Database Management Systems

ICT1213

Relational Database Design by ER- and EER to Relational Mapping

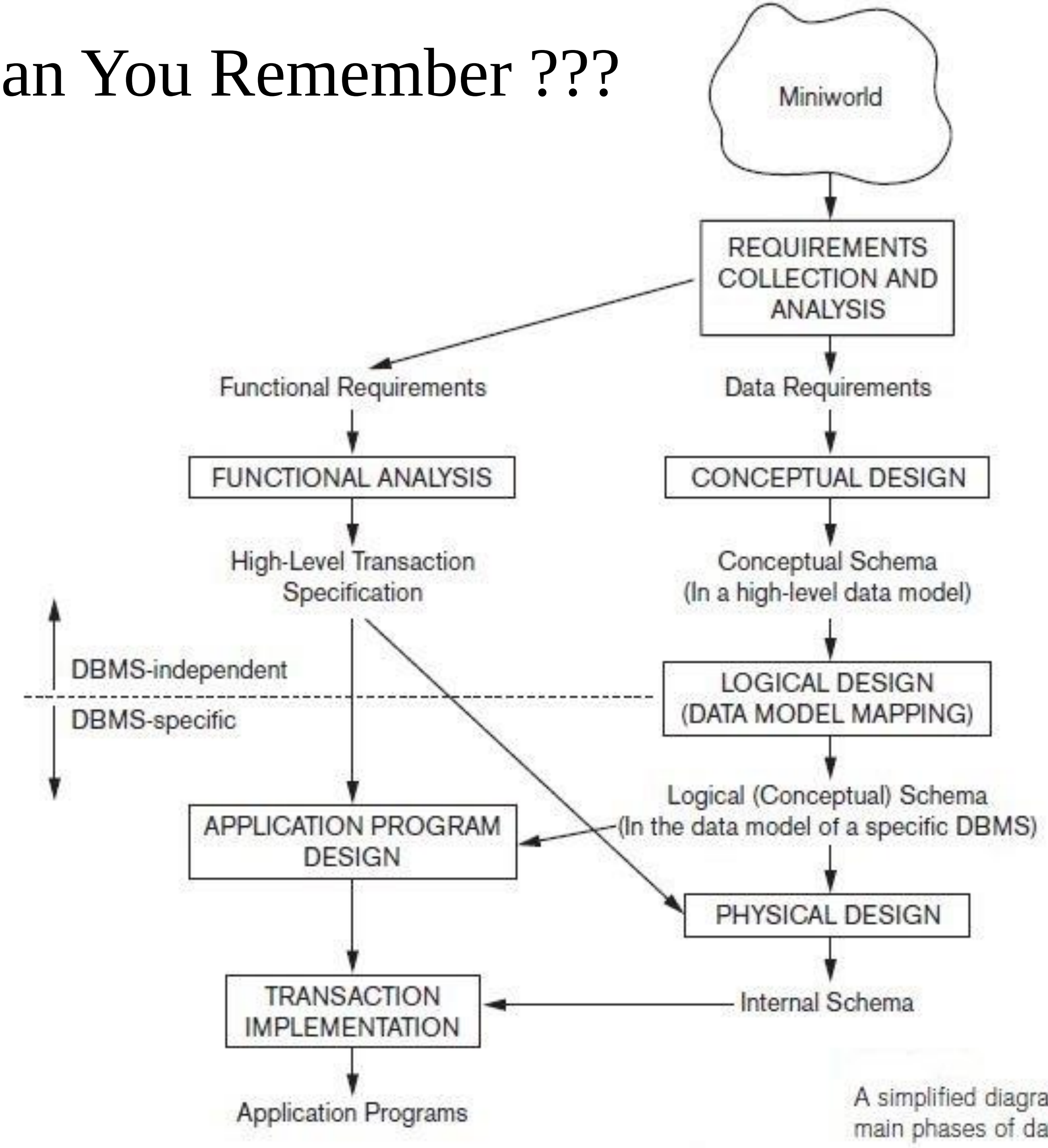
Department of ICT
Faculty of Technology
University of Ruhuna

Lecture 6

What we discuss Today ……..

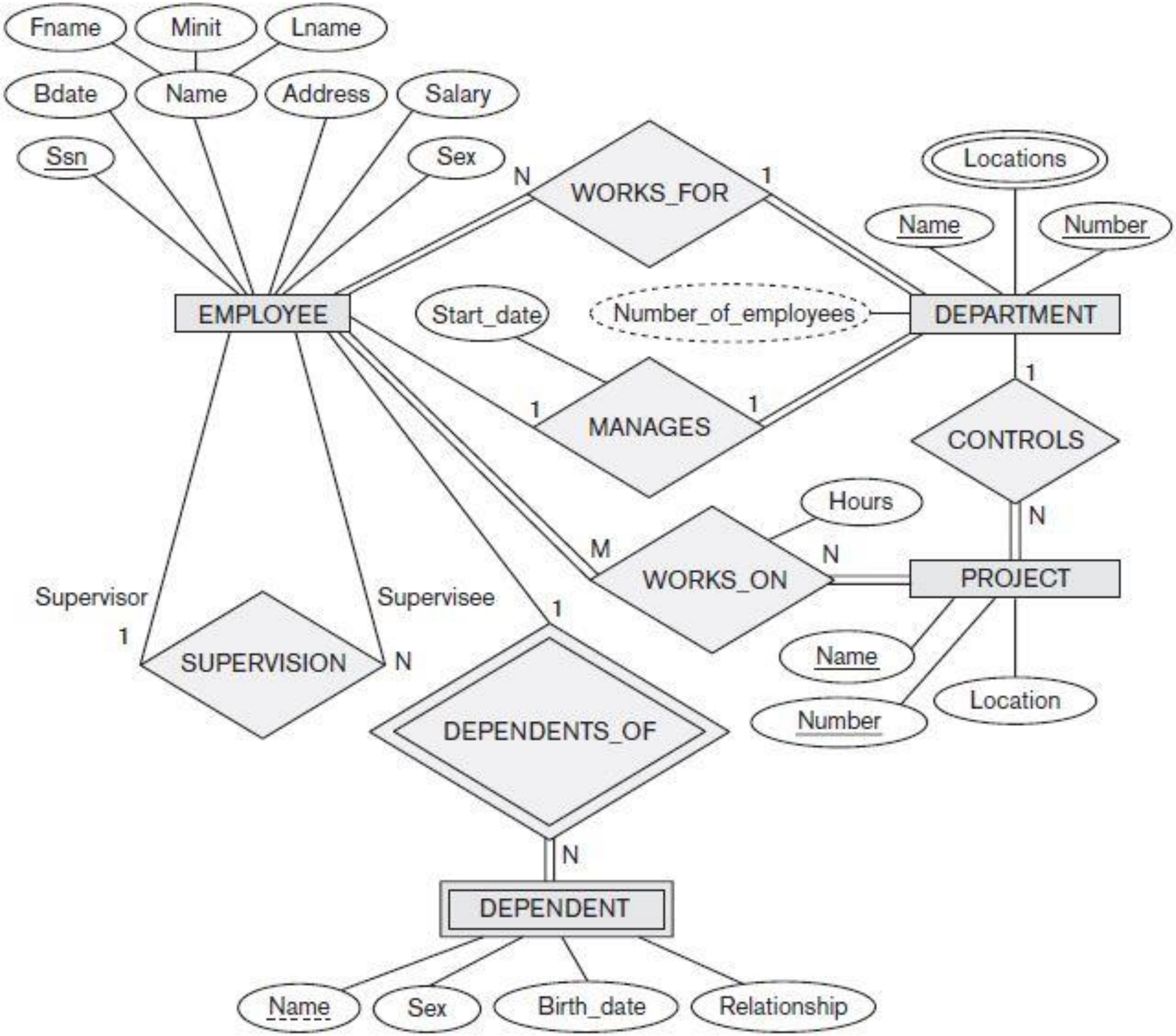
- Relational Database Design Using ER-to-Relational Mapping
- Relational Database Design Using EER-to-Relational Mapping

Can You Remember ???


Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data 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

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

The ER conceptual schema diagram for the COMPANY database.



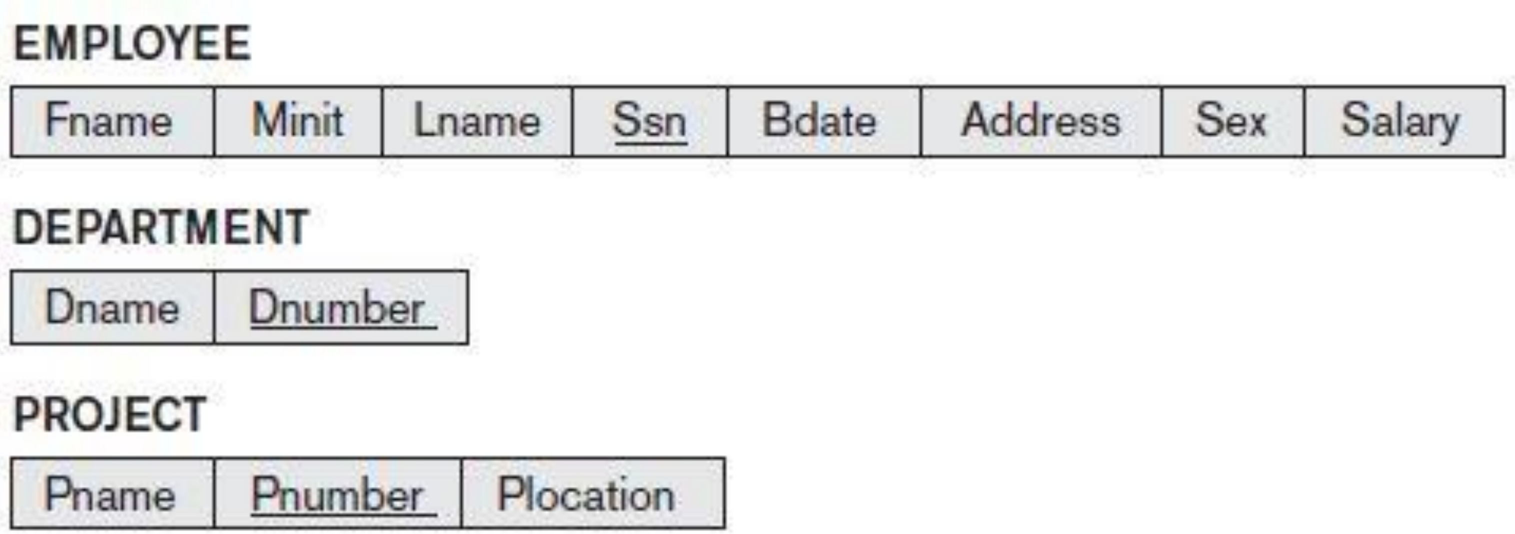
Mapping
ER Model Constructs
to
Relations

Step 1: Mapping of Regular Entity Types

- For each regular (strong) entity type *E* in the ER schema, create a relation *R* that includes all the simple attributes of *E*
- Include only the simple component attributes of a composite attribute
- Choose one of the key attributes of *E* as the primary key for *R*
- If the chosen key of *E* is a composite, then the set of simple attributes that form it will together form the primary key of *R*
- If multiple keys were identified for *E* during the conceptual design, the information describing the attributes that form each additional key is kept in order to specify secondary (unique) keys of relation *R*

Step 1: Mapping of Regular Entity Types

- The relations that are created from the mapping of entity types are sometimes called **entity relations** because each tuple represents an entity instance



