figure 9.3(d)): relation DEPT_LOCATIONS이 생성되었다.
- relation R은
 A에 해당하는 attribute
 A를 attribute로서 가지는 entity type의
 PK를 FK로, ?A, K?를 PK로 가짐.

- DEPT_LOCATIONS의 foreign key는?
- DEPT_LOCATIONS의 primary key는?

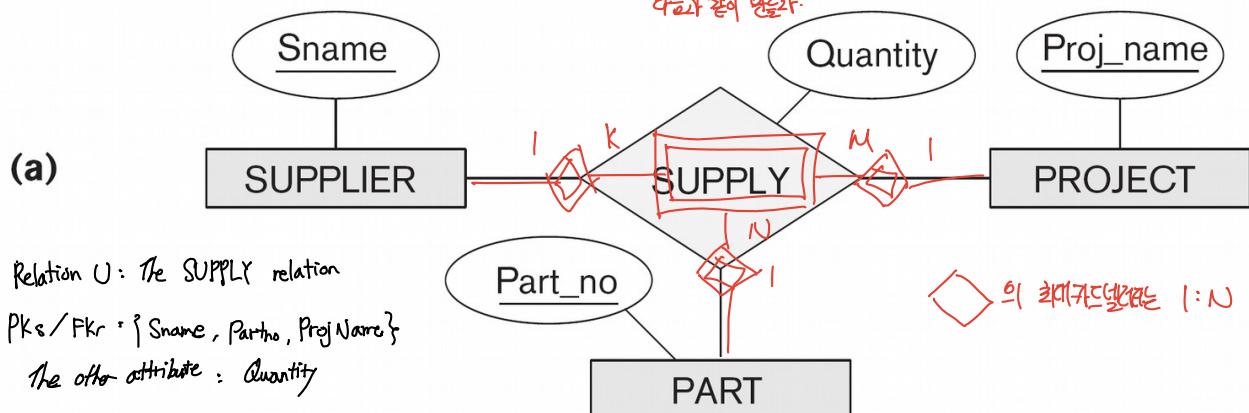
A: DEPARTMENT entity type Locations로부터 가져온 Dlocation
 FK: DEPARTMENT entity type의 PK를 가져온 Dnumber
 PK: ?Dlocation, Dnumber?

STEP 7: Mapping of N-ary Relationship Types

- n>2인 n-ary relationship type R에 대해 새로운 relationship relation S을 만든다.
- 참여하는 모든 entity type의 primary key는 S의 foreign key로 포함시킨다.
- 참여하는 모든 entity type의 primary key들의 조합으로 S의 primary key를 구성한다.
- N-ary relationship type의 attribute들을 S의 attribute로 포함시킨다.
- Relationship instance는 참여하는 Entity에 존재 종속적(existence dependency)이기 때문에 Foreign Key Option에서 (ON UPDATE, ON DELETE) CASCADE option을 선택하는 것이 일반적이다.
- 다음 슬라이드에 STEP 7을 적용해 보자.



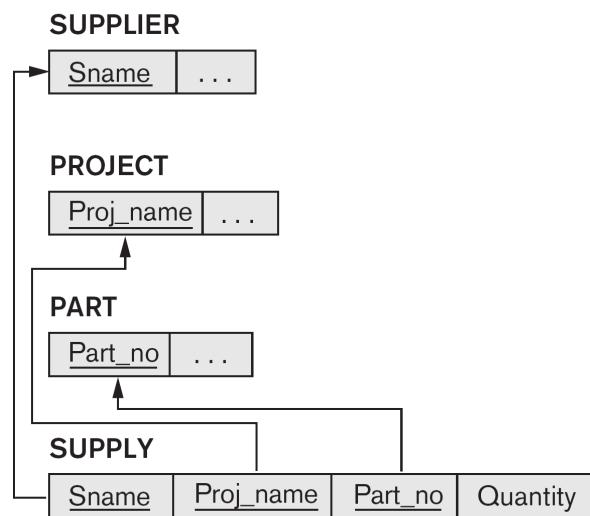
Ternary Relationship: Supply



Ternary Relationship: Supply

Figure 9.4

Mapping the n -ary relationship type SUPPLY from Figure 7.17(a).

