Chap 7. ER-and EER-to-Relational Mapping, and Other Relational Languages

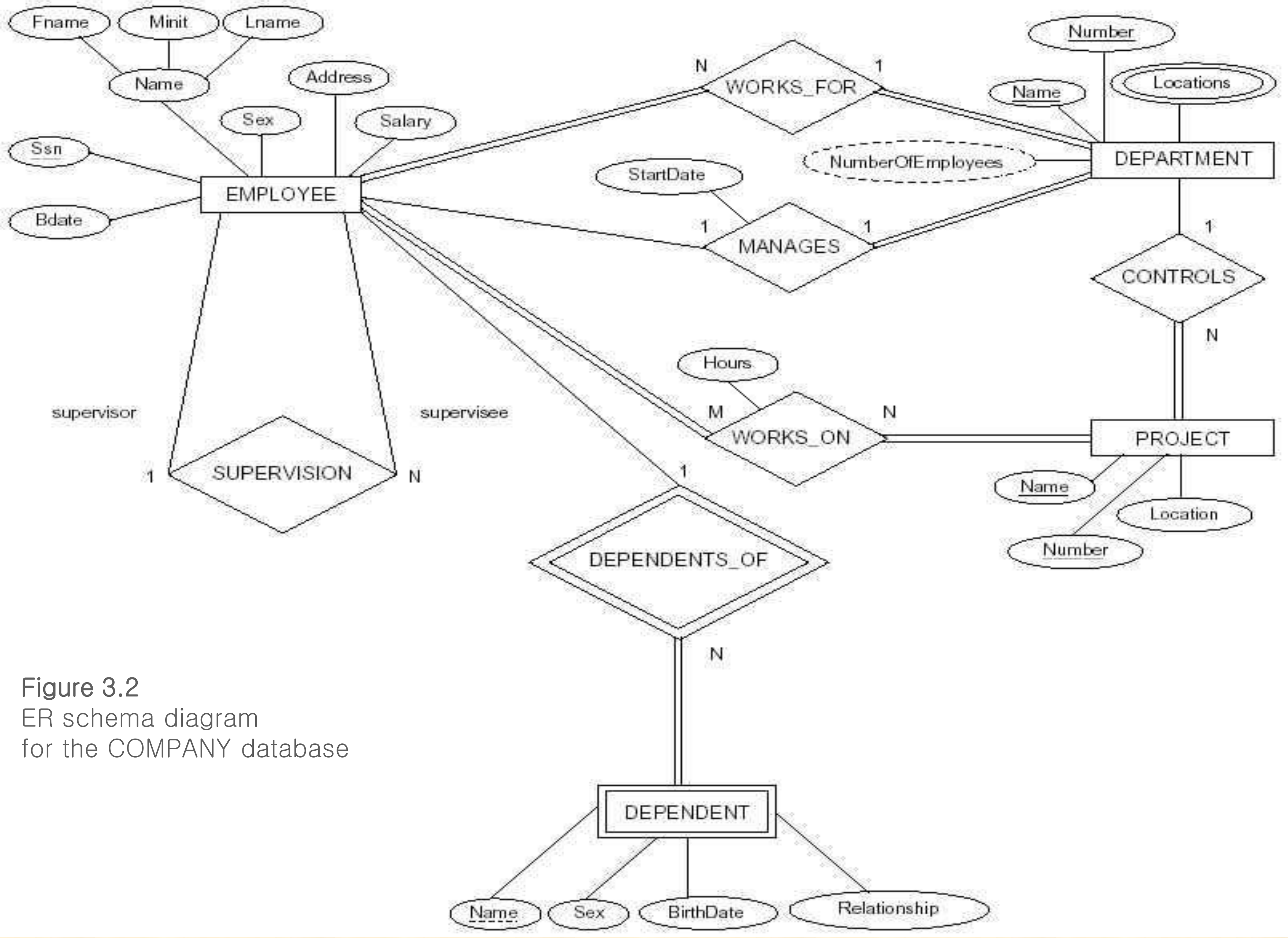


Figure 3.2
ER schema diagram
for the COMPANY database

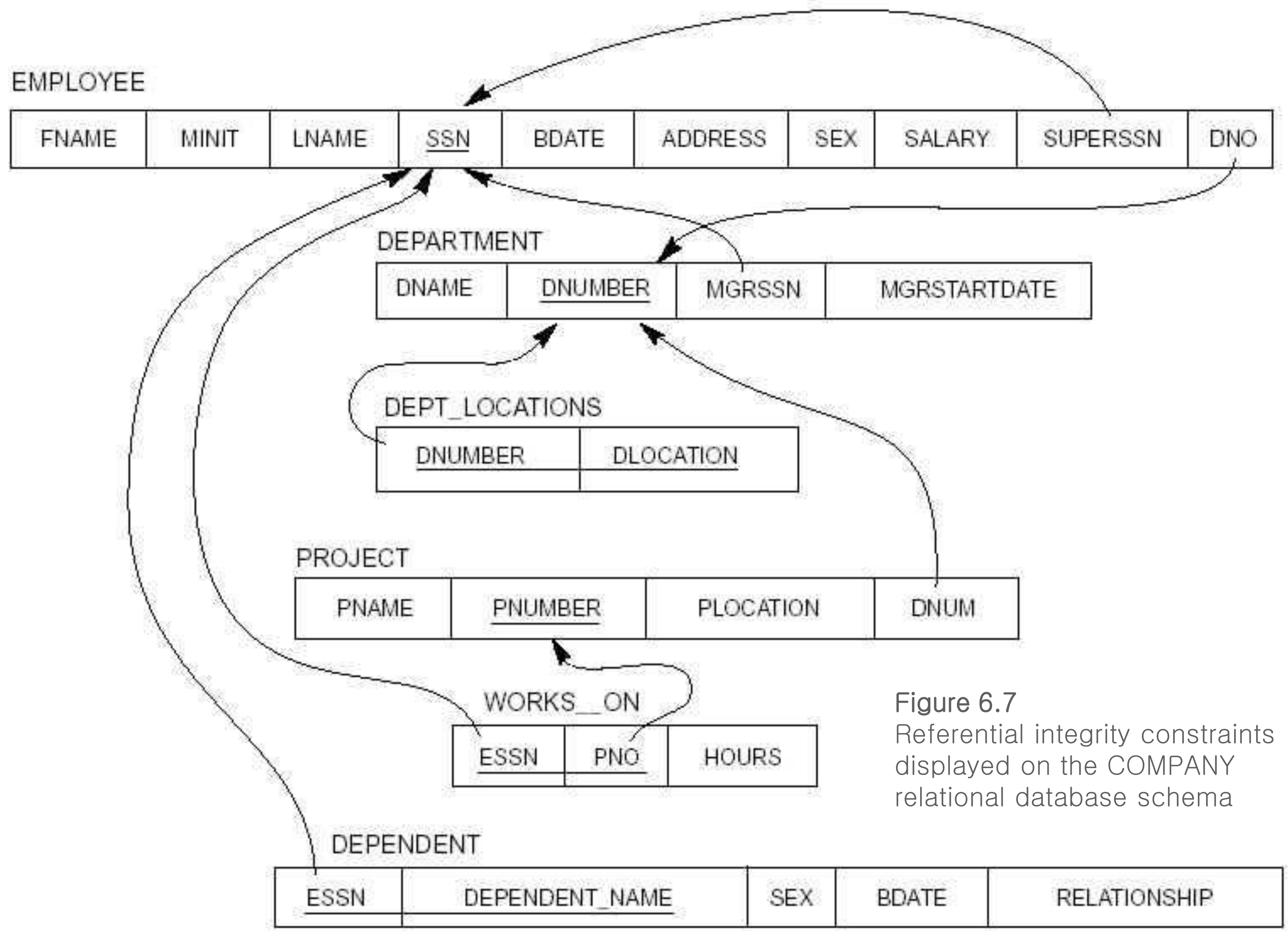


Figure 6.7
Referential integrity constraints displayed on the COMPANY relational database schema

ER to Relational Mapping Algorithm

- Step 1 :regular entity E in ER model
  => relation R in RDB
  - 복합 속성 -> 단순 속성을 변환
  - key 속성들 -> 1개를 primary key로 선정

Entities	Relations	primary key
EMPLOYEE	EMPLOYEE	SSN
DEPARTMENT	DEPARTMENT	DNUMBER
PROJECT	PROJECT	PNUMBER

ER to Relational Mapping Algorithm

- Step 2 Weak Entity Types

ER model	→	Relation Schema
Owner entity type E		S
Weak entity type E		R (?)

- primary key: primary key of S + partial key of W
- foreign key : primary key of S

EMPLOYEE (SSN)	EMPLOYEE (SSN)
DEPENDENT (DEP_NAME)	DEPENDENT

- primary key : ESSN, DEPENDENT_NAME
- foreign key : ESSN (<-renamed)

ER to Relational Mapping Algorithm

- Step 3 : 1:1 binary relationship type R
  - participating entity type (in ER)
    -> relation S(PK),T(PK)
  - choose either S or T (say S) [total participation]
  - S의 foreign key <= T의 primary key
  - S의 속성 <= R의 속성

MANAGE-1:1 binary relationship type

participating entity type : department, employee
(S=department(total), T=employee)

department의 foregin key<=EMPLOYEE의 primary key
(SSN rename as MGRSSN)

add startdate to department (renamed as MGESTARTDATE)

ER to Relational Mapping Algorithm

- alternative method: (E1 <- R -> E2)

  - relationship type& two entity types
    => one relation

  - If total participation & no other relationship type

ER to Relational Mapping Algorithm

- Step 4 : binary 1:N relationship type R
  - N side : relation S
  - 1 side : relation T
  - S의 foreign key <- T의 primary key
  - S의 속성 <- R의 속성