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

DEPARTMENT

Dname	Dnumber

PROJECT

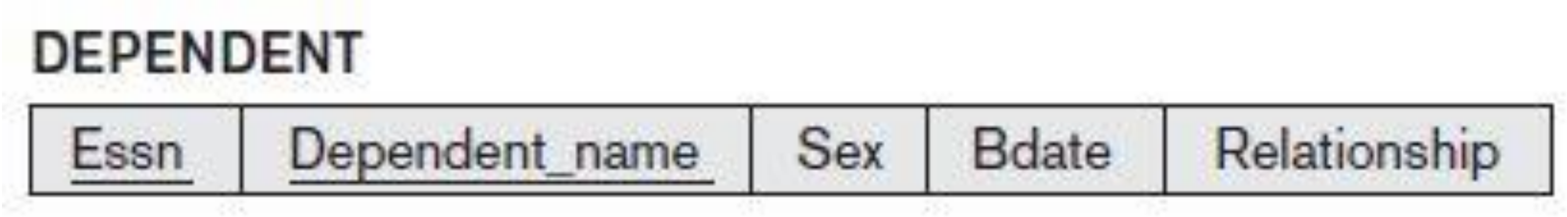
Pname	Pnumber	Plocation

Step 2: Mapping of Weak Entity Types

- For each weak entity type *W* in the ER schema with owner entity type *E*, create a relation *R* and include all simple attributes (or simple components of composite attributes) of *W* as attributes of *R*
- In addition, include as foreign key attributes of *R*, the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s)
- This takes care of mapping the identifying relationship type of *W*
- The primary key of *R* is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type *W*, if any
- If there is a weak entity type *E*2 whose owner is also a weak entity type *E*1, then *E*1 should be mapped before *E*2 to determine its primary key first

Step 2: Mapping of Weak Entity Types

- The primary key of the DEPENDENT relation is the combination {Essn, Dependent_name}, because Dependent_name is the partial key of DEPENDENT
- It is common to choose the propagate (CASCADE) option for the referential triggered action on the foreign key in the relation corresponding to the weak entity type, since a weak entity has an existence dependency on its owner entity.
- This can be used for both ON UPDATE and ON DELETE



DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

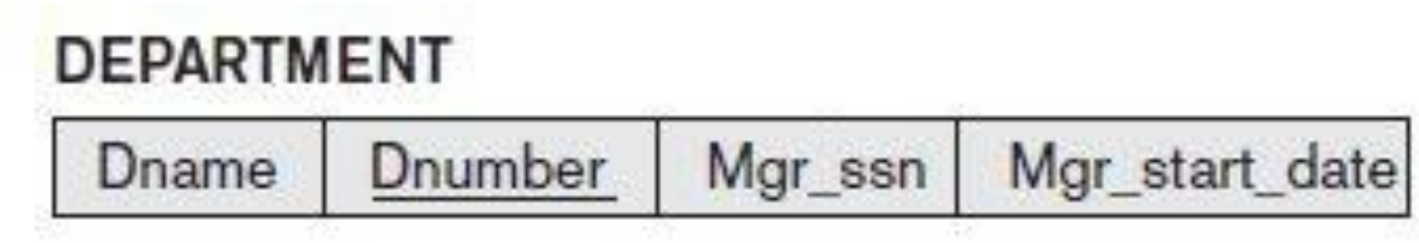
Step 3: Mapping of Binary 1:1 Relationship Types

- For each binary 1:1 relationship type *R* in the ER schema, identify the relations *S* and *T* that correspond to the entity types participating in *R*. There are three possible approaches
  - The foreign key approach
  - The merged relationship approach
  - The cross reference or relationship relation approach

Step 3: Mapping of Binary 1:1 Relationship Types

The foreign key approach

- Choose one of the relations—*S*, say—and include as a foreign key in *S* the primary key of *T*
- It is better to choose an entity type with *total participation* in *R* in the role of *S*.
- Include all the simple attributes (or simple components of composite attributes) of the 1:1 relationship type *R* as attributes of *S*



DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

Step 3: Mapping of Binary 1:1 Relationship Types

Merged relation approach

- An alternative mapping of a 1:1 relationship type is to merge the two entity types and the relationship into a single relation
- This is possible when *both participations are total*, as this would indicate that the two tables will have the exact same number of tuples at all times

Step 3: Mapping of Binary 1:1 Relationship Types

Cross-reference or relationship relation approach

- The third option is to set up a third relation *R* for the purpose of cross-referencing the primary keys of the two relations *S* and *T* representing the entity types
- The relation *R* is called a **relationship relation** (or sometimes a **lookup table**), because each tuple in *R* represents a relationship instance that relates one tuple from *S* with one tuple from *T*

Step 3: Mapping of Binary 1:1 Relationship Types

Cross-reference or relationship relation approach

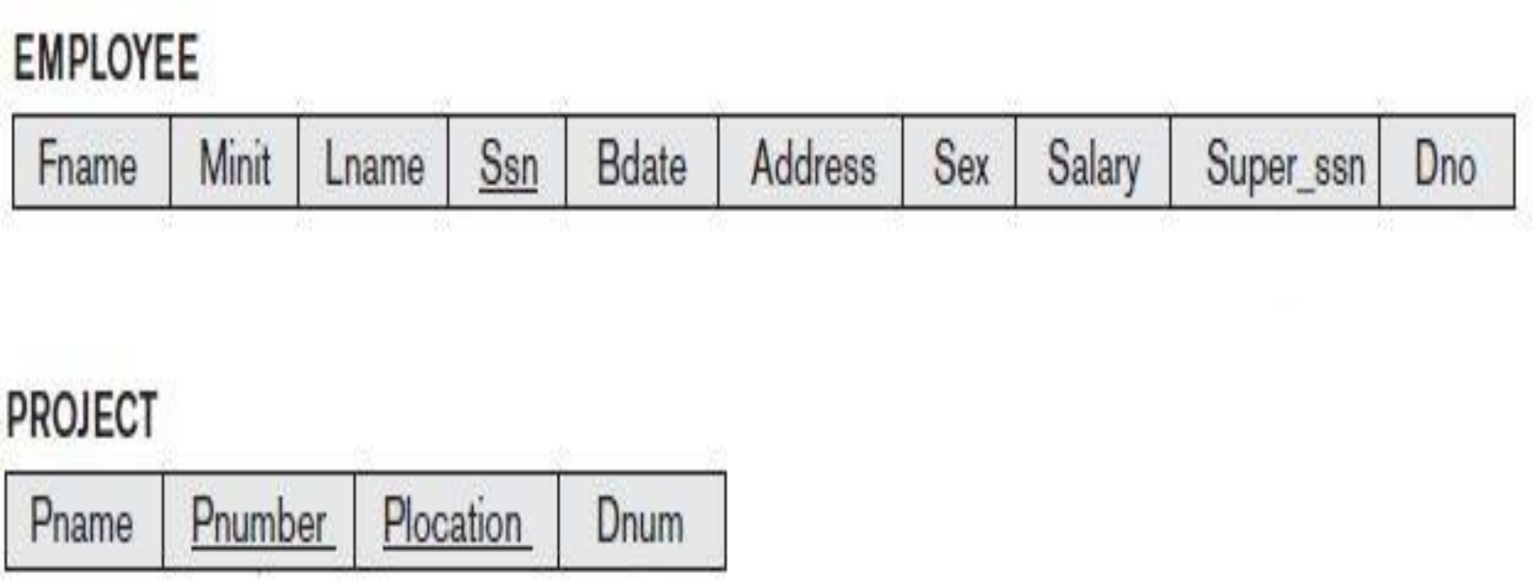
- The relation *R* will include the primary key attributes of *S* and *T* as foreign keys to *S* and *T*
- The primary key of *R* will be one of the two foreign keys, and the other foreign key will be a unique key of *R*
- The drawback is having an extra relation, and requiring an extra join operation when combining related tuples from the tables

Step 4: Mapping of Binary 1:N Relationship Types

- For each regular binary 1:N relationship type *R*, identify the relation *S* that represents the participating entity type at the *N-side* of the relationship type.
- Include as foreign key in *S* the primary key of the relation *T* that represents the other entity type participating in *R*
- we do this because each entity instance on the N-side is related to at most one entity instance on the 1-side of the relationship type.
- Include any simple attributes (or simple components of composite attributes) of the 1:N relationship type as attributes of *S*

Step 4: Mapping of Binary 1:N Relationship Types

- An alternative approach is to use the **relationship relation** (cross-reference) option as in the third option for binary 1:1 relationships



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

PROJECT

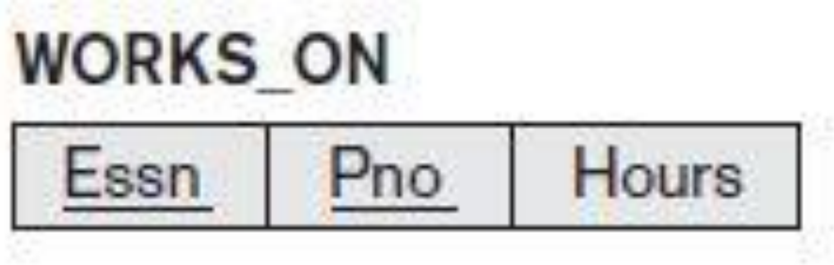
Pname	Pnumber	Plocation	Dnum

Step 5: Mapping of Binary M:N Relationship Types

- For each binary M:N relationship type *R*, create a new relation *S* to represent *R*.
- Include as foreign key attributes in *S* the primary keys of the relations that represent the participating entity types
- Their *combination* will form the primary key of *S*
- Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of *S*
- Notice that we cannot represent an M:N relationship type by a single foreign key attribute in one of the participating relations (as we did for 1:1 or 1:N relationship types) because of the M:N cardinality ratio, we must create a separate *relationship relation S*

Step 5: Mapping of Binary M:N Relationship Types

- The propagate (CASCADE) option for the referential triggered action should be specified on the foreign keys in the relation corresponding to the relationship *R*, since each relationship instance has an existence dependency on each of the entities it relates.
- This can be used for both ON UPDATE and ON DELETE



WORKS_ON

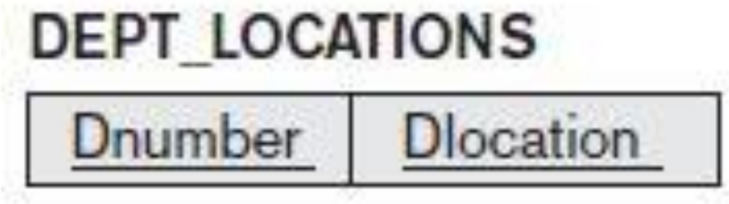
Essn	Pno	Hours

Step 6: Mapping of Multivalued Attributes

- For each multivalued attribute *A*, create a new relation *R*.
- This relation *R* will include an attribute corresponding to *A*, plus the primary key attribute *K*—as a foreign key in *R*—of the relation that represents the entity type or relationship type that has *A* as a multivalued attribute.
- The primary key of *R* is the combination of *A* and *K*.
- If the multivalued attribute is composite, we include its simple components