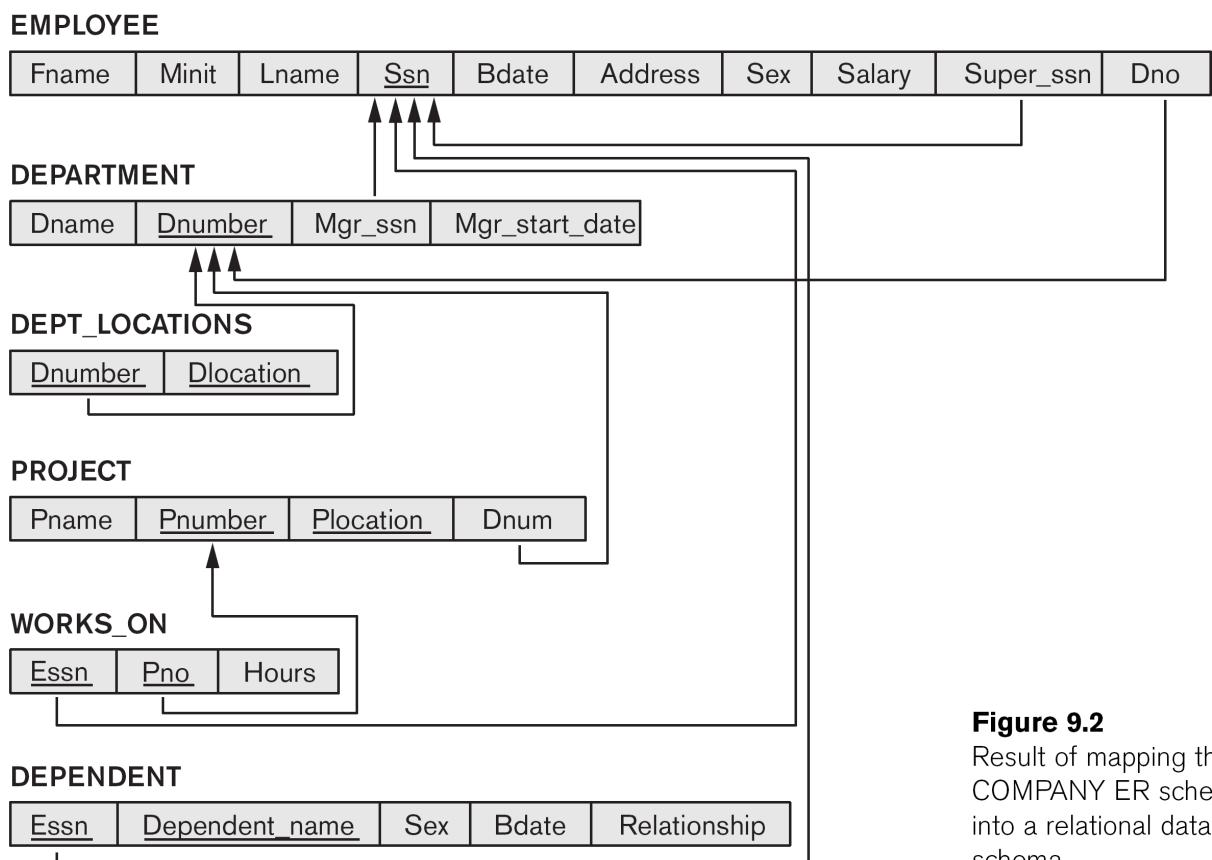


Correspondence between ER and Relational Model

Table 9.1 Correspondence between ER and Relational Models

ER MODEL	RELATIONAL MODEL
Entity type	<i>Entity</i> relation
1:1 or 1:N relationship type	Foreign key (or <i>relationship</i> relation)
M:N relationship type	<i>Relationship</i> relation and <i>two</i> foreign keys
<i>n</i> -ary relationship type	<i>Relationship</i> relation and <i>n</i> foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key

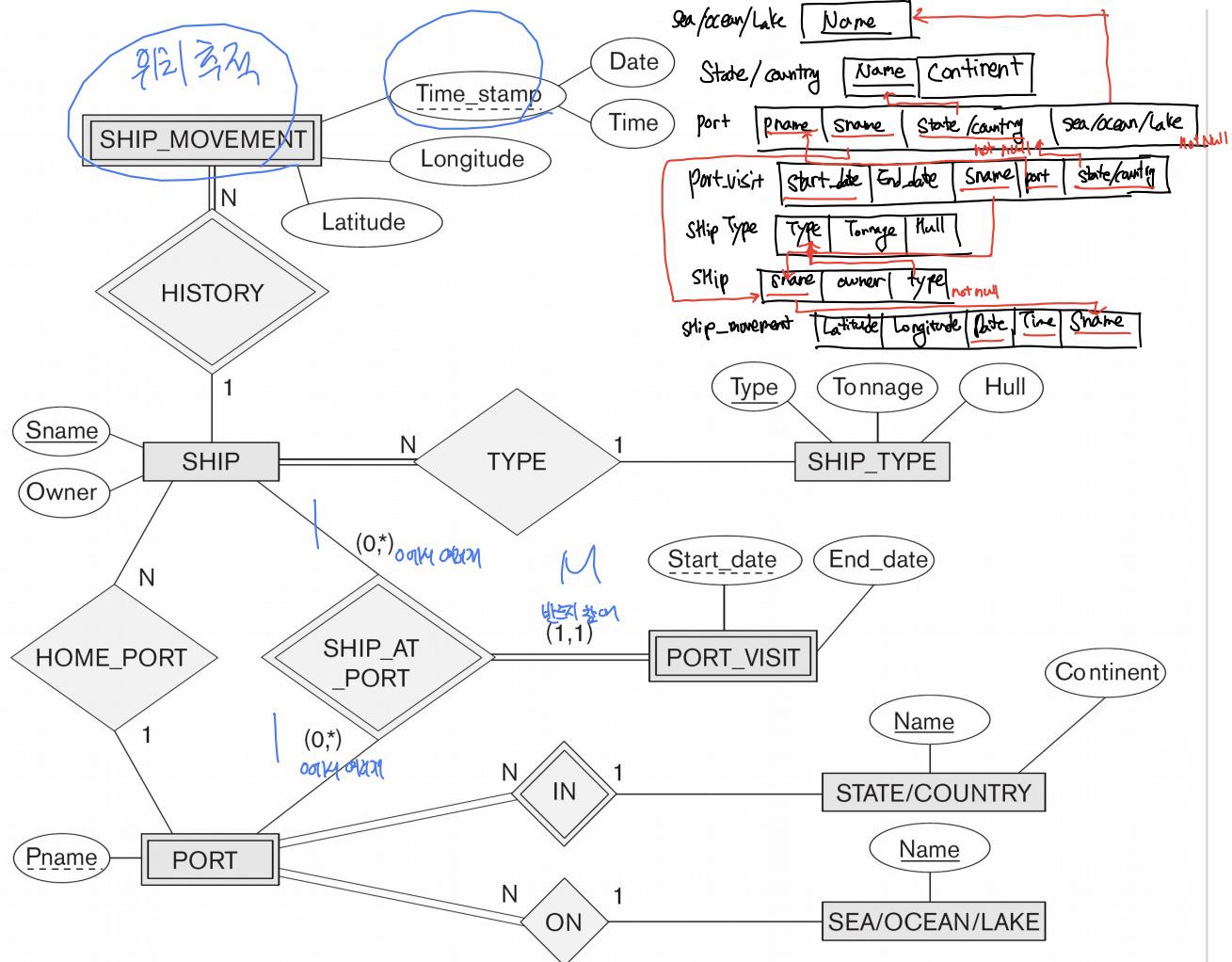
ER Model : Company

**Figure 9.2**

Result of mapping the COMPANY ER schema into a relational database schema.

In-Class Exercise

- Draw a relational schema diagram from this ER diagram by applying STEP 0 through 7.

**Figure 9.8**

An ER schema for a SHIP_TRACKING database.

EER-Relation Mapping

EER을 relation mapping x

STEP 8: Mapping of Specialization or Generalization

- Option 8A: 다수의 테이블(Multiple relations)—superclass 와 subclasses 모두 테이블로 변환
 - For any specialization (total or partial, disjoint or overlapping)
- Option 8B: 다수의 테이블 (Multiple relations)—subclass 만 테이블로 변환 Drill down
 - Subclasses are total
 - Specialization has disjointedness constraint
- Option 8C: 하나의 type attribute를 포함하는 하나의 테이블로 변환
 - Type (or discriminating) attribute가 subclass를 결정한다.
- Option 8D: 여러 개의 type attribute를 포함하는 하나의 테이블로 변환
 - Subclasses are overlapping
 - Will also work for a disjoint specialization

수퍼클래스
테이블 합계
→ Flags을 드롭시켜 주고 올려놓기 Roll-up

Regular (strong) entity

- ER Schema의 일반적인 Entity의 모든 attribute는 relation R을 형성할 때 포함된다.
- 복합 attribute의 경우 각각의 요소를 Simple attribute로 포함한다.
- name과 |name은 포함하지만, name이라는 복합 attribute는 추가하지 않는다.
- E가 가진 attribute 중 하나를 Primary key로 설정한다.
- 만약 선택된 attribute가 복합 attribute라면, 이를 구성하는 Simple attribute Set이 primary key가 된다.