WORK_FOR : N side=EMPLOYEE(S), 1 side=DEPARTMENT(T)
EMPLOYEE의 FK <- DEPARTMENT의 PK (DNO)

SUPERVISION : EMPLOYEE의 FK <- EMPLOYEE의 PK(superssn)
CONTROL : PROJECT의 FK <- DEPARTMENT의 PK (DNUM)

ER to Relational Mapping Algorithm

- Step 5 – N:M relationship type R, participating entity type E1, E2 -> relation schema U, V
  - a new relation S (relationship relation schema)
  - S의 primary key <- U, V의 primary keys
  - S의 foreign key <- U, V의 primary keys
  - S의 속성 <- R의 속성

  WORK_ON entity type --------------> WORK_ON

  PK: {ESSN from EMPLOYEE + PNO from PROJECT}

  FK: ESSN, PNO attribute : HOUR
- 1:1, 1:N relationship type R ==> relation schema S
  - 1:N : S의 primary key <- N side relation schema의 PK
  - 1:1 : S의 primary key <- any relation schema의 PK

ER to Relational Mapping Algorithm

- Step 6 : multi-valued attribute A in ER

  -> new relation schema R

  - R의 속성 <-- A + K
    - k=A를 갖는 entity type 또는 relationship type에 해당되는 relation schema의 primary key
  - R의 primary key : A+K, foreign key : K

DEPARTMENT의 DLOCATION DEPT_LOCATIONS

PK : DLOCATION, DNUMBER

FK : DNUMBER to department

ER to Relational Mapping Algorithm

- Step 7 : n-ary relationship type R, n>2
  - new relation S
  - FK <- primary keys of participating entity types
  - PF <- FK's
    - 만약 참여하는 entity type(E)중에 max=1이 있으면 PK<-entity type E에 해당되는 relation E'의 PK

relationship type SUPPLY ----> relation SUPPLY

FK's : SNAME to SUPPLIER

PROJNAME to PROJECT

PARTNO to PART

PK : SNAME PROJNAME PARTNO

ER to Relational Mapping Algorithm

- ER model <------> Relational model
  - relationship type R over entity S, T
    - relation schema S' with PK & FK to T'
  - two tuples in S' and T' are related

    when S'PK = T'FK

    ==> equijoin

  - 1:1 and 1:N relationship type -> one join 연결
  - M:N relationship type ---> two join 연결
  - n-ary relationship type ---> n join 연결

ER to Relational Mapping Algorithm

직원이름, 과제이름, 과제 수행시간을 연결하려면

EMPLOYEE <- WORK_ON -> PROJECT

equijoin on SSN=ESSN and PNO=PNUMBER

(primary key <-> foreign key)