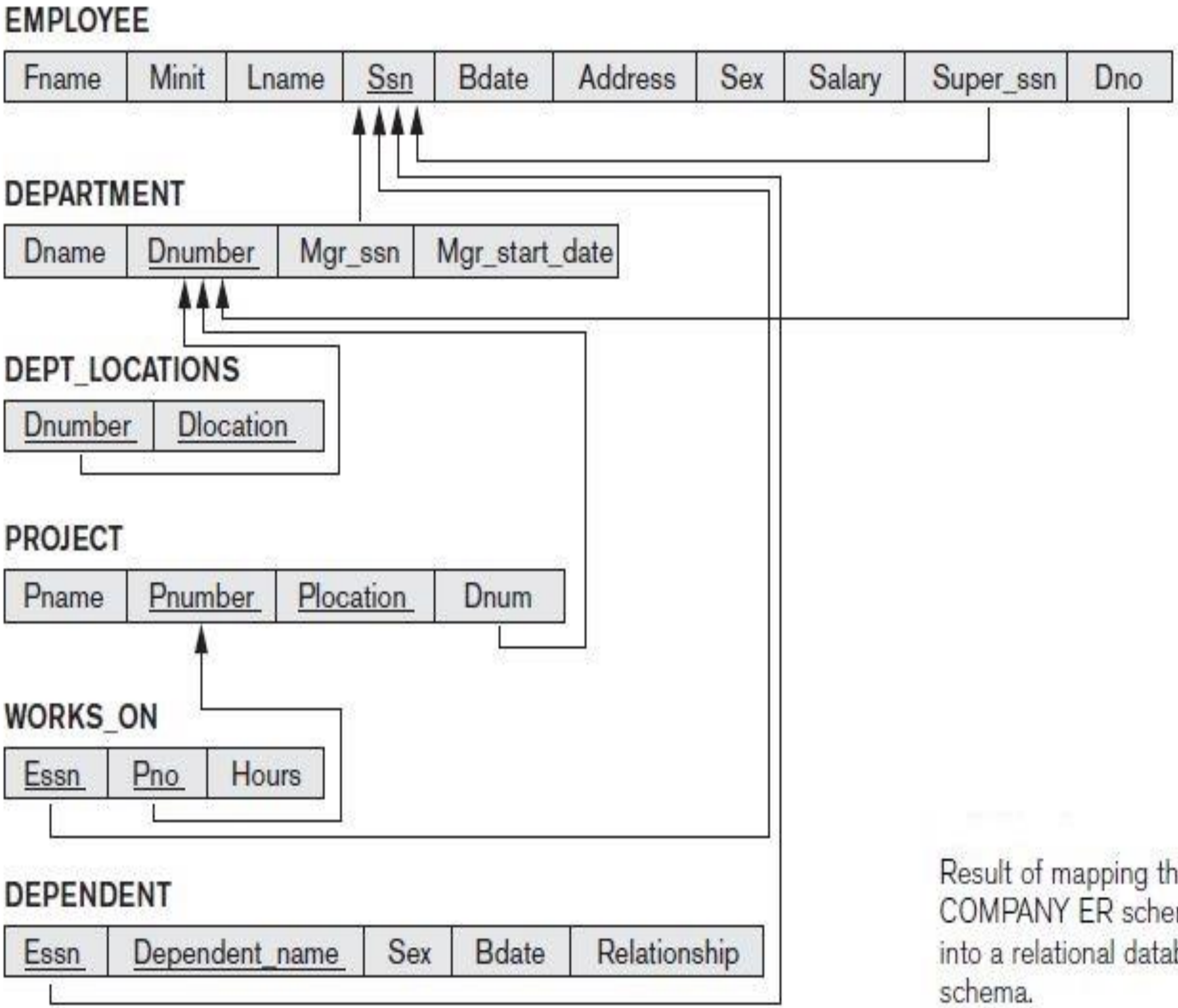
Step 6: Mapping of Multivalued Attributes

- The propagate (CASCADE) option for the referential triggered action should be specified on the foreign key in the relation *R* corresponding to the multivalued attribute for both ON UPDATE and ON DELETE
- When a multivalued attribute is composite, only some of the component attributes are required to be part of the key of *R*
- *These attributes are similar to a partial key of a weak entity type that corresponds to the multivalued attribute*



DEPT_LOCATIONS

Dnumber	Dlocation



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

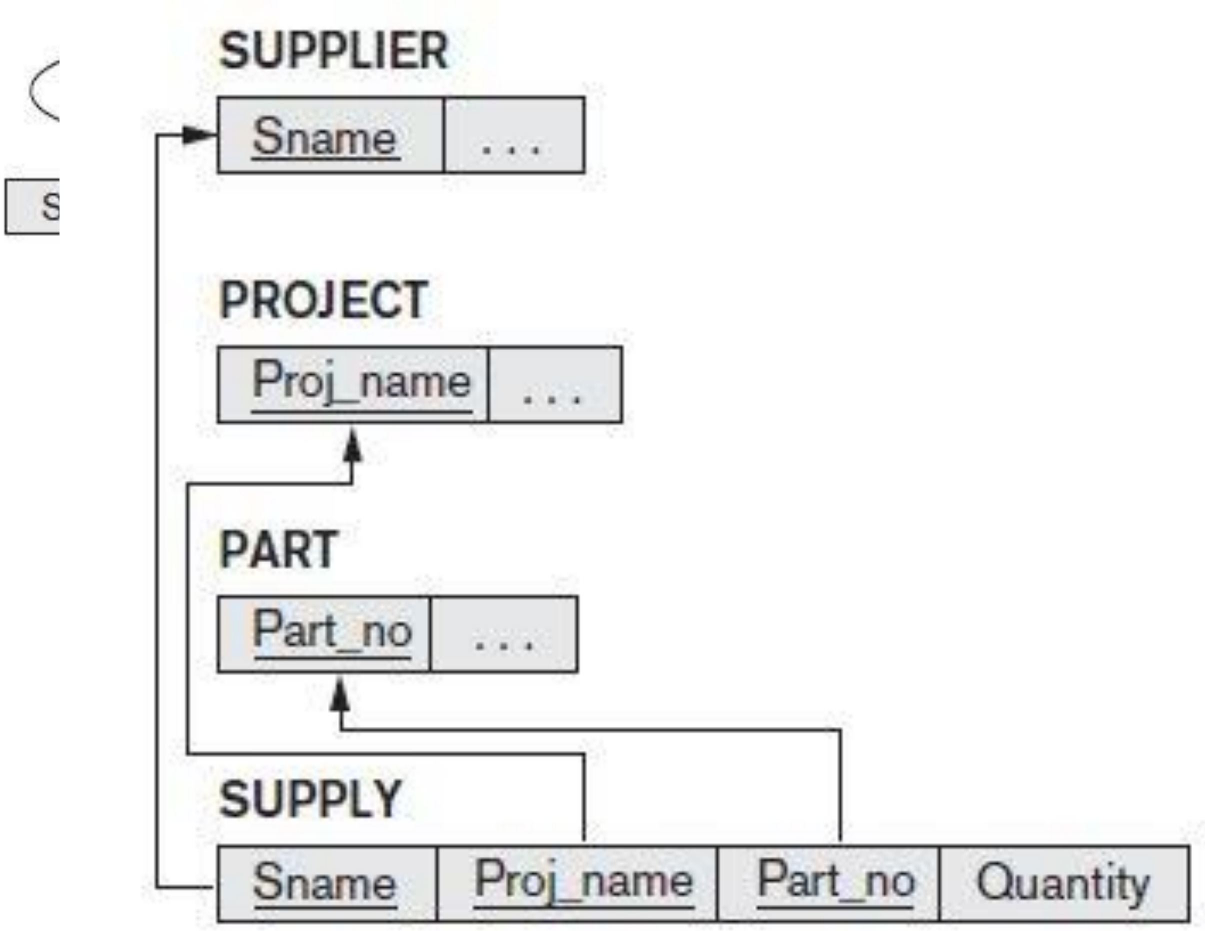
Essn	Dependent_name	Sex	Bdate	Relationship

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

Step 7: Mapping of *N*-ary Relationship Types

- For each *n*-ary relationship type *R*, where *n* > 2, create a new relation *S* to represent *R*. Include as foreign key attributes in *S* the primary keys of the relations that represent the participating entity types.
- Also include any simple attributes of the *n*-ary relationship type (or simple components of composite attributes) as attributes of *S*.
- The primary key of *S* is usually a combination of all the foreign keys that reference the relations representing the participating entity types.
- If the cardinality constraints on any of the entity types *E* participating in *R* is 1, then the primary key of *S* should not include the foreign key attribute that references the relation *E* corresponding to *E*

Step 7: Mapping of *N*-ary Relationship Types



SUPPLIER

Sname	. . .

PROJECT

Proj_name	. . .

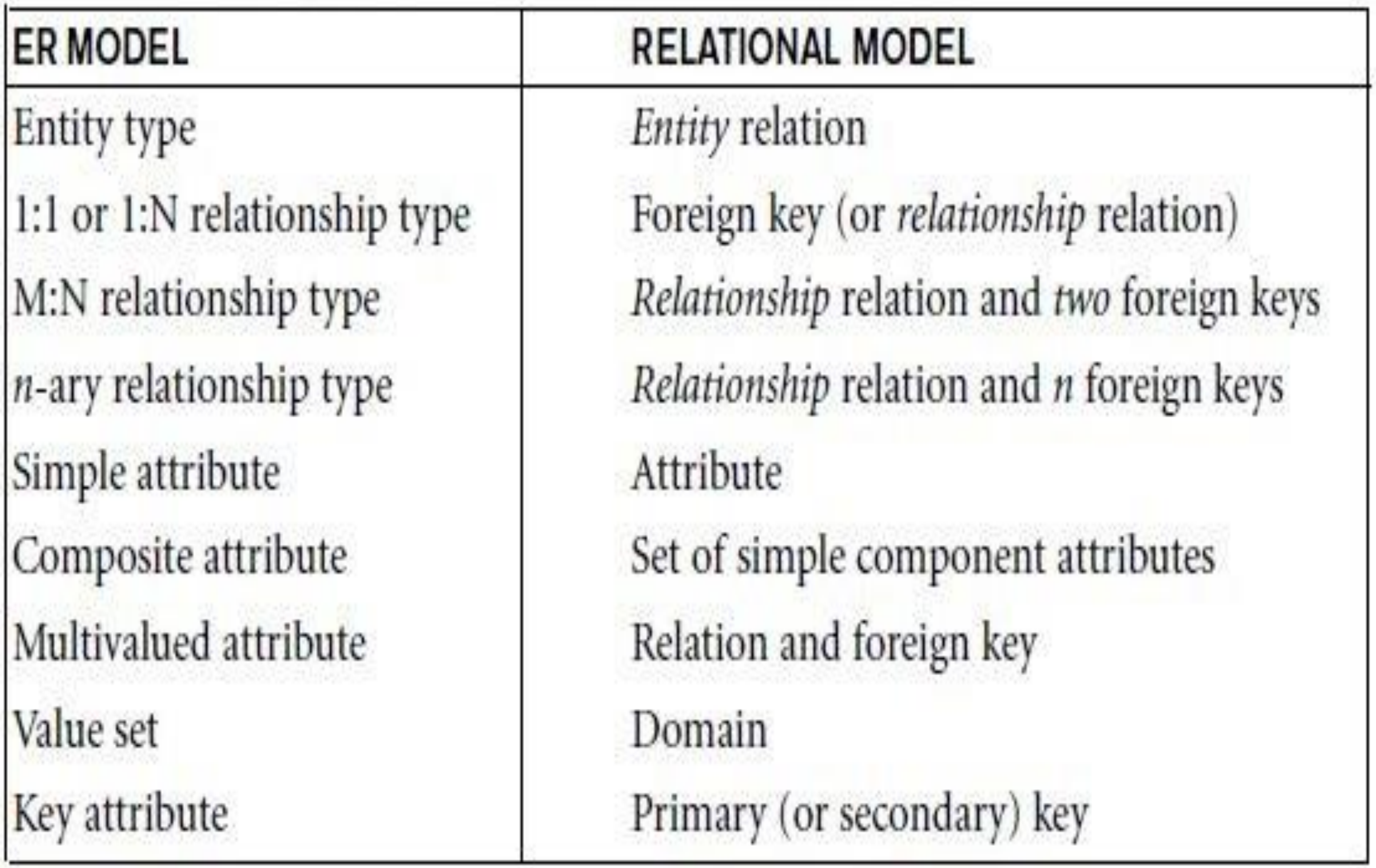
PART

Part_no	. . .