Weak entity

- weak entity가 포함하는 모든 attribute를 Relation Key에 추가한다.
- weak entity의 partial key or owner entity의 primary key를除 Relation Key PK가 된다.
- 이 때 weak entity or Owner entity는 identifying relationship을 형성한다.
- Referential triggered action
 - owner or weak entity는 CASCADE optional 적용, UPDATE / DELETE operation
 - ① 대해서 CASCADE 하지 적용된다

Binary 1:1 relationship

- relation S와 T가 1:1 관계를 형성할 때의 경우이다.
- 양쪽 모두 Partial participation인 경우, S의 PK가 T의 FK가 될 수도 있고, 반대로 가능하다.
- 한 쪽이 total participation인 경우 partial participation인 relation의 PK를 다른 쪽의 FK로 추가한다.

Binary M:N relationship

relation S와 W가 M:N 관계를 형성할 때의 경우이다.

PK or FK로는 M:N 관계를 표현할 수 없다.

N-ary relationship type

N개의 entity에 대한 관계를 나타내는 relation S를 정의한다.

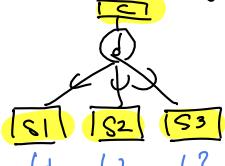
각 entity의 pk를 S의 FK로 설정한다.

S의 PK는 모든 PK의 합집합으로 구성된다.

단, N개의 entity 중 cardinality가 1인 entity의 PK는 S의 PK를 구성하는 PK set의 원소로 포함되어야 한다.

option 8A

- total / partial, disjoint / overlapping 모든 경우에 사용할 수 있는 방법



- Create a relation L for C

부모 entity에 대해서는 자신의 attribute와 key만을 포함한다.

$$\text{Attrs}(L) = \{k, a_1, \dots, a_n\} \text{ and } \text{PK}(L) = k$$

For each subclass Si, 1 ≤ i ≤ m, create a relation Li

- 각각 entity의 경우 자신의 attribute와 key + 부모의 key attribute를 갖는다.

$$\text{Attrs}(L_i) = \{k\} \cup \{\text{attributes of } S_i\}$$

$$\text{PK}(L_i) = k$$

- inclusion dependency

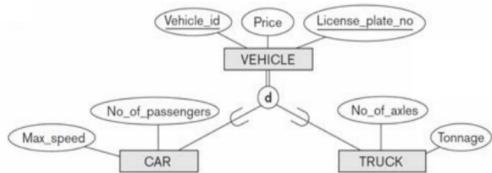
• 부모의 key 없이 자신의 key가 존재할 수 없다.

• $\pi: [key] \cap$

$$\pi(L_i) \subseteq \pi(L)$$

$$\pi(L_i) \subseteq \pi(L)$$

Option 8B



CAR

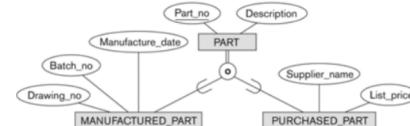
Vehicle_id	License_plate_no	Price	Max_speed	No_of_passengers
------------	------------------	-------	-----------	------------------

TRUCK

Vehicle_id	License_plate_no	Price	No_of_axles	Tonnage
------------	------------------	-------	-------------	---------

- 부모 relation은 생성하지 않음
- disjoint이고 total인 경우 잘 적용된다.
 - 만약 total이 아닌 경우(CAR, TRUCK이 아닌 VEHICLE이 있는 경우) : 정보를 표현할 수 없음
 - 만약 disjoint이 아닌 경우(CAR 이면서 TRUCK 인 경우) : 동일한 정보가 여러 subtype에 중복되어서 나오는 문제가 있음
 - 따라서 VEHICLE 전체의 구조를 보고 싶으면 OUTERJOIN(구조가 서로 다른 table을 JOIN) 또는 FULL OUTERJOIN을 CAR와 TRUCK에 적용해야 한다.
- Create a relation Li for each subclass Si, $1 \leq i \leq m$,
 - Attrs(Li) = { attributes of Si } \cup {k, a1,..., an}
 - PK(Li) = k

Option 8D

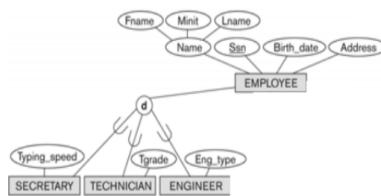


PART

Part_no	Description	Mflag	Drawing_no	Manufacture_date	Batch_no	Pflag	Supplier_name	List_price
---------	-------------	-------	------------	------------------	----------	-------	---------------	------------

- subclass들이 overlap되는 경우 사용할 수 있는 방식
- Create a single relation schema L
 - Attrs(L) = {k,a1,...,an} \cup {attributes of S1} $\cup \dots \cup$ {attributes of Sm} \cup {t1, t2, ..., tm}
 - PK(L) = k
 - each ti, $1 \leq i \leq m$, is a Boolean attribute indicating whether a tuple belongs to subclass Si,
- subclass의 attribute가 많으면 추천하지 않는 방식
- subclass의 attribute가 별로 없고, JOIN이 자주 사용되는 경우.(저장공간과 성능 사이의 상충 관계)

Option 8C



EMPLOYEE

Ssn	Fname	Minit	Lname	Birth_date	Address	Job_type	Typing_speed	Tgrade	Eng_type
-----	-------	-------	-------	------------	---------	----------	--------------	--------	----------

EMPLOYEE

EMPLOYEE

Ssn

Fname

Minit

Lname

Birth_date

Address

Job_type

Typing_speed

Tgrade

Eng_type

- 부 모 relation만 생성한다.