- multi-valued 속성 ==> relation schema

equijoin on DNUMBER to DEPT_LOCATION and DEPARTMENT

EER-to-Relation Mapping

Specialization / generalization

- Step 8

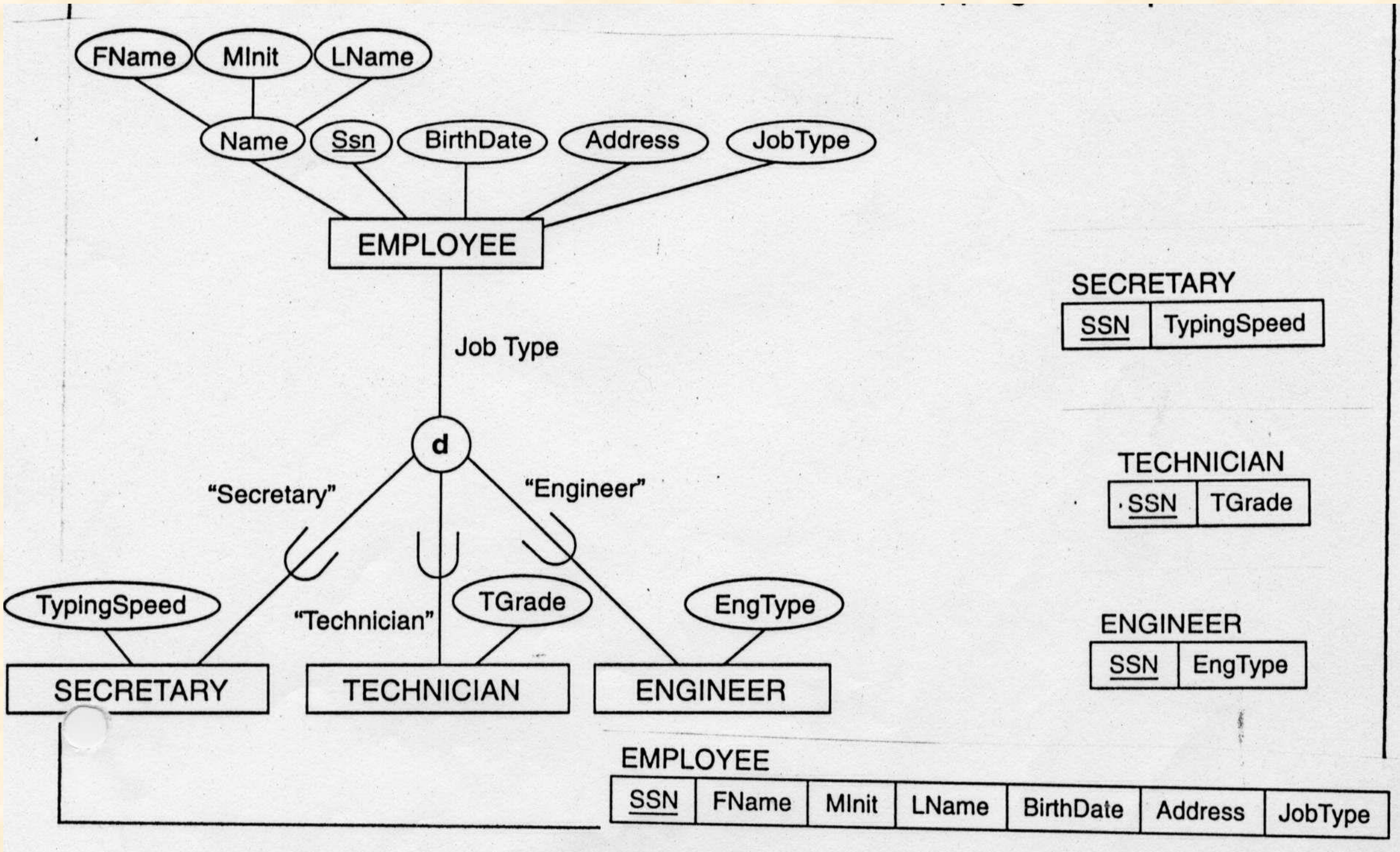
  (i) C <=> { S_1, S_2, … ,S_m }

  Superclass entity C(k, a_1, …, a_n), Key of C = k

  - create a relation L for C with attrs(L)={k,a_1,…,a_n}
    let PK(L) = k
  - create a relation L_i for each subclass S_i, i = 1..m
    attrs(L_i) = { k } ∪ { attributes of S_i }, PK(L_i) = k

    - equi-join on PK k between L and Li : inherited info
      (inclusion dependency: $\pi_{<k>}(L_i) \subseteq \pi_{<k>}(L)$)
    - work for any constraint(disjoint/overlapping, total/partial)

Specialization / generalization



Specialization / generalization

(ii) create a relation L_i for each subclass S_i

attrs(L_i) = { attributes of S_i } ∪ { k, a_1. ..., a_n }

PK(L_i) = k

- disjoint & total (if partial : lost entity, if overlapping : redundant info.)
- to get info. on C, outer-join on L_i's or search each L_i