SUPPLY

Sname	Proj_name	Part_no	Quantity

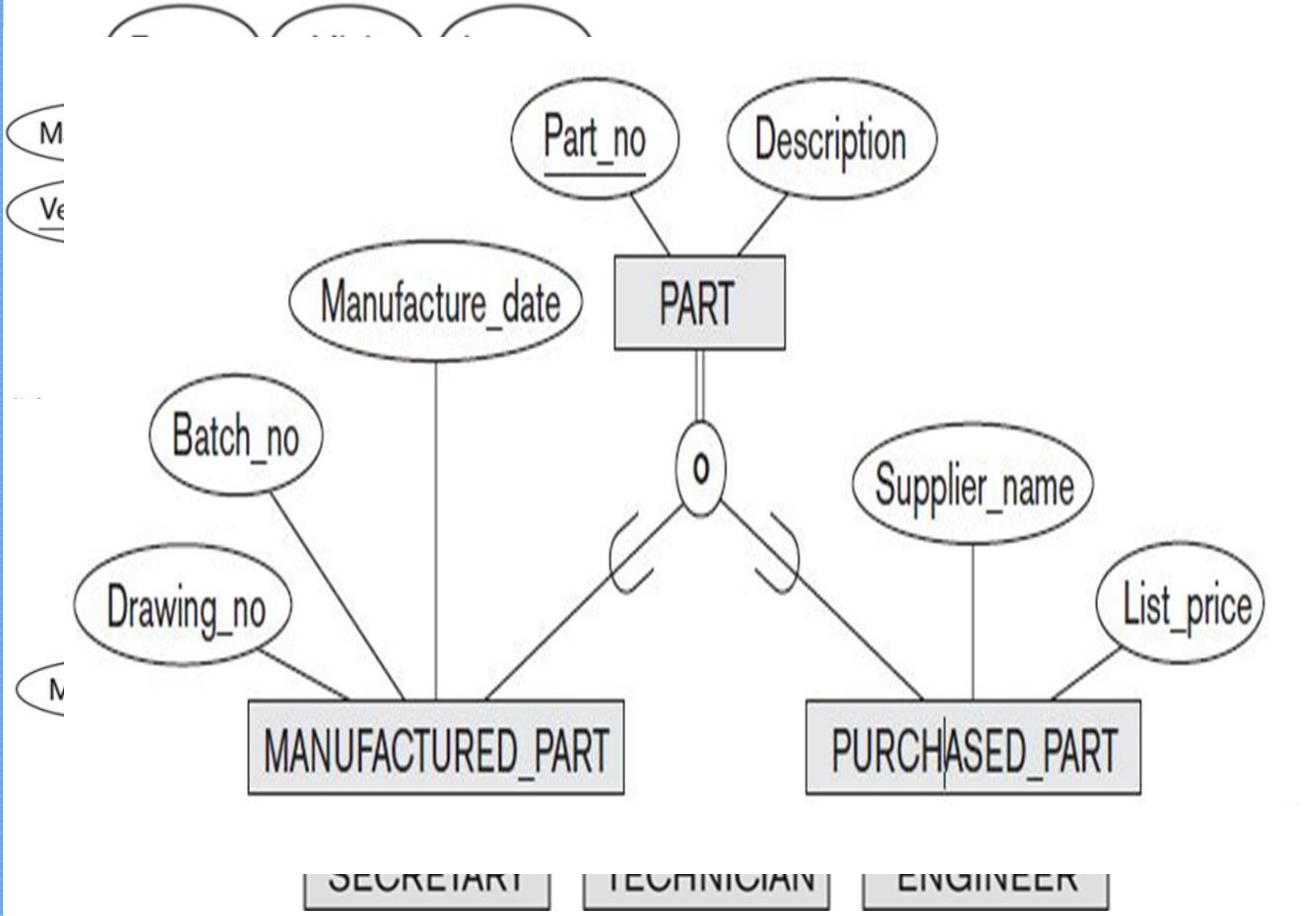
Correspondence between ER and Relational Models



ER MODEL	RELATIONAL MODEL
Entity type	*Entity* relation
1:1 or 1:N relationship type	Foreign key (or *relationship* relation)
M:N relationship type	*Relationship* relation and *two* foreign keys
n-ary relationship type	*Relationship* relation and *n* 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

Mapping
EER Model Constructs
to
Relations

Mapping of Specialization or Generalization



Step 8: Options for Mapping Specialization or Generalization

- Convert each specialization with *m* subclasses {*S*1, *S*2, ..., *Sm*} and (generalized) superclass *C*, where the attributes of *C* are {*k*, *a*1, ...*an*} and *k* is the (primary) key
- Option 8A: Multiple relations—superclass and subclasses
- Option 8B: Multiple relations—subclass relations only
- Option 8C: Single relation with one type attribute
- Option 8D: Single relation with multiple type attributes

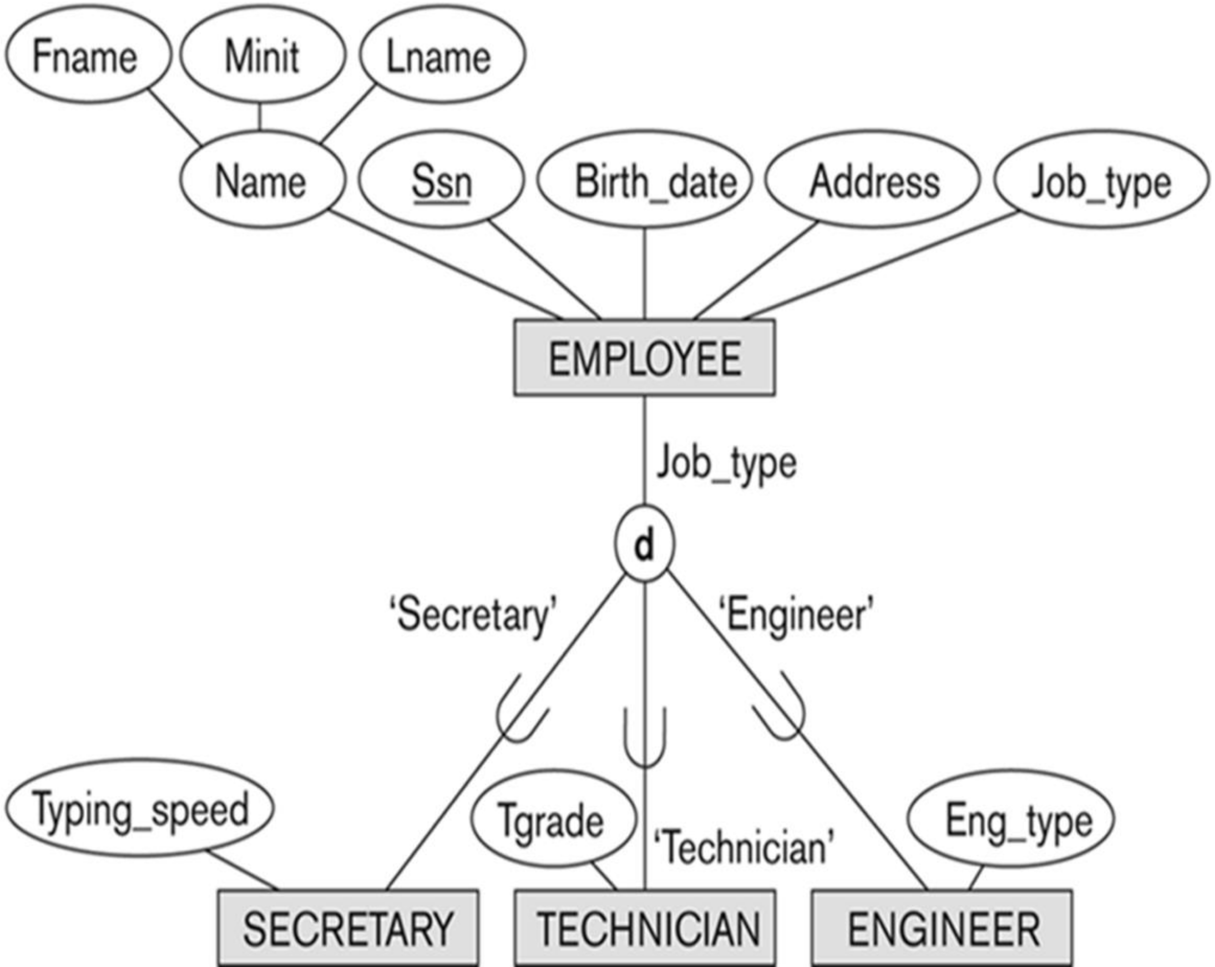
Step 8: Options for Mapping Specialization or Generalization

Option 8A: Multiple relations—superclass and subclasses

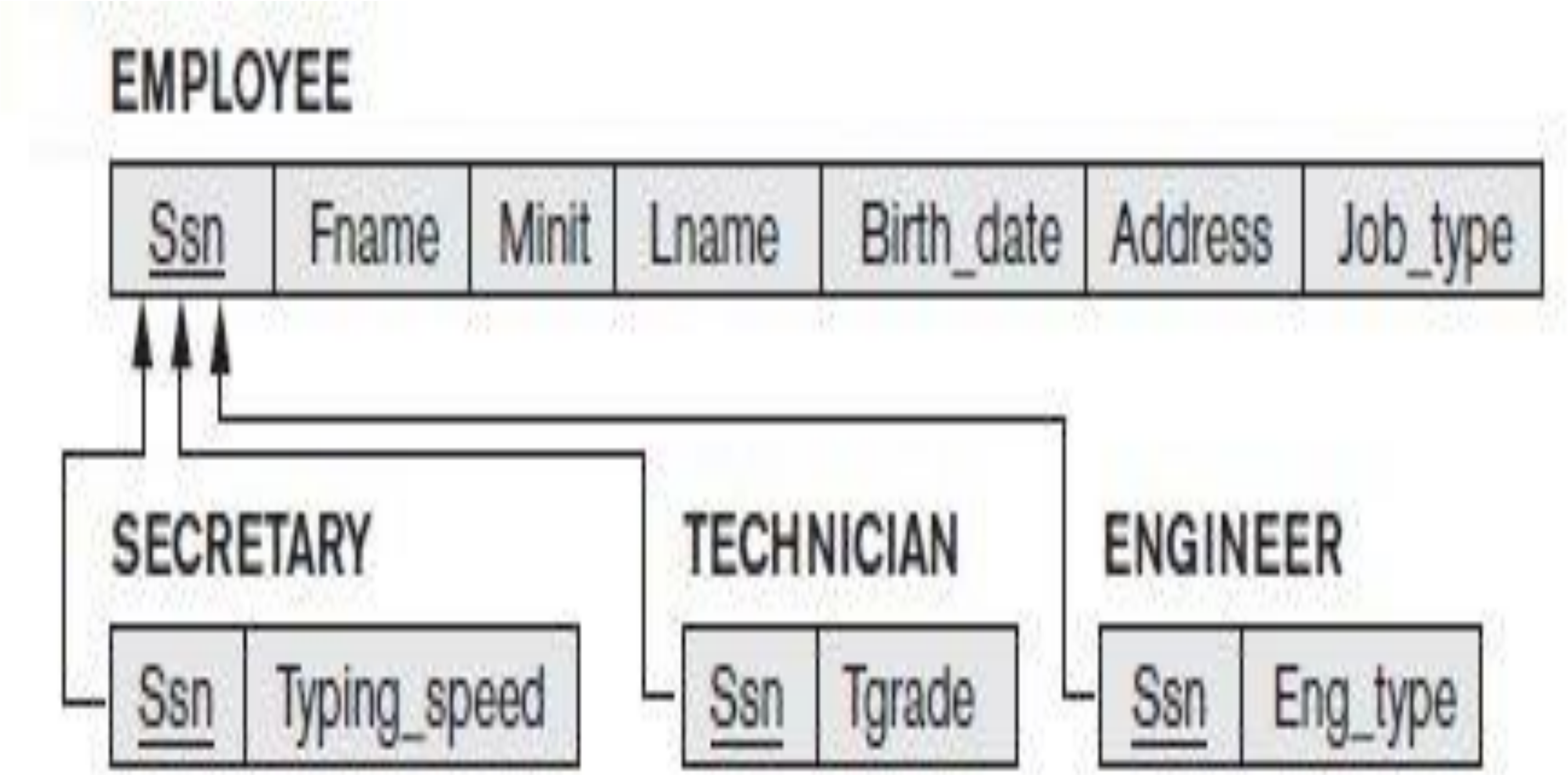
- Create a relation *L* for *C* with attributes Attrs(*L*) = {*k*, *a*1, ..., *an*} and PK(*L*) = *k*.
- Create a relation *Li* for each subclass *Si*, 1 ≤ *i* ≤ *m*, with the attributes Attrs(*Li*) = {*k*} ∪ {attributes of *Si*} and PK(*Li*) = *k*.
- This option works for any specialization (total or partial, disjoint or overlapping)