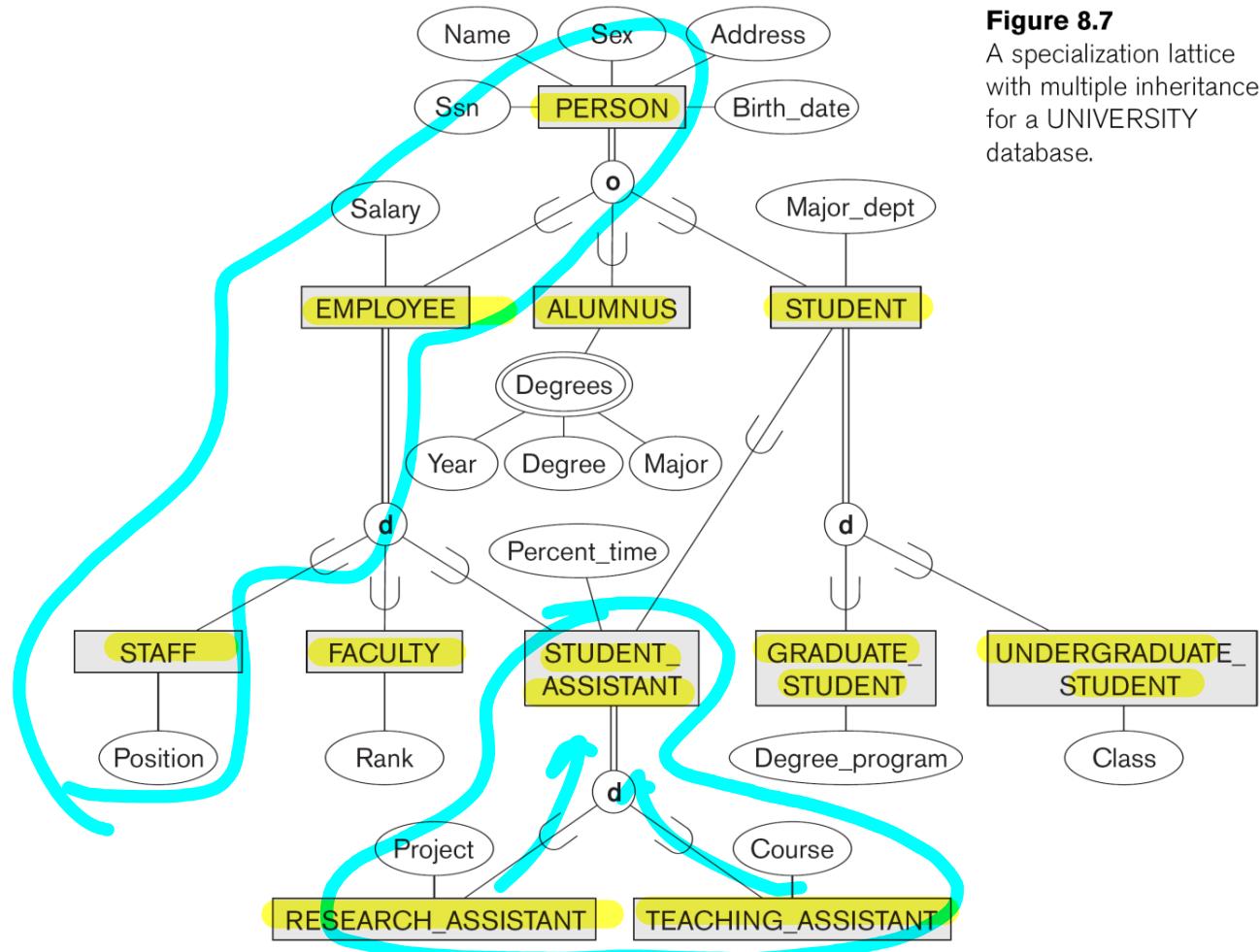
- Create a single relation L

- Attrs(L) = {k,a1,...,an} \cup {attributes of S1} $\cup \dots \cup$ {attributes of Sm} $\cup \{t\}$
- t = type (or discriminating) attribute that indicates the subclass to which each tuple belongs, if any.
- PK(L) = k

- t를 추가하기 때문에 disjoint인 상황에서만 사용할 수 있는 방법

- subtype 중 하나에만 속하기 때문에 굉장히 많은 NULL 값이 생성될 수 있는 방법

Specialization / Generalization Lattice Example (UNIVERSITY)

**Figure 8.7**

A specialization lattice with multiple inheritance for a UNIVERSITY database.

Mapping of Shared Subclasses (Multiple Inheritance)

- STEP 8 선택사항의 어떤 것도 shared subclass에 적용 가능

PERSON

<u>Ssn</u>	Name	Birth_date	Sex	Address
------------	------	------------	-----	---------

EMPLOYEE

<u>Ssn</u>	Salary	<u>Employee_type</u>	Position	Rank	Percent_time	<u>Ra_flag</u>	<u>Ta_flag</u>	Project	Course
------------	--------	----------------------	----------	------	--------------	----------------	----------------	---------	--------

2017.8.13. 2교시

ALUMNUS

ALUMNUS_DEGREES

<u>Ssn</u>	<u>Ssn</u>	<u>Year</u>	<u>Degree</u>	<u>Major</u>
------------	------------	-------------	---------------	--------------

STUDENT

<u>Ssn</u>	Major_dept	Grad_flag	Undergrad_flag	Degree_program	Class	Student_assist_flag
------------	------------	-----------	----------------	----------------	-------	---------------------

Figure 8.6

Mapping the EER specialization lattice in Figure 7.26 using multiple options.

STEP 9: Mapping of Union Types

- Superclass들은 각각 다른 key를 가진다.
- Subclass의 key를 새로운 대체 key attribute로 도입한다.
 - Surrogate key