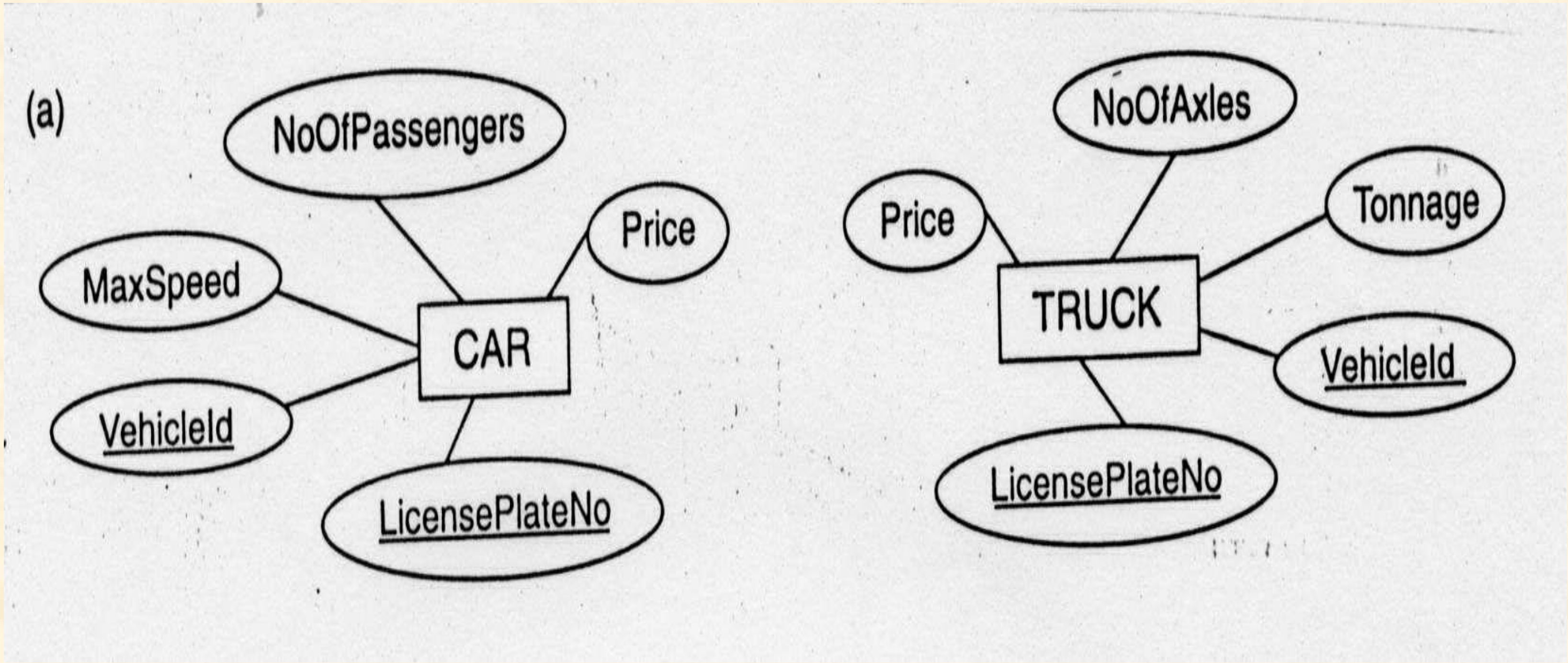
CAR

VehicleId	LicensePlateNo	Price	MaxSpeed	NoOfPassengers

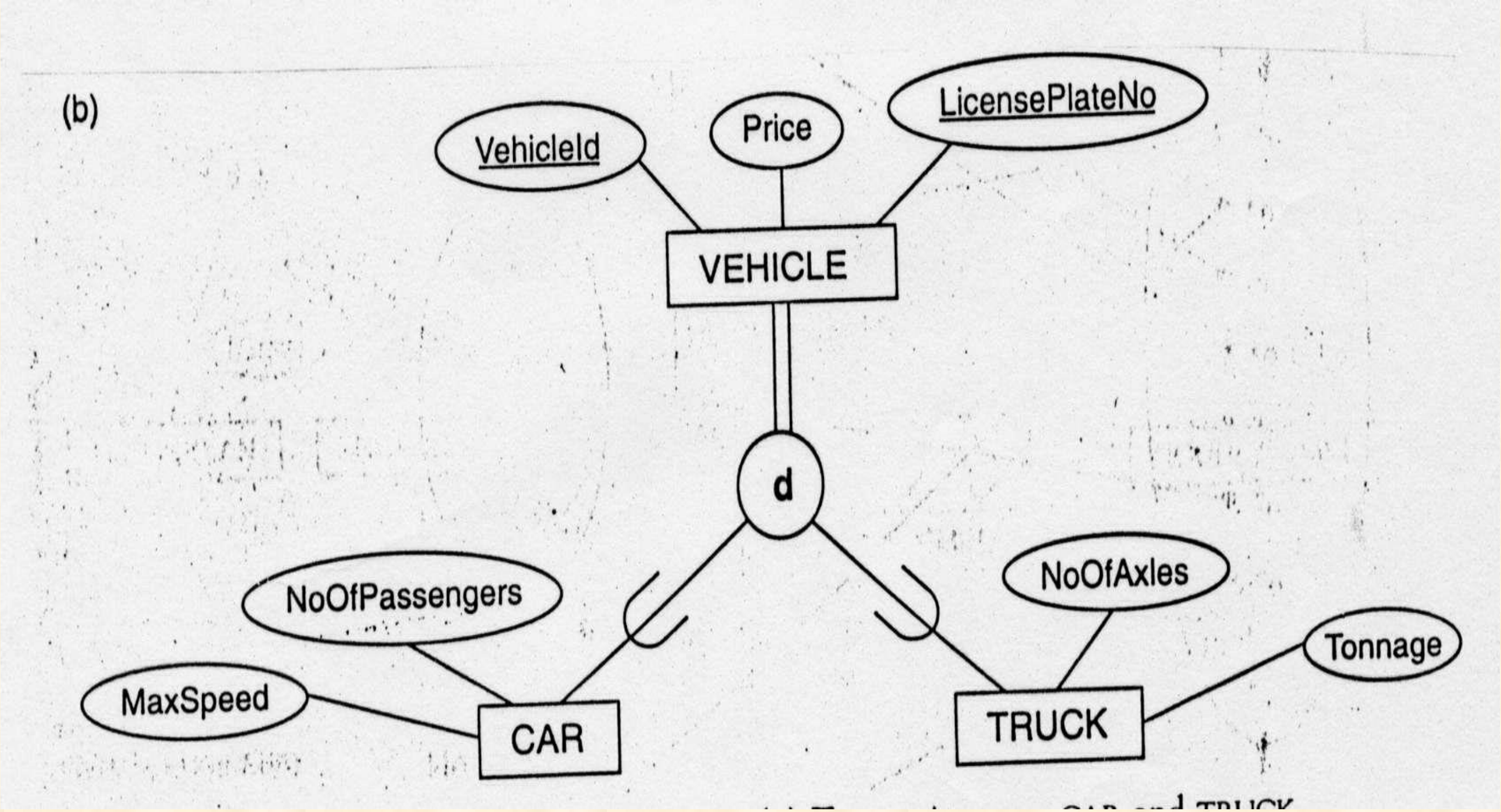
TRUCK

VehicleId	LicensePlateNo	Price	NoOfAxles	Tonnage

Specialization / generalization



Specialization / generalization



Specialization / generalization

- (iii) create a single relation L

  attrs(L) = { k, a_1, …, a_n } $\cup$ { attributes of S_1 } $\cup$… $\cup$ { attributes of S_m } $\cup$ { t }