Step 8: Options for Mapping Specialization or Generalization

Option 8A: Multiple relations—superclass and subclasses



Answer



EMPLOYEE

Ssn	Fname	Minit	Lname	Birth_date	Address	Job_type

SECRETARY

Ssn	Typing_speed

TECHNICIAN

Ssn	Tgrade

ENGINEER

Ssn	Eng_type

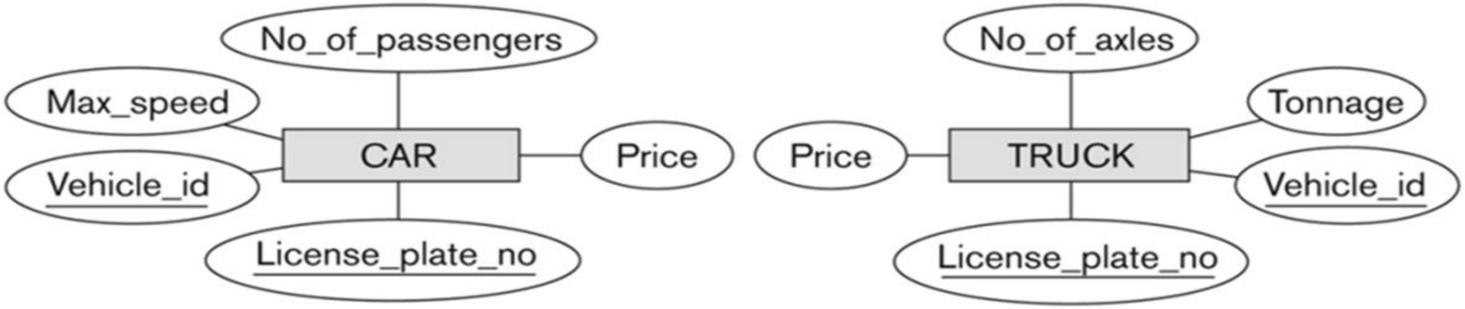
Step 8: Options for Mapping Specialization or Generalization

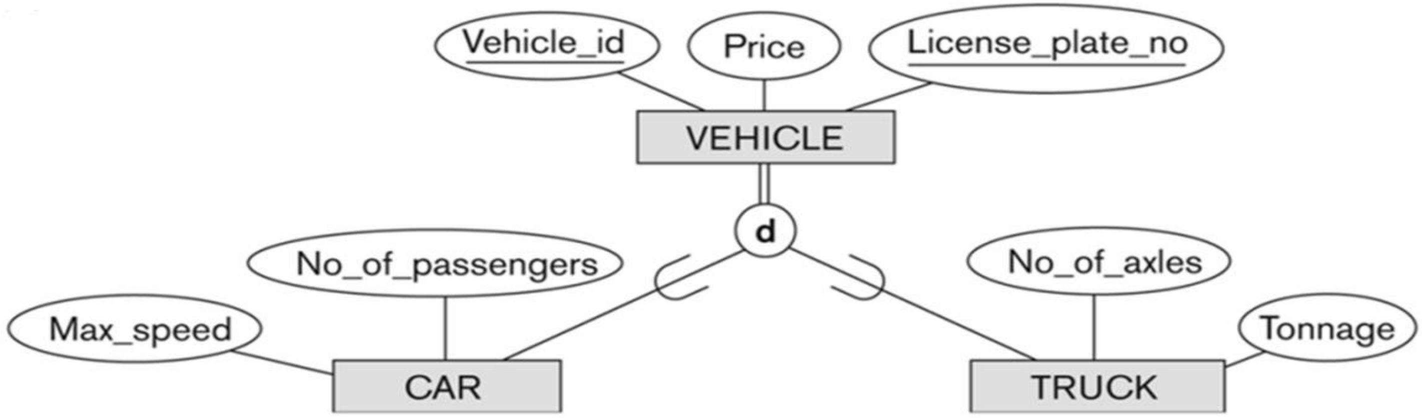
Option 8B: Multiple relations—subclass relations only

- Create a relation L_i for each subclass S_i, $1 \leq i \leq m$, with the attributes $\text{Attrs}(L_i) = \{\text{attributes of } S_i\} \cup \{k, a_1, \ldots, a_n\}$ and $\text{PK}(L_i) = k$.
- This option only works for a specialization whose subclasses are *total* (every entity in the superclass must belong to (at least) one of the subclasses).
- Additionally, it is only recommended if the specialization has the *disjointedness constraint*.
- If the specialization is *overlapping*, the same entity may be duplicated in several relations

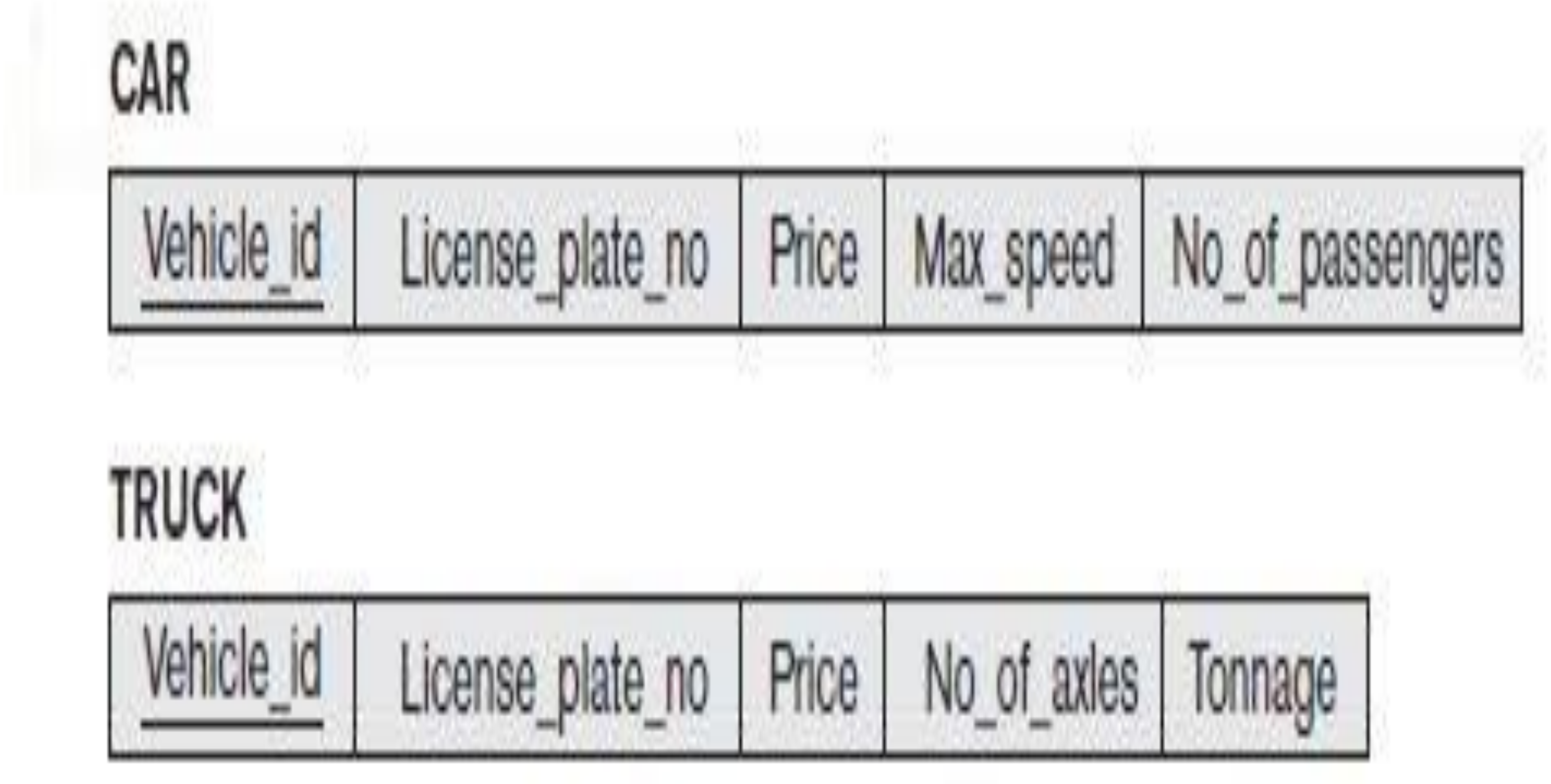
Step 8: Options for Mapping Specialization or Generalization

Option 8B: Multiple relations—subclass relations only





Answer



CAR

Vehicle_id	License_plate_no	Price	Max_speed	No_of_passengers

TRUCK

Vehicle_id	License_plate_no	Price	No_of_axles	Tonnage

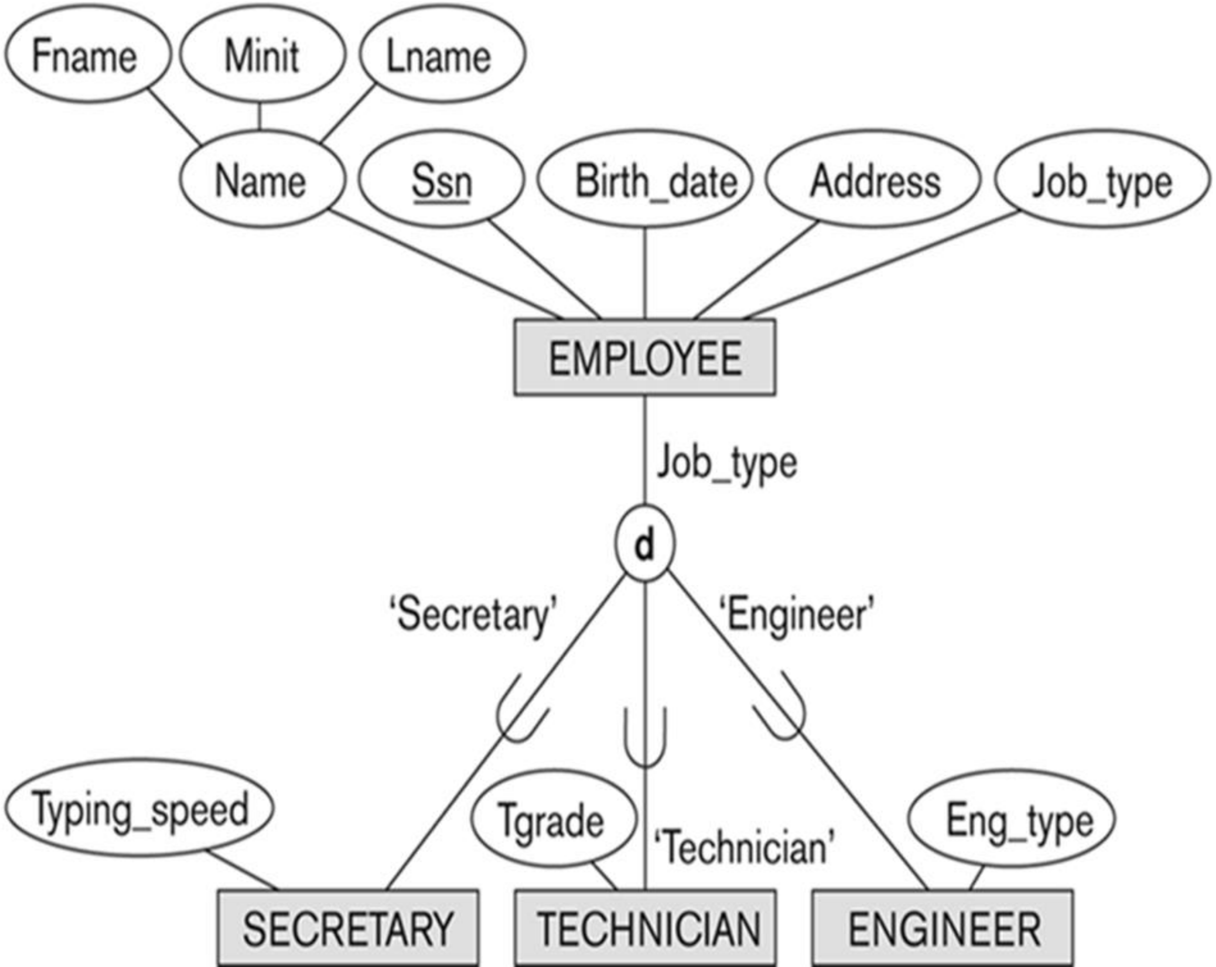
Step 8: Options for Mapping Specialization or Generalization