Two categories (UNION types): OWNER, REGISTERED_VEHICLE

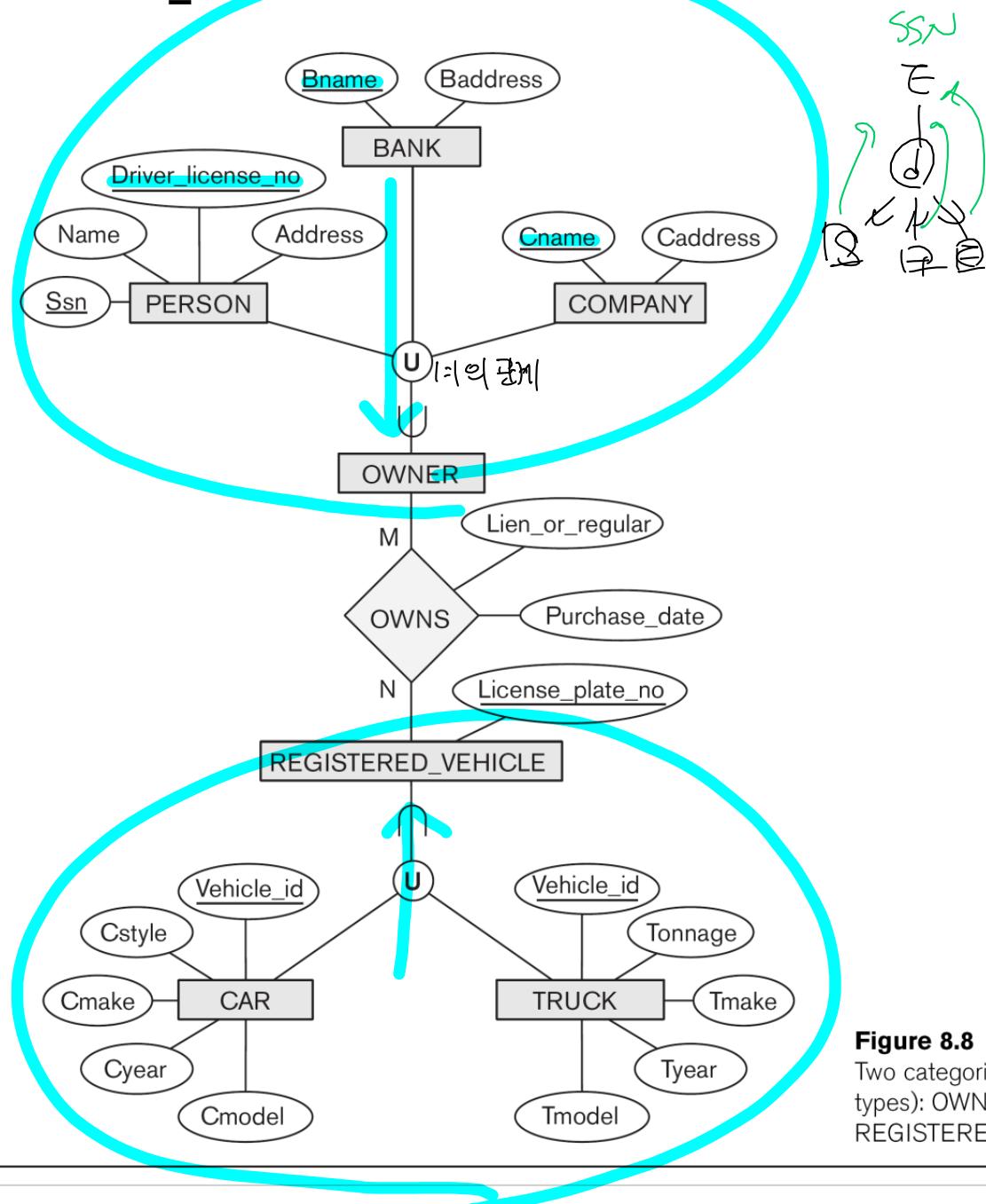
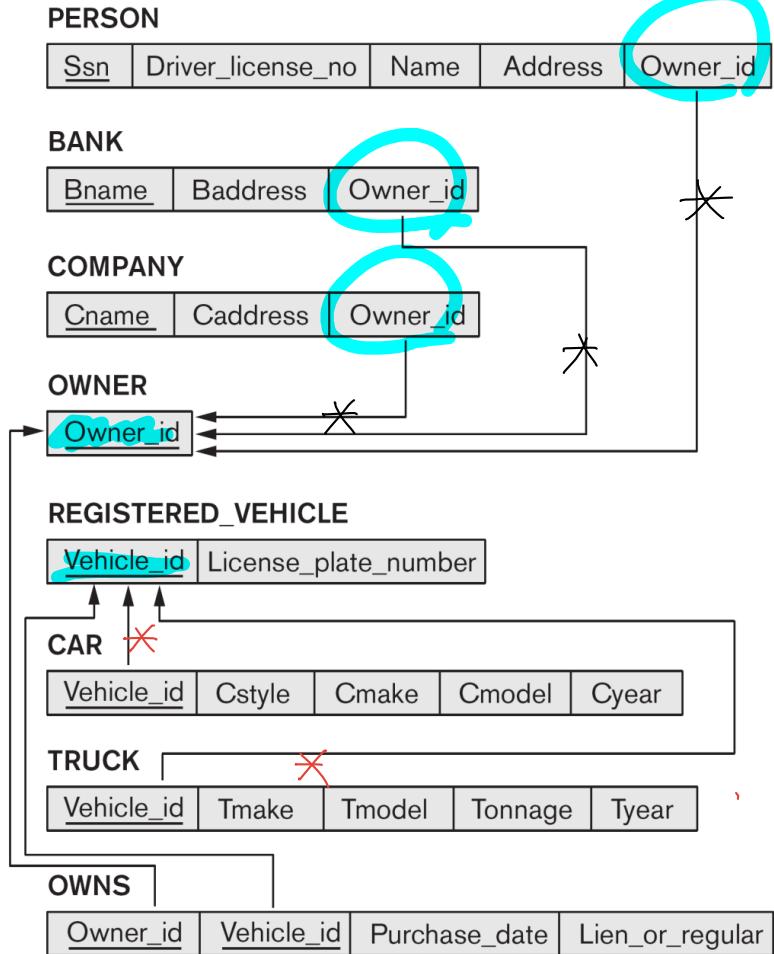


Figure 8.8
Two categories (union types): OWNER and REGISTERED_VEHICLE.

Two categories (UNION types): OWNER, REGISTERED_VEHICLE

Figure 8.7

Mapping the EER categories (union types) in Figure 7.26 to relations.



13 장. 관계 데이터베이스의 함수적 종속성과 정규화 기본 이론

Contents

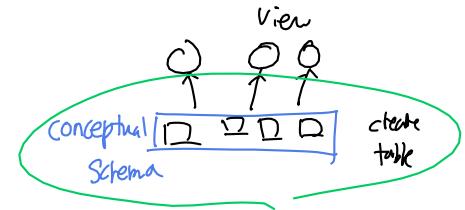
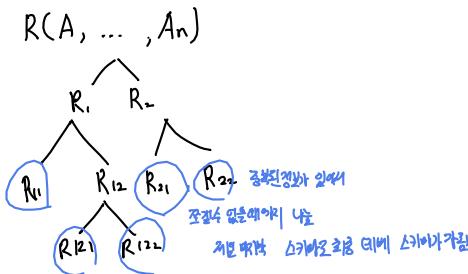
- 릴레이션 스키마를 설계하는 몇 가지 개략적인 지침
- 함수적 종속성 (functional dependencies, FDs)**
- 기본 키를 기반으로 한 정규형
- 제 2 정규형과 제 3 정규형의 일반적인 정의
- BCNF (Boyce-Codd Normal Form)

Relational Database design의 가장 중요한 목표 2개

- Information preservation
- minimum redundancy

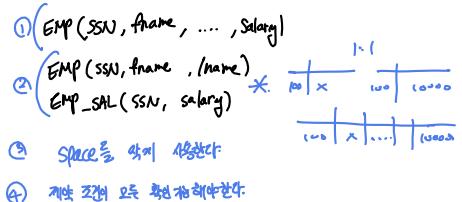
릴레이션 스키마를 설계하는 몇 가지 개략적인 지침

- 관계형 데이터베이스 설계란?
 - “좋은” 릴레이션 스키마를 생성하기 위하여 **애트리뷰트들을 그루핑하는 과정**
 - “좋은” 릴레이션에 대한 기준은?
- 릴레이션 스키마의 두 가지 수준
 - 논리적인 “사용자 뷰(user view)” 수준
 - 저장이 되는 “기본 릴레이션(base relation)” 수준
- 데이터베이스 설계는 주로 기본 릴레이션을 대상으로 함
- 내용
 - 여기서는 좋은 릴레이션 설계에 관한 개괄적인 지침을 논한 후, 함수적 종속성과 정규형 개념에 관해 논의함
 - 1NF (제 1 정규형)
 - 2NF (제 2 정규형)
 - 3NF (제 3 정규형)
 - BCNF (Boyce-Codd 정규형)



one information at one place

No Redundancy



릴레이션 애트리뷰트들의 의미

설계자는 데이터베이스에 포함될 전체 애트리뷰트들을 대상으로 실세계에서 어떤 연관성이 있는 애트리뷰트들을 묶어서 하나의 릴레이션 스키마를 만들어 나간다.

- 여러 엔티티(EMPLOYEE, DEPARTMENT, PROJECT)의 애트리뷰트들이 하나의 릴레이션에 혼합되면 안된다.
 ^{불이되어 있어야 함.}
- 다른 엔티티를 참조하기 위해서는 외래키만을 사용해야 한다.