  PK(L) = k

  where t = type info. of subclass for each tuple

  - disjoint [t = { 1,….,m}]
    - if partial, t= null
    - if attribute-defined subclasses, use defining attr as t

EMPLOYEE

SSN	FName	MInit	LName	BirthDate	Address	JobType	TypingSpeed	TGrade	EngType

  - potentially large # of null values (when # of subclass's attributes is small)

Specialization / generalization

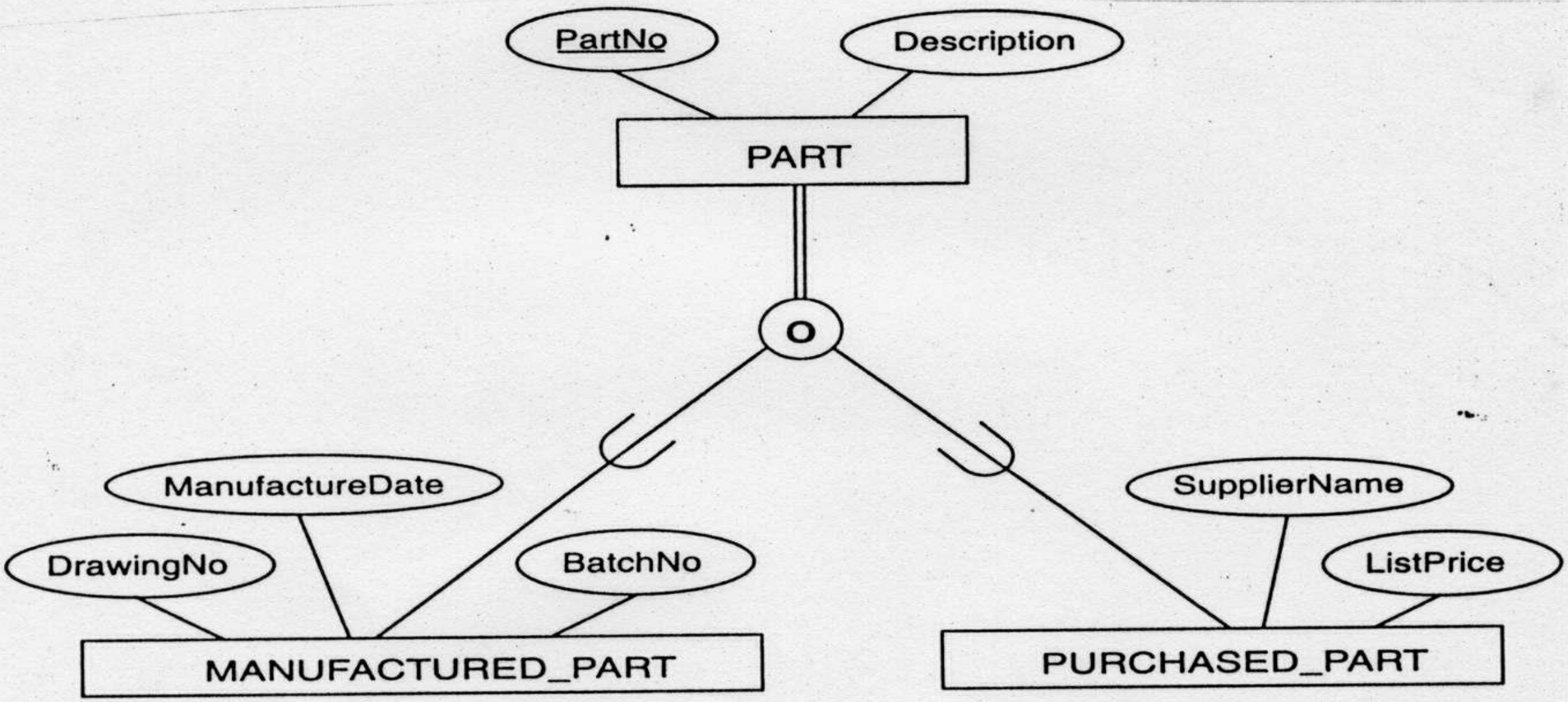
- (iv) create a single relation schema L

  attrs(L) = { k, a_1, ..., a_n } $\cup$ { attributes of S_1 }$\cup$ ... $\cup$ { attributes of S_m } $\cup$ {t_1, t_2,..., t_m }