Option 8C: Single relation with one type attribute

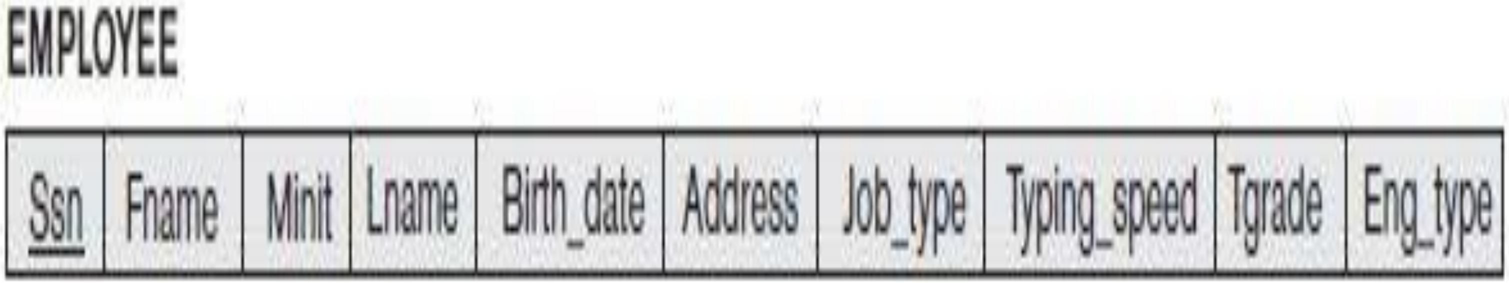
- Create a single relation *L* with attributes Attrs(*L*) = {*k*, *a*1, ..., *an*} ∪ {attributes of *S*1} ∪ ... ∪ {attributes of *Sm*} ∪ {*t*} and PK(*L*) = *k*.
- The attribute *t* is called a **type** (or **discriminating**) attribute whose value indicates the subclass to which each tuple belongs, if any.
- This option works only for a specialization whose subclasses are *disjoint*, and has the potential for generating many NULL values if many specific attributes exist in the subclasses

Step 8: Options for Mapping Specialization or Generalization

Option 8C: Single relation with one type attribute



Answer



EMPLOYEE

Ssn	Fname	Minit	Lname	Birth_date	Address	Job_type	Typing_speed	Tgrade	Eng_type

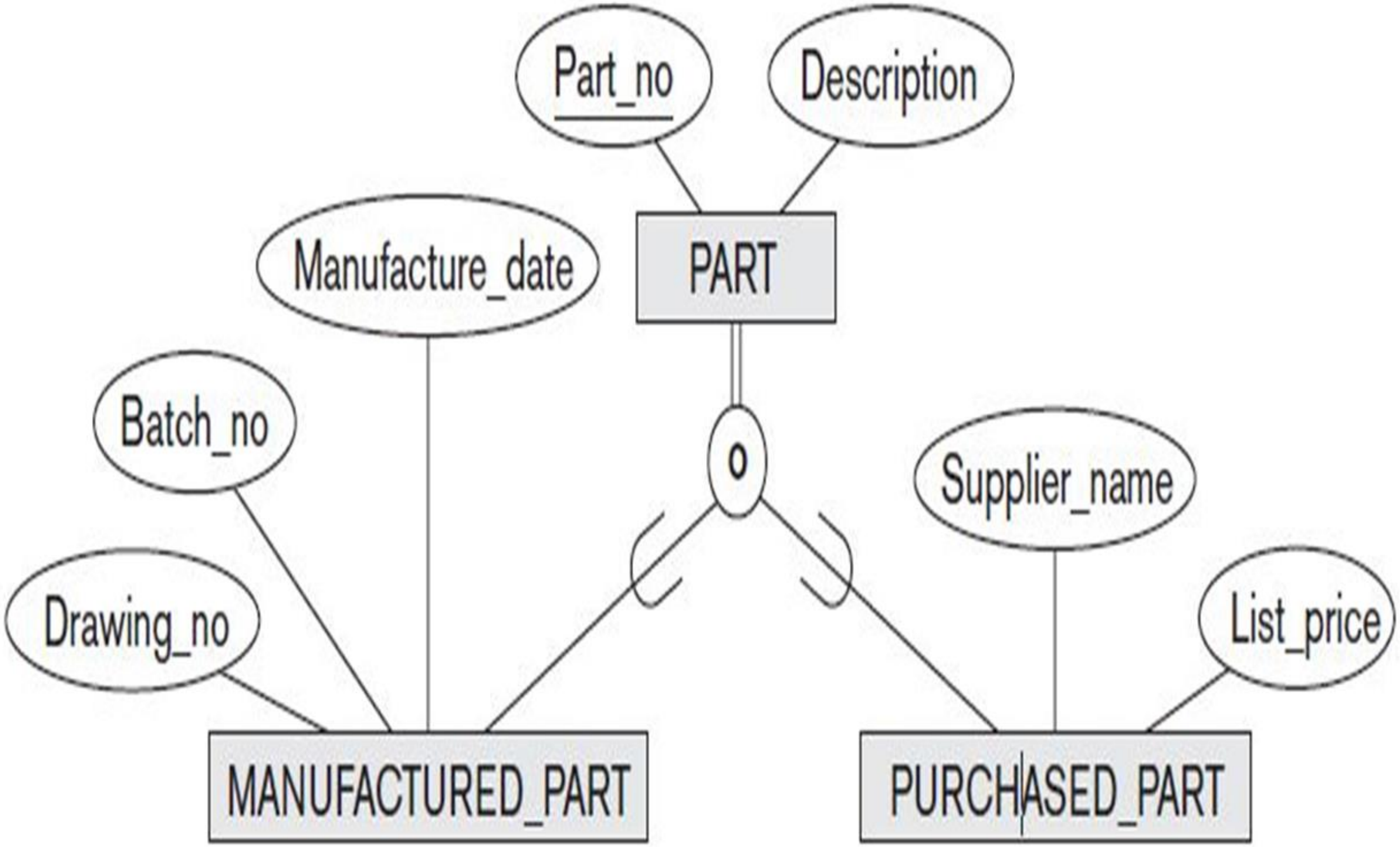
Step 8: Options for Mapping Specialization or Generalization

Option 8D: Single relation with multiple type attributes

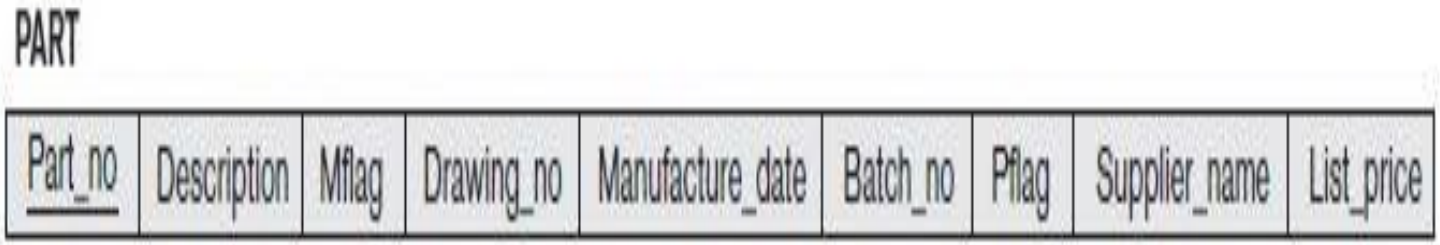
- Create a single relation schema *L* with attributes Attrs(*L*) = {*k*, *a*1, ..., *an*} ∪ {attributes of *S*1} ∪ ... ∪ {attributes of *Sm*} ∪ {*t*1, *t*2, ..., *tm*} and PK(*L*) = *k*.
- Each *ti*, 1 ≤ *i* ≤ *m*, is a **Boolean type attribute** indicating whether a tuple belongs to subclass *Si*.
- This option is used for a specialization whose subclasses are *overlapping* (but will also work for a disjoint specialization)