예 :

foreign keys 만에 서로 다른 entity들을
참조할 수 있어야 한다

- 잘 설계된 경우 (Fig 15.1)
- 데이터베이스 인스턴스 (Fig 15.2)
- 여러 엔티티의 속성들이 하나의 릴레이션에 혼합되어 문제 발생 (Fig 15.3)

Figure 15.1

A simplified COMPANY relational database schema.

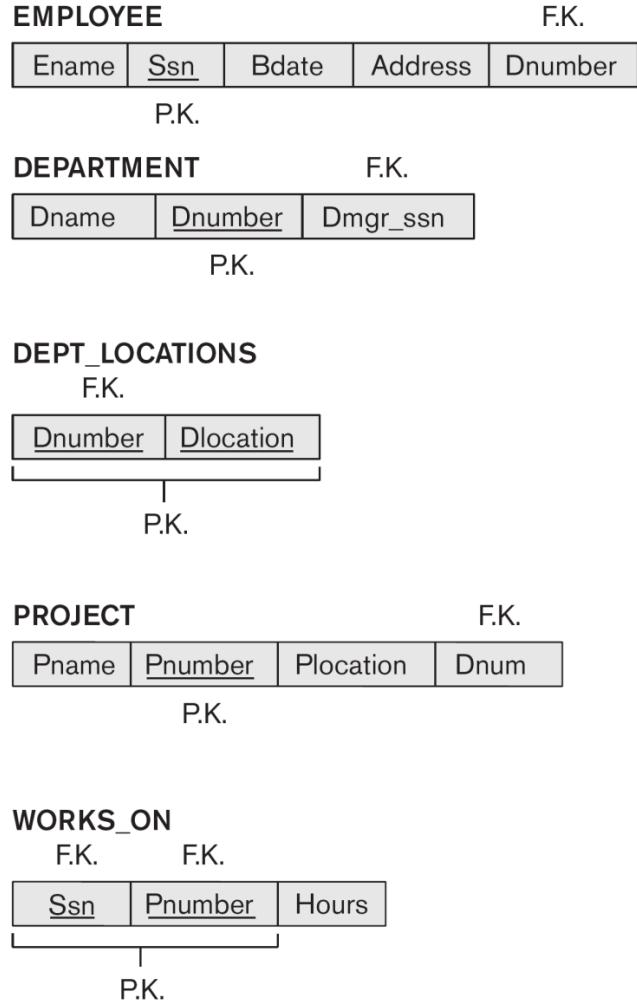


Figure 15.2

Sample database state for the relational database schema in Figure 15.1.

EMPLOYEE

Ename	Ssn	Bdate	Address	Dnumber
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4
Wallace, Jennifer S.	987654321	1941-06-20	291Berry, Bellaire, TX	4
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1

DEPARTMENT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Ssn	Pnumber	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

Figure 15.3

Two relation schemas suffering from update anomalies. (a) EMP_DEPT and (b) EMP_PROJ.

(a)

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn

(b)

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation



Insert anomaly

1. Employee 가 할당되지 않은 project 정보는 Insert 할 수 없다.
Insert 하기 되면 Ssn의 사이즈로, Entity Constraint 위반

2. Project를 부여받지 않은 employee 정보는 Insert 할 수 있다.
Insert 하기 되면 Pnumber가 Null=1로, Entity constraint 위반

Delete Anomaly \Rightarrow 2개의 다른 정보들의 함께
제거되어 발생하는 문제

- Project가 삭제되면 해당 project에 참여하고 있는 모든 employees 정보 또한 삭제될 것이다.
- 어떤 employee가 어떤 project에 참여하는 유일한 직원인 경우, 해당 employee 정보를 삭제한 그 project 정보 또한 삭제되게 된다.

튜플에서 중복된 정보와 이상(anomaly)

하나의 릴레이션에 하나 이상의 엔티티의 애트리뷰트들을 혼합하는 것은 여러 가지 문제를 일으킨다. (Fig 15.4.)

- 정보가 중복 저장되며, 저장 공간을 낭비하게 된다 (Fig 15.2의 EMPLOYEE와 DEPARTMENT \leftrightarrow 15.3 및 15.4의 EMP_DEPT 비교)
- 갱신 이상이 발생하게 된다; 동일한 정보를 한 릴레이션에는 변경하고, 나머지 릴레이션에서는 변경하지 않은 경우 어느 것이 정확한지 알 수 없게 된다.
- 갱신 이상의 종류
 - 삽입 이상 (insertion anomalies)
 - EMP_DEPT에 객체를 삽입할 때 부서가 정해지지 않은 직원이나 직원이 없는 부서를 insert 하는데 문제가 발생함
 - 삭제 이상 (deletion anomalies)
 - 부서의 마지막 직원을 삭제하면 부서 정보도 없어짐
 - 수정 이상 (modification anomalies)
 - 부서 정보를 변경하면 부서의 모든 직원 투플에서 동일하게 변경해야 함

project number 10의 name은

"computerization" $\&$ "Customer-Accounting"으로 변경하는 경우

project number 10에 종사하는 모든 employees를 update 해야 한다.

일관성의 원칙

abnormal 비정상적

anomaly 비정상

one information one place 原則

5번처럼 이름 Research → 4번만 복중임.

Redundancy

EMP_DEPT						
Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

EMP_PROJ						
Ssn	Pnumber	Hours	Ename	Pname	Plocation	
123456789	1	32.5	Smith, John B.	ProductX	Bellaire	
123456789	2	7.5	Smith, John B.	ProductY	Sugarland	
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston	
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire	
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland	
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland	
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston	
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford	
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston	
999887777	30	30.0	Zelaya, Alicia J.	Newbenefits	Stafford	
999887777	10	10.0	Zelaya, Alicia J.	Computerization	Stafford	
987987987	10	35.0	Jabbar, Ahmad V.	Computerization	Stafford	
987987987	30	5.0	Jabbar, Ahmad V.	Newbenefits	Stafford	
987654321	30	20.0	Wallace, Jennifer S.	Newbenefits	Stafford	
987654321	20	15.0	Wallace, Jennifer S.	Reorganization	Houston	
888665555	20	Null	Borg, James E.	Reorganization	Houston	

Figure 15.4

Sample states for EMP_DEPT and EMP_PROJ resulting from applying NATURAL JOIN to the relations in Figure 15.2. These may be stored as base relations for performance reasons.

튜플에서 널 값

Null

- 릴레이션의 튜플들이 널 값을 가지지 않도록 설계해야 함
 - 널 값은 저장 단계에서 공간을 낭비하게 되고 이미 값이 있으면 의미를 명확히 하기 어렵다.
 - 논리적 차원에서는 조인 연산들을 지정하기 힘들고 해석의 어려움
 - 애트리뷰트들의 의미를 이해하기 어려움
 - COUNT나 AVG와 같은 집단 함수들이 적용되었을 때 널 값의 해석이 모호함
 - 널 값은 다음과 같이 여러 가지로 해석이 가능함
 - 그 애트리뷰트가 이 튜플에는 적용되지 않는다. (존재 여부를 모른다)
 - 이 튜플에서 애트리뷰트의 값이 아직 알려져 있지 않다 (존재하지만 모른다).
 - 애트리뷰트 값이 알려져 있지만 DB에 기록되지는 않았다.
 - 모든 널 값을 동일하게 표현하면 널 값이 갖는 여러 의미를 훼손하게 된다.