  PK(L) = k

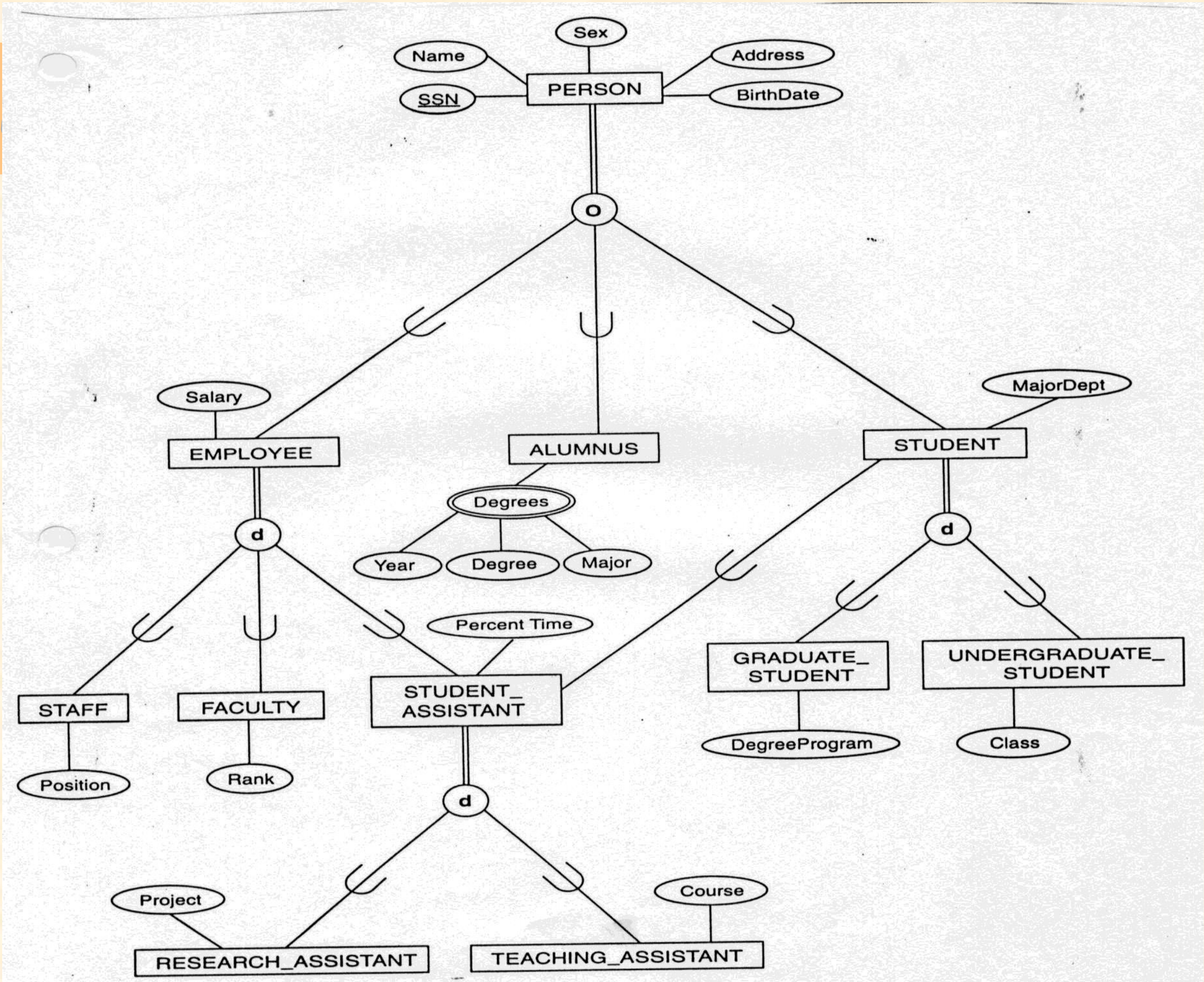
  - overlapping : t_i = boolean type indicating the membership of each entity for subclass S_i

Specialization / generalization



PART

PartNo	Description	MFlag	DrawingNo	ManufactureDate	BatchNo	PFlag	SupplierName	ListPrice


Sex
Name
Address
SSN
PERSON
BirthDate
o
Salary
MajorDept
EMPLOYEE
ALUMNUS
STUDENT
Degrees
d
d
Year
Degree
Major
Percent Time
GRADUATE_
STUDENT
UNDERGRADUATE_
STUDENT
STAFF
FACULTY
STUDENT_
ASSISTANT
DegreeProgram
Class
Position
Rank
d
Project
Course
RESEARCH_ASSISTANT
TEACHING_ASSISTANT

Specialization / generalization

- Hierarchy Mapping

PERSON

<u>SSN</u>	Name	BirthDate	Sex	Address

EMPLOYEE

<u>SSN</u>	Salary	EmployeeType	Position	Rank	PercentTime	RAFlag	TAFlag	Project	Course

ALUMNUS

<u>SSN</u>

ALUMNUS_DEGREES

<u>SSN</u>	<u>Year</u>	<u>Degree</u>	<u>Major</u>

STUDENT

<u>SSN</u>	MajorDept	GradFlag	UndergradFlag	DegreeProgram	Class	StudAssistFlag

Categories

- Step 9: Mapping of union types
  - Define a new surrogate key in a relation correspond to a category.
  - Use the surrogate key as foreign keys in the super-classes of the category.