Step 8: Options for Mapping Specialization or Generalization

Option 8D: Single relation with multiple type attributes



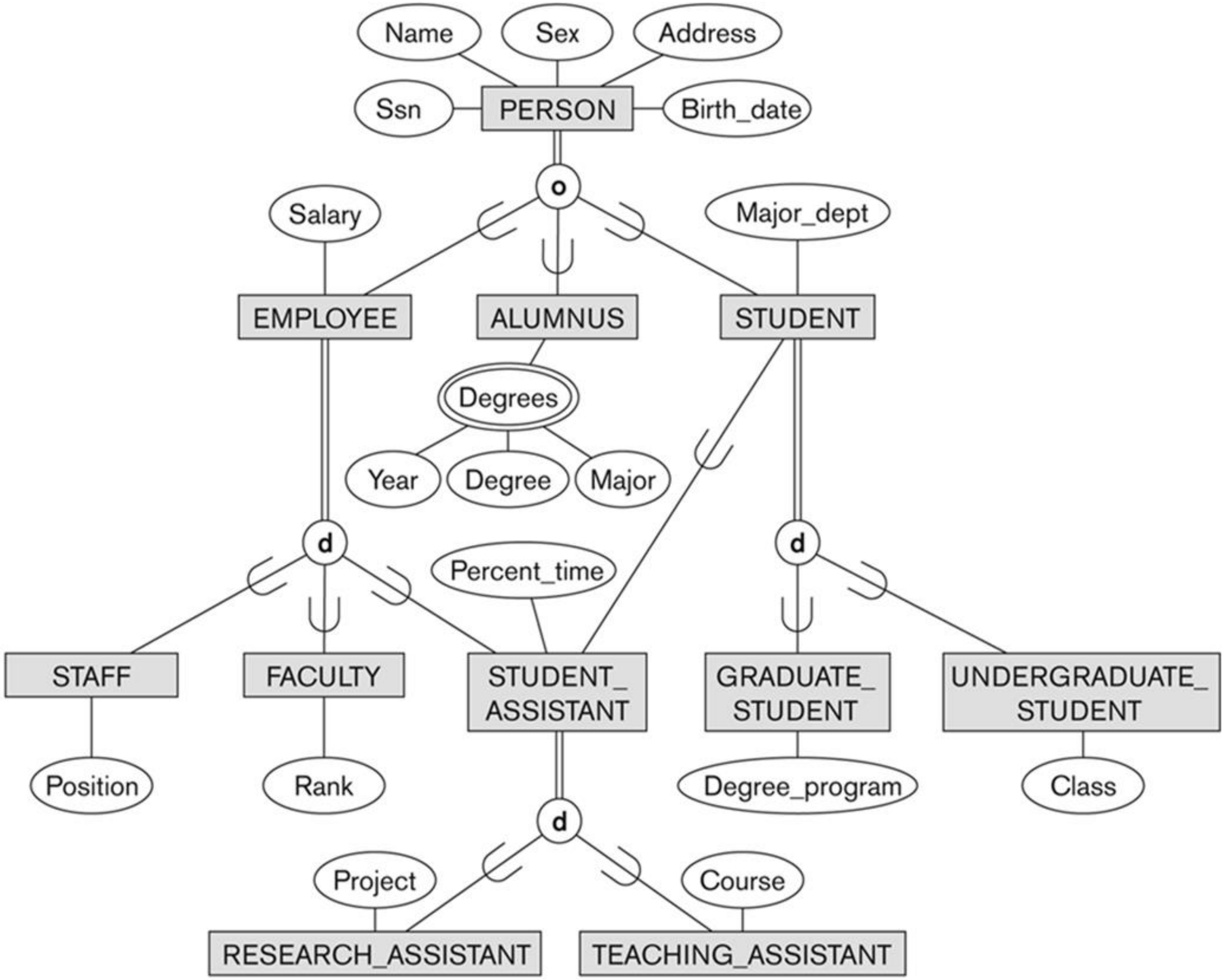
Answer



PART

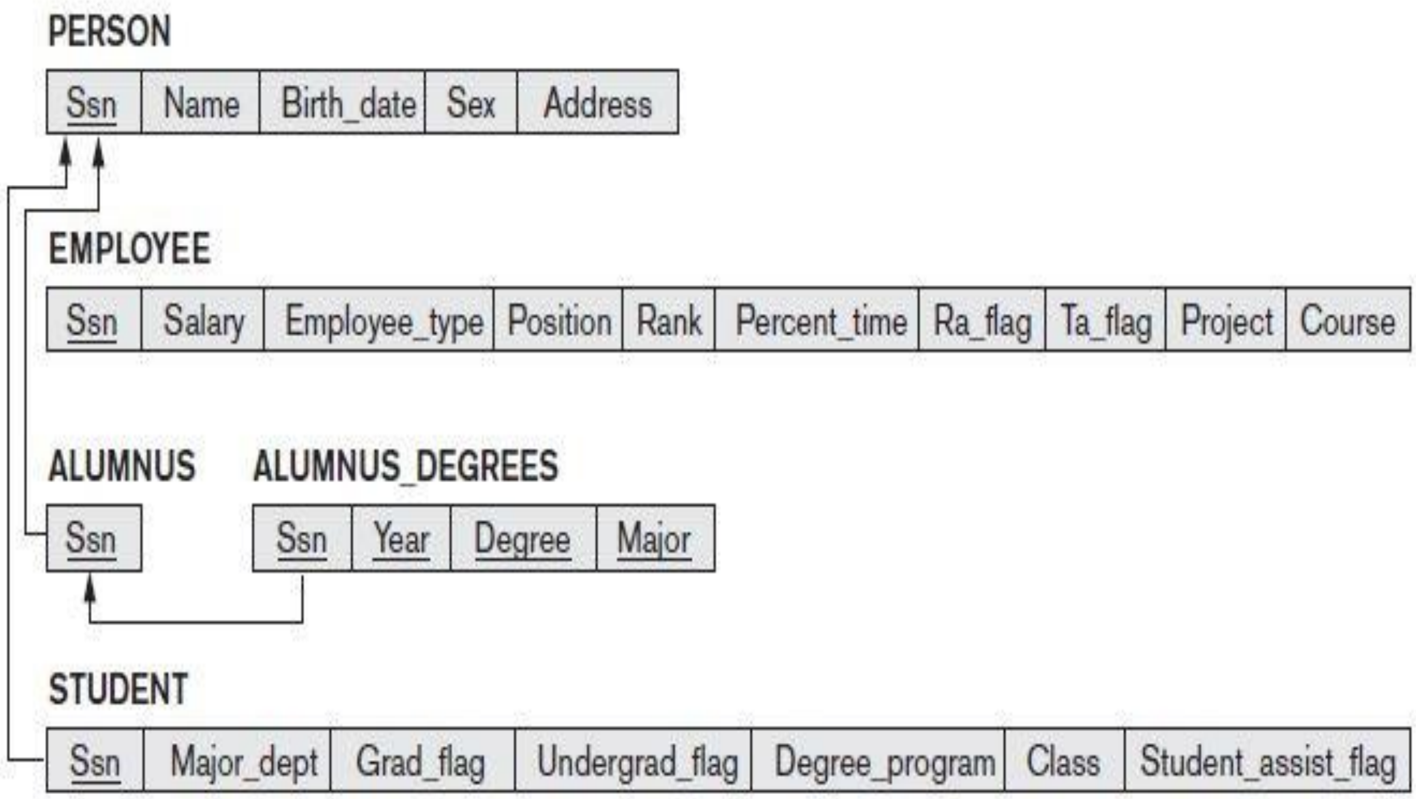
Part_no	Description	Mflag	Drawing_no	Manufacture_date	Batch_no	Pflag	Supplier_name	List_price

Specialization and Generalization Hierarchies and Lattices



A specialization lattice with multiple inheritance for a UNIVERSITY database.

Specialization and Generalization Hierarchies and Lattices



PERSON

Ssn	Name	Birth_date	Sex	Address

EMPLOYEE

Ssn	Salary	Employee_type	Position	Rank	Percent_time	Ra_flag	Ta_flag	Project	Course

ALUMNUS

Ssn

ALUMNUS_DEGREES

Ssn	Year	Degree	Major

STUDENT

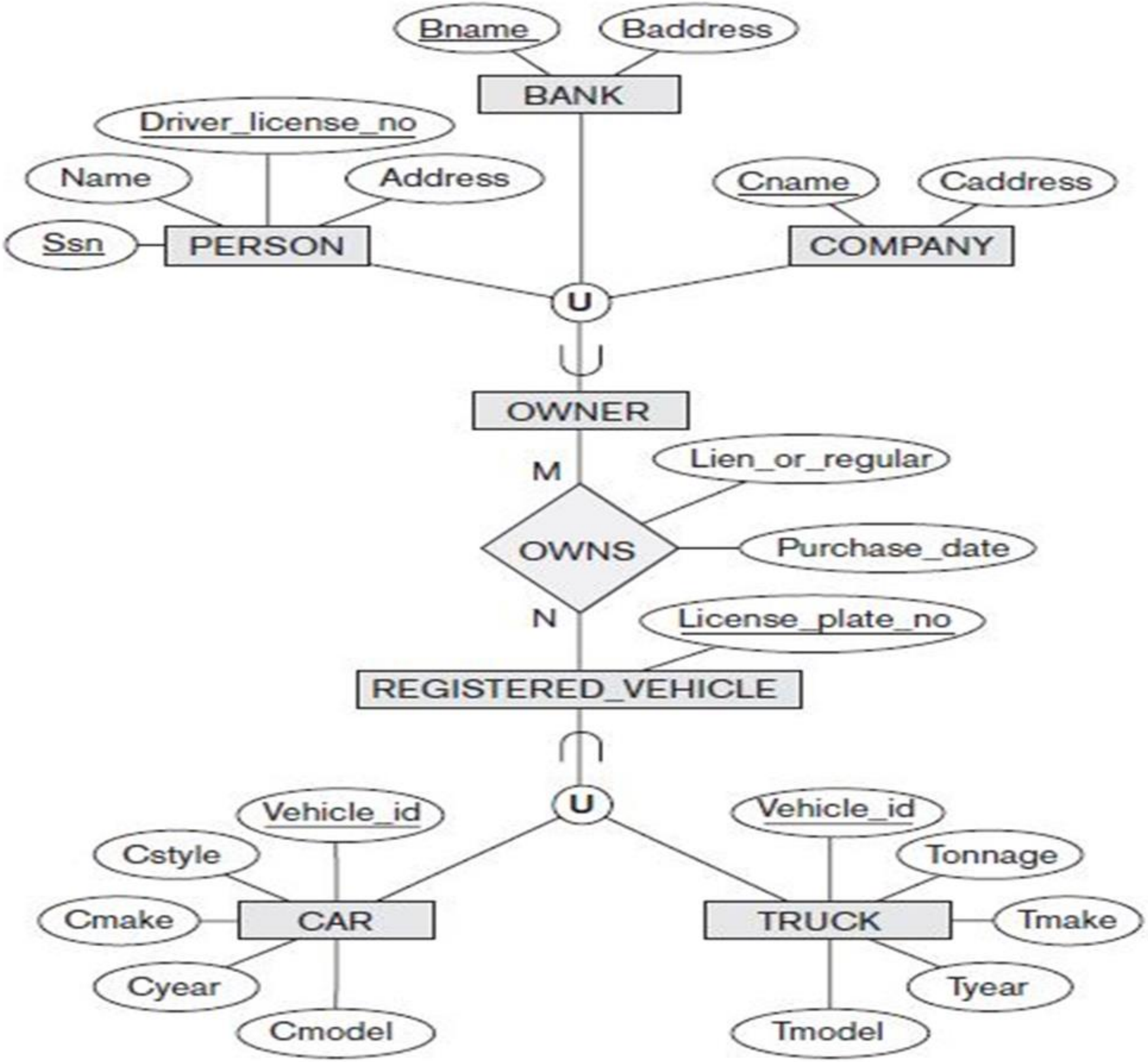
Ssn	Major_dept	Grad_flag	Undergrad_flag	Degree_program	Class	Student_assist_flag

Mapping the EER specialization lattice in Figure 8.8 using multiple options.

Mapping of Shared Subclasses (Multiple Inheritance)

- A shared subclass, such as ENGINEERING_MANAGER is a subclass of several superclasses, indicating multiple inheritance.
- These classes must all have the same key attribute; otherwise, the shared subclass would be modeled as a category (union type)
- We can apply any of the options discussed in step 8 to a shared subclass, subject to the restrictions discussed in step 8 of the mapping algorithm.
- In above figure options 8C and 8D are used for the shared subclass STUDENT_ASSISTANT.
- Option 8C is used in the EMPLOYEE relation (Employee_type attribute) and option 8D is used in the STUDENT relation (Student_assist_flag attribute)

Mapping of Categories (Union Types)