튜플에서 널 값

- 널 값의 방지 기법 - 릴레이션의 분리

- 널 값이 많이 나타나는 애트리뷰트들은 별개의 릴레이션으로 분리함

1:1			
I	K	30	100

Employee(ssn, ename, age, office_no)

↳ 한 번에 여러 개의 열(행)으로 office_no 있는 업무를 개별적 office 할당된 4값

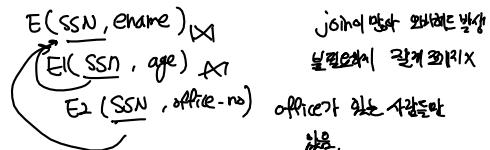
분리하는 경우 1:1 관계

대부분의 Employee는 개별 office_no를 가지고 있지 않다. 위의 Employee를 아래와 같이 두 개로 분리하는 것이 좋다.

- Employee(ssn, ename, age)

→ 모든 행은 의미가 많아서 좋겠다.

Emp_Office(ssn, office_no)



가짜 투플 (spurious tuples)

- 관계 데이터베이스 설계를 잘못하게 되면, 조인 연산들이 틀린 결과를 생성할 수 있다.
- 조인 연산의 결과가 올바르기 위해서는, 릴레이션들이 “무손실 조인(lossless join)” 조건을 만족하도록 설계되어야 한다.

(a)

EMP_LOCS

Ename	Plocation

P.K.



EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation

P.K.

join attribute
Plocation.

아예 합계가 아니요
이상한 내용들이 들어온다.

Figure 15.5

Particularly poor design for the EMP_PROJ relation in Figure 15.3(b). (a) The two relation schemas EMP_LOCS and EMP_PROJ1. (b) The result of projecting the extension of EMP_PROJ from Figure 15.4 onto the relations EMP_LOCS and EMP_PROJ1.

(b)

EMP_LOCS



설계와 굉장히 헛된 편

EMP_PROJ1

Ename	Plocation
Smith, John B.	Bellaire
Smith, John B.	Sugarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland
Wong, Franklin T.	Sugarland
Wong, Franklin T.	Houston
Wong, Franklin T.	Stafford
Zelaya, Alicia J.	Stafford
Jabbar, Ahmad V.	Stafford
Wallace, Jennifer S.	Stafford
Wallace, Jennifer S.	Houston
Borg, James E.	Houston

Ssn	Pnumber	Hours	Pname	Plocation
123456789	1	32.5	ProductX	Bellaire
123456789	2	7.5	ProductY	Sugarland
666884444	3	40.0	ProductZ	Houston
453453453	1	20.0	ProductX	Bellaire
453453453	2	20.0	ProductY	Sugarland
333445555	2	10.0	ProductY	Sugarland
333445555	3	10.0	ProductZ	Houston
333445555	10	10.0	Computerization	Stafford
333445555	20	10.0	Reorganization	Houston
999887777	30	30.0	Newbenefits	Stafford
999887777	10	10.0	Computerization	Stafford
987987987	10	35.0	Computerization	Stafford
987987987	30	5.0	Newbenefits	Stafford
987654321	30	20.0	Newbenefits	Stafford
987654321	20	15.0	Reorganization	Houston
888665555	20	NULL	Reorganization	Houston

Ssn	Pnumber	Hours	Pname	Plocation	Ename
123456789	1	32.5	ProductX	Bellaire	Smith, John B.
* 123456789	1	32.5	ProductX	Bellaire	English, Joyce A.
123456789	2	7.5	ProductY	Sugarland	Smith, John B.
* 123456789	2	7.5	ProductY	Sugarland	English, Joyce A.
* 123456789	2	7.5	ProductY	Sugarland	Wong, Franklin T.
666884444	3	40.0	ProductZ	Houston	Narayan, Ramesh K.
* 666884444	3	40.0	ProductZ	Houston	Wong, Franklin T.
* 453453453	1	20.0	ProductX	Bellaire	Smith, John B.
453453453	1	20.0	ProductX	Bellaire	English, Joyce A.
* 453453453	2	20.0	ProductY	Sugarland	Smith, John B.
453453453	2	20.0	ProductY	Sugarland	English, Joyce A.
* 453453453	2	20.0	ProductY	Sugarland	Wong, Franklin T.
* 333445555	2	10.0	ProductY	Sugarland	Smith, John B.
* 333445555	2	10.0	ProductY	Sugarland	English, Joyce A.
333445555	2	10.0	ProductY	Sugarland	Wong, Franklin T.
* 333445555	3	10.0	ProductZ	Houston	Narayan, Ramesh K.
333445555	3	10.0	ProductZ	Houston	Wong, Franklin T.
333445555	10	10.0	Computerization	Stafford	Wong, Franklin T.
* 333445555	20	10.0	Reorganization	Houston	Narayan, Ramesh K.
333445555	20	10.0	Reorganization	Houston	Wong, Franklin T.

*
*
*

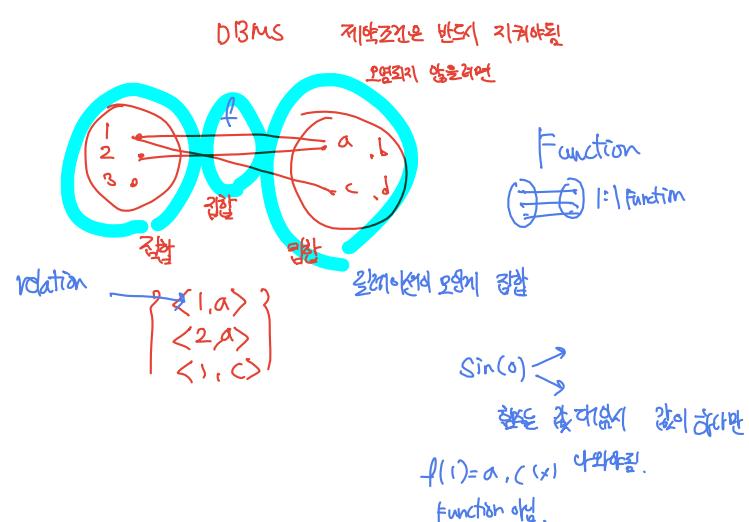
Figure 15.6

Result of applying NATURAL JOIN to the tuples above the dashed lines in EMP_PROJ1 and EMP_LOCS of Figure 15.5. Generated spurious tuples are marked by asterisks.