Categories

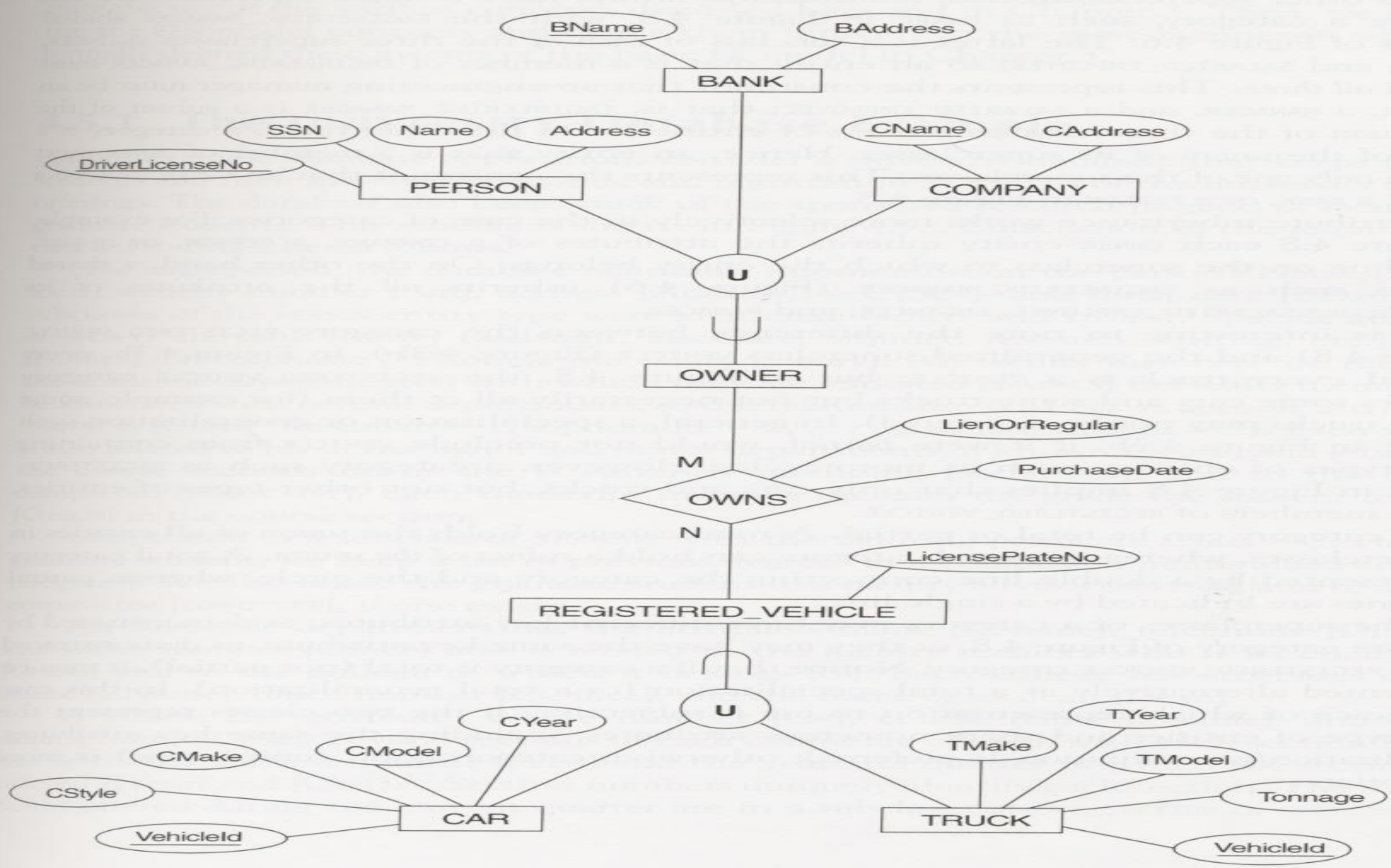


FIGURE 4.8 Two categories (union types): OWNER and REGISTERED_VEHICLE.

Categories

PERSON

SSN	DriverLicenseNo	Name	Address	OwnerId

BANK

BName	BAddress	OwnerId

COMPANY

CName	CAddress	OwnerId

OWNER

OwnerId

REGISTERED_VEHICLE

VehicleId	LicensePlateNumber

CAR

VehicleId	CStyle	CMake	CModel	CYear

TRUCK

VehicleId	TMake	TModel	Tonnage	TYear

OWNS

OwnerId	VehicleId	PurchaseDate	LienOrRegular

FIGURE 7.6 Mapping the EER categories (union types) in Figure 4.7 to relations.