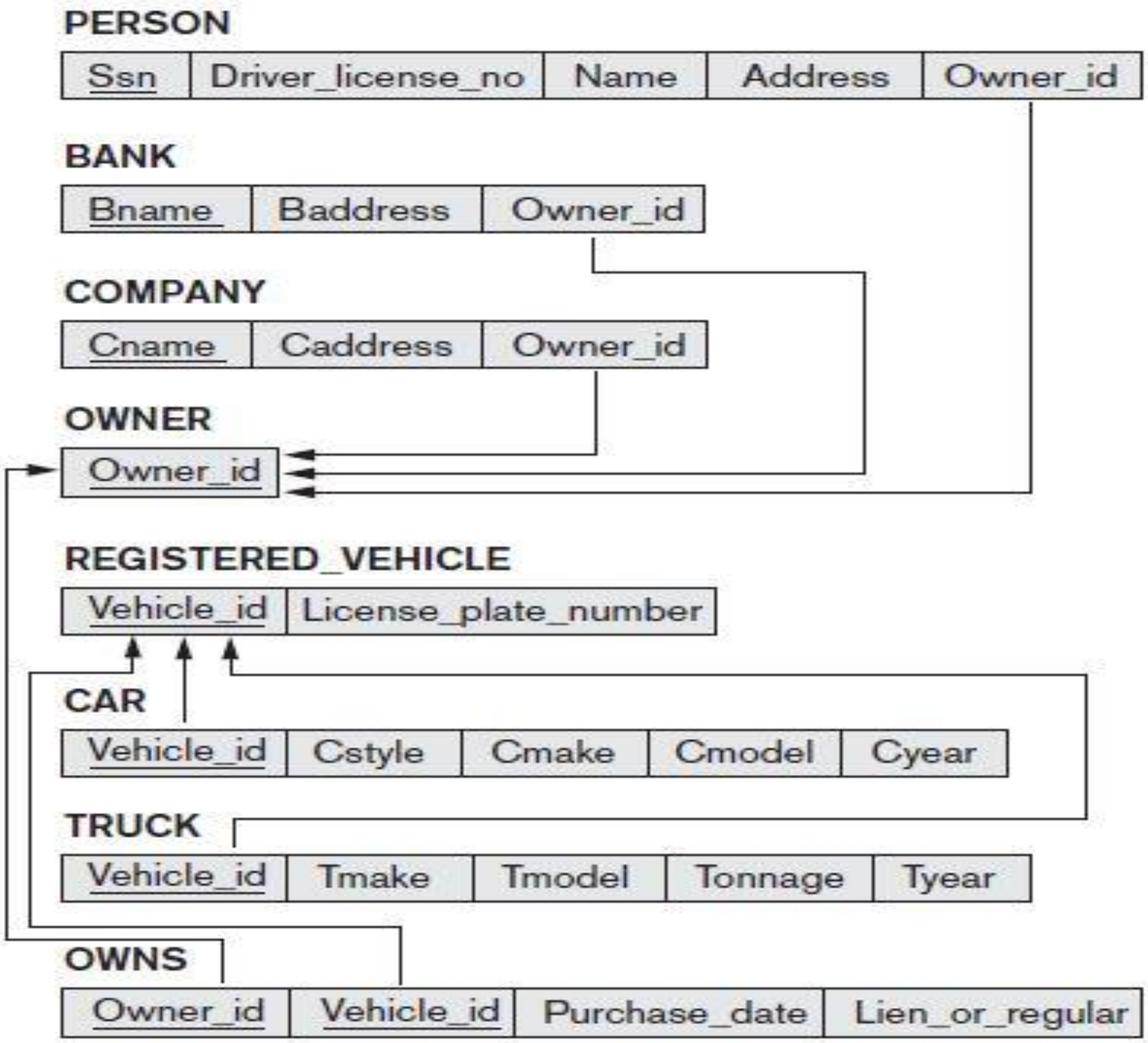
Mapping of Categories (Union Types)

- A category (or union type) is a subclass of the *union* of two or more superclasses that can have different keys because they can be of different entity types
- An example is the OWNER category which is a subset of the union of three entity types PERSON, BANK, and COMPANY.
- The other category in that figure, REGISTERED_VEHICLE, has two super classes that have the same key attribute.

Step 9: Mapping of Union Types (Categories)

- For mapping a category whose defining super classes have different keys, it is customary to specify a new key attribute, called a **surrogate key**, when creating a relation to correspond to the category.
- The keys of the defining classes are different, so we cannot use any one of them exclusively to identify all entities in the category
- For a category whose superclasses have the same key,there is no need for a surrogate key

Step 9: Mapping of Union Types (Categories)



PERSON

Ssn	Driver_license_no	Name	Address	Owner_id

BANK

Bname	Baddress	Owner_id

COMPANY

Cname	Caddress	Owner_id

OWNER

Owner_id

REGISTERED_VEHICLE

Vehicle_id	License_plate_number

CAR

Vehicle_id	Cstyle	Cmake	Cmodel	Cyear

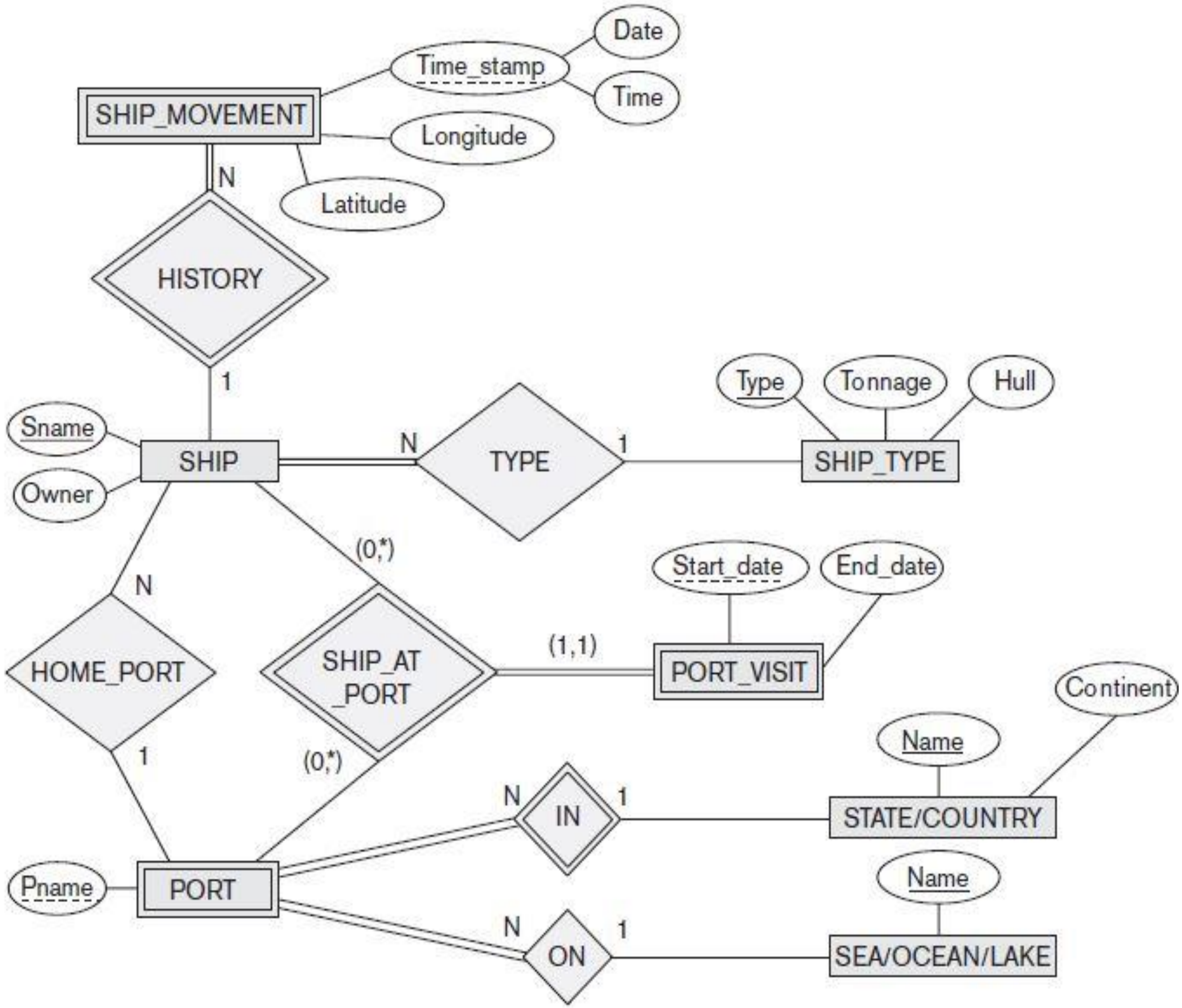
TRUCK

Vehicle_id	Tmake	Tmodel	Tonnage	Tyear

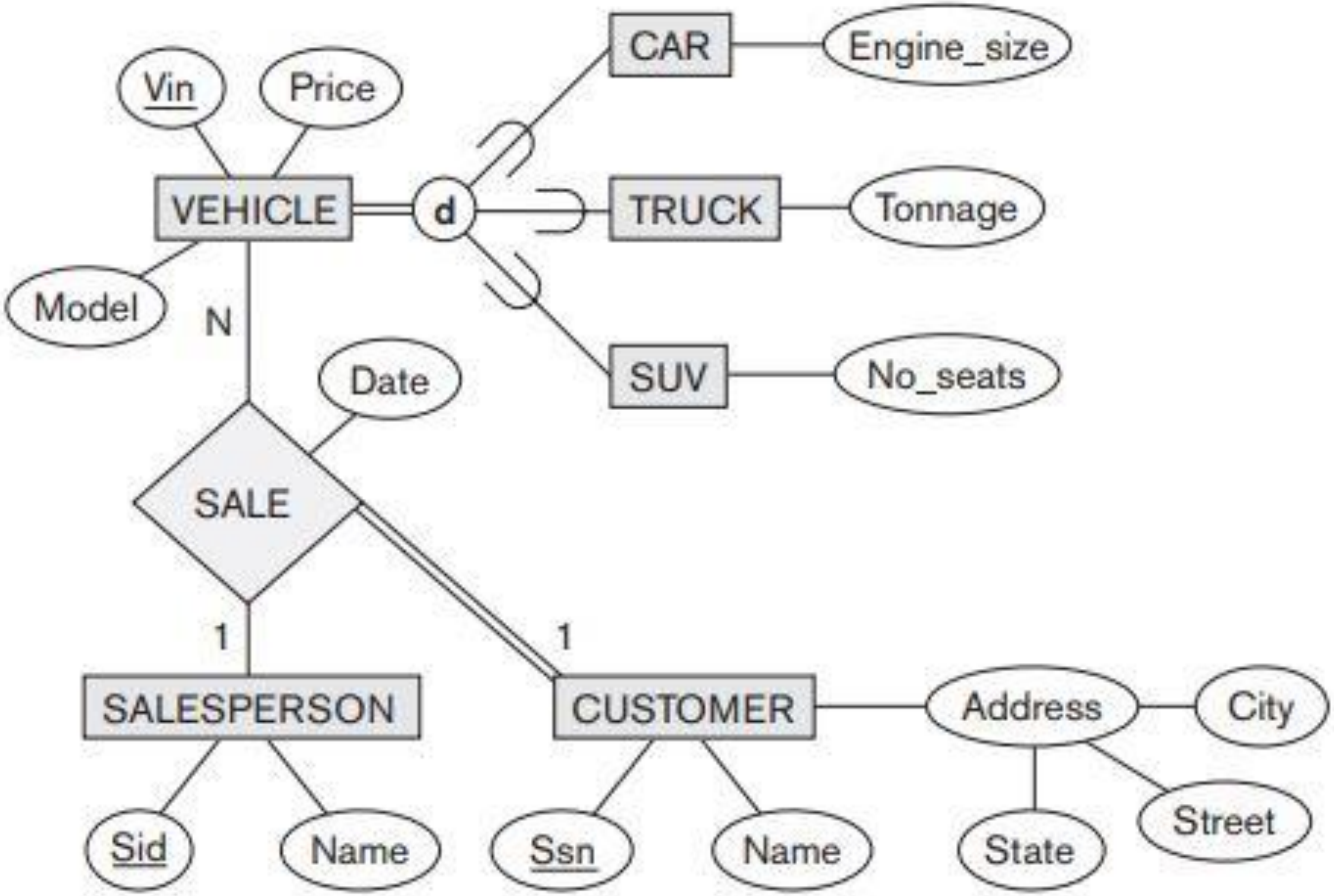
OWNS

Owner_id	Vehicle_id	Purchase_date	Lien_or_regular

Exercise


Date
Time_stamp
Time
SHIP_MOVEMENT
Longitude
N
Latitude
HISTORY
1
Type
Tonnage
Hull
Sname
SHIP
N
TYPE
1
SHIP_TYPE
Owner
(0,*)
Start_date
End_date
N
HOME_PORT
SHIP_AT
_PORT
(1,1)
PORT_VISIT
Continent
Name
1
(0,*)
N
IN
1
STATE/COUNTRY
Pname
PORT
Name
N
ON
1
SEA/OCEAN/LAKE

Exercise


CAR
Engine_size
Vin
Price
VEHICLE
d
TRUCK
Tonnage
Model
N
SUV
No_seats
Date
SALE
1
1
SALESPERSON
CUSTOMER
Address
City
Street
State
Sid
Name
Ssn
Name

Reference

- *Chapter 9 - Fundamentals of Database Systems*

(6th Edition) By Remez Elmasri & Shamkant B. Navathe

Questions ???

Thank You