함수적 종속성

key or candidate key PK(X)

- 함수적 종속성(FD: functional dependency)은 좋은 릴레이션 설계의 정형적 기준으로 사용된다.
- FD와 키는 릴레이션의 정규형을 정의하기 위해 사용된다.
- FD는 데이터 애트리뷰트들의 의미와 애트리뷰트들 간의 상호 관계로부터 유도되는 제약조건(constraints)의 일종이다.
- 구성
 - 함수의 종속성(functional dependency)의 정의
 - 함수적 종속성의 추론 규칙
 - 함수적 종속성 집합의 동등성
 - 함수적 종속성의 최소집합



key는 relation의 normal form을 정하는게 사용

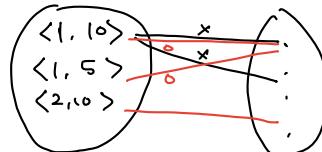
X의 value와 Y의 "unique" value를 결합하면

"attribute X의 값은 attribute Y의 값을 functionally determines"

함수적 종속성의 정의

{SSN, Pnumber}

{SSN, Pnumber}



$X \rightarrow Y$

Pnumber \rightarrow Pname

함수적 종속성

- X와 Y를 임의의 애트리뷰트들의 집합이라고 할 때, X의 값이 Y의 값을 유일하게(unique) 결정한다면

"X는 Y를 함수적으로 결정한다(functionally determines)"라고 함 (X, Y는 R의 Subset)

- X \rightarrow Y로 표기하고, "Y는 X에 함수적으로 종속된다"라고 함 (값은 X에 의존하게 된다.)

- 함수적 종속성은 모든 릴레이션 인스턴스 r(R)에 대하여 성립해야 함

함수적 종속성의 검사 방법

- 릴레이션 인스턴스 r(R)에 속하는 어떠한 임의의 두 투플에 대해서도 속성들의 집합 X에 대해 동일한

값을 가질 때마다 Y에 대해서도 동일한 값을 가진다면 X \rightarrow Y라는 함수적 종속성이 성립한다.

- 즉, r(R)에서의 임의의 두 투플 t_1 과 t_2 에 대해 $t_1[X] = t_2[X]$ 이면, $t_1[Y] = t_2[Y]$ 이다.

FD는 특정 릴레이션 인스턴스보다는 실세계에서 존재하는 애트리뷰트들 사이의 제약조건으로부터 유도된다.

FD 제약조건의 예제

- 주민등록번호는 사원의 이름을 결정한다.

o SSN \rightarrow ENAME

- 프로젝트 번호는 프로젝트 이름과 위치를 결정한다.

o PNUMBER \rightarrow {PNAME, PLOCATION}

- 사원의 주민등록번호와 프로젝트 번호는 그 사원이 일주일동안 그 프로젝트를 위해서 일하는 시간을 결정한다.

o {SSN, PNUMBER} \rightarrow HOURS

- FD는 스키마 R에 있는 애트리뷰트들의 특성이며, 모든 릴레이션 인스턴스 r(R)에서 성립해야 하는 성질이다.

- K가 R의 key이면 K는 R의 모든 애트리뷰트들을 함수적으로 결정한다 \rightarrow key에 대해서는 동일한 값을 갖는 두개의

($t_1[K] = t_2[K]$ 인 서로 다른 두개의 투플이 존재하지 않기 때문에). "distinct" tuple을 가질 수 없기 때문이다.

SK

X \rightarrow Y라고 해서 반드시 Y \rightarrow X인 것은

아니다

$$\forall t_i \in r(R), \forall t_k \in r(R) \quad t_i[x] = t_k[x] \rightarrow t_i[y] = t_k[y] \Rightarrow \text{True}$$

$$\begin{array}{lll} t_i = 1 \rightarrow t_i = 2 & t_i = 1 \rightarrow t_i = 1 & D \rightarrow ABCD \\ T \rightarrow F \quad \text{False} & T \rightarrow T = T & D \rightarrow AB \quad \text{True} \\ & & 1 = 1 \rightarrow \langle 1, 2, 3, 4 \rangle \\ & & = \langle 1, 2, 3, 4 \rangle \end{array}$$

8번은 True이므로 4번은 False로 True

$$\begin{array}{lll} t_i = 1 \rightarrow t_i = 2 & t_i = 1 \rightarrow t_i = 1 & D \rightarrow ABCD \\ T \rightarrow F \quad \text{False} & T \rightarrow T = T & D \rightarrow AB \quad \text{True} \\ & & 1 = 1 \rightarrow \langle 1, 2, 3, 4 \rangle \\ & & = \langle 1, 2, 3, 4 \rangle \end{array}$$

함수적 종속성의 추론 규칙

- 설계자는 주어진(알려진) FD의 집합 F를 가지고, 추가로 성립하는 FD들을 추론할 수 있다.

암스트롱의 추론 규칙들

- A1. (재귀성 규칙) Y \subseteq X이면, X \rightarrow Y이다.
- A2. (부가성 규칙) X \rightarrow Y이면, XZ \rightarrow YZ이다. (표기: XZ는 X \cup Z를 의미)
- A3. (이행성 규칙) X \rightarrow Y이고 Y \rightarrow Z이면, X \rightarrow Z이다.

- A1, A2, A3는 sound하고 complete 추론 규칙 집합을 형성한다.

추가적으로 유용한 추론 규칙들

- (분해 규칙) X \rightarrow YZ이면, X \rightarrow Y이고 X \rightarrow Z이다.
- (합집합 규칙) X \rightarrow Y이고 X \rightarrow Z이면, X \rightarrow YZ이다.
- (의사이행성 규칙) X \rightarrow Y이고 WY \rightarrow Z이면, WX \rightarrow Z이다.

- 위의 세 규칙 뿐 아니라 다른 추론 규칙들도 A1, A2, A3으로부터 추론 가능하다 (완전성 특성).

FD의 집합 F의 폐포(closure) : F⁺

- F로부터 추론할 수 있는 모든 가능한 함수적 종속성들의 집합

F 하에서 속성 집합 X의 폐포(closure of X under F) : X⁺

- 함수적 종속성 집합 F를 사용하여 X에 의해 함수적으로 결정되는 모든 애트리뷰트들의 집합

주어진 relation instance에 대해서, FD는

특정 attributes 사이에 조건을 수도 있다:

in CAR, ?State, Driver_license_number \rightarrow ?SSN?

dependencies이 위처럼 보이는 tuples로 인해 특정

$$\begin{array}{l} A \rightarrow B \\ B \rightarrow C \\ A \rightarrow C \end{array}$$

Second, IP는 강의실

강의실 \rightarrow 강의실 태그.

두번만 같은 태그.

을 넣을 수 있다.

① ?A \rightarrow B, B \rightarrow C

② ?A \rightarrow B, B \rightarrow C, A \rightarrow C

증거와 사업 범위

in Project, ?Placement \rightarrow ?Pnumber?

동일한 Placement에 Project가

존재할 수도 있기 때문이다.

$$\begin{array}{l} ① X \rightarrow Y \\ ② X \rightarrow W, ③ WY \rightarrow Z \Rightarrow ④ X \rightarrow Z \\ X \rightarrow WY \quad X \rightarrow WY \rightarrow Z \quad Z \end{array}$$