

# RELATIONAL DATA MODEL

## Aims:

At the end of this group of five lectures, you should be able to understand the concepts of relational data model. You should be able to map ER models into relational schemas and specify queries in relational algebra.

Reading: Elmasri & Navathe, Chapters 5, 8 and 9

# OVERVIEW

1. Structural component of the relational model
2. Integrity component
3. EER to relational mapping algorithm
4. Operational component
5. Views

# STRUCTURAL COMPONENT OF RM

- Proposed by E.F. Codd (IBM)

"A Relational Model for Large Shared Data Banks,"  
Communications of the ACM, June 1970 (**Turing award**)

- **RELATION**: A table of values

- a **set of rows (tuples)**
- a **set of columns (attributes)**
- Each row represents an **entity or relationship**
- Each row is unique

- Degree of a relation = the number of attributes

- Cardinality of a relation = the number of tuples

# RELATION

Relation Name

**STUDENT**

Attributes

Tuples

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

**Figure 5.1**

The attributes and tuples of a relation STUDENT.

# FORMAL DEFINITIONS

- **Schema of a relation:**  $R (A_1, A_2, \dots, A_n)$   
CUSTOMER (Cust-id, Cust-name, Address, Phone)
- **Domain:** a set of valid values (**atomic!**)  
The domain of Cust-id is 6 digit numbers
- **Tuple:** an ordered set of values
- Each value is a value from the appropriate domain
- **Relation:** a set of tuples
- **Relational Database Schema:** A set  $S$  of relation schemas that belong to the same database.  $S$  is the *name* of the database.

$$S = \{R_1, R_2, \dots, R_n\}$$

# FORMAL DEFINITIONS (CONT.)

- The relation is a subset of Cartesian product of the domains
- Given  $R(A_1, A_2, \dots, A_n)$   
$$r(R) \subset \text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n)$$
- R: schema of the relation
- r of R: a specific "value" or population of R.
- R - **intension** of a relation
- r - **extension** of a relation

# FORMAL DEFINITIONS (CONT.)

- Let  $S1 = \{0,1\}$
- Let  $S2 = \{a,b,c\}$
- Let  $R \subset S1 \times S2$
- $r(R) = \{ \langle 0,a \rangle , \langle 0,b \rangle , \langle 1,c \rangle \}$   
is one possible “state”  $r$  of the relation  $R$ , with  
three tuples

# SUMMARY

<b>Informal Terms</b>	<b>Formal Terms</b>
Table	Relation
Column	Attribute
Row	Tuple
Values in a column	Domain
Table definition	Schema of a relation
Populated Table	Extension



# CHARACTERISTICS OF RELATIONS

- Ordering of tuples: not important
- Ordering of attributes in a relation schema  $R$  (and of values within each tuple): important
$$t = \langle v_1, v_2, \dots, v_n \rangle$$
- All values are considered *atomic* (indivisible)
- A special **null** value is used to represent values that are unknown or inapplicable to certain tuples
- Each relation name is unique within the database
- Each attribute in a relation has a distinct name

# OVERVIEW

1. Structural component of the relational model
2. Integrity component
  - Domain constraint
  - Key constraint
  - Entity integrity
  - Referential integrity
3. EER to relational mapping algorithm
4. Operational components
5. Views

# INTEGRITY COMPONENT

- Constraints are *conditions* that must hold on *all* valid relation instances
- **Domain** constraint: all values in a relation come from corresponding domains
- Key constraint
- Entity integrity
- Referential integrity

# KEY CONSTRAINT

- **Superkey** of R: A set of attributes SK of R such that no two tuples *in any valid relation instance*  $r(R)$  will have the same value for SK.  
For any distinct tuples  $t1$  and  $t2$  in  $r(R)$ :  
$$t1[SK] \neq t2[SK]$$
- **Key** of R: A "minimal" superkey
  - Unique
  - Irreducible
- If a relation has *several* **candidate keys**, one is chosen to be the **primary key**.
- The primary key attributes are *underlined*.

# KEY CONSTRAINT

**CAR**

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

**Figure 5.4**

The CAR relation, with two candidate keys: License\_number and Engine\_serial\_number.

## EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

## DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

## DEPT\_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

## PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

## WORKS\_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

## DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

**Figure 5.5**  
Schema diagram for  
the COMPANY  
relational database  
schema.

# ENTITY INTEGRITY

- No primary key value can be null  
 $t[\text{PK}] \neq \text{null}$  for any tuple  $t$  in  $r(R)$
- Note: Other attributes of  $R$  may be similarly constrained to disallow null values, even though they are not members of the primary key

# REFERENTIAL INTEGRITY

- A constraint involving *two* relations
- Used to specify a *relationship* among tuples in two relations: the **referencing relation** and the **referenced relation**.
- Tuples in the *referencing relation*  $R_1$  have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the *referenced relation*  $R_2$
- A tuple  $t_1$  in  $R_1$  is said to **reference** a tuple  $t_2$  in  $R_2$  if  $t_1[\text{FK}] = t_2[\text{PK}]$ .



# REFERENTIAL INTEGRITY CONSTRAINT

The value of the foreign key FK of the **referencing relation**  $R_1$  can be either:

1. a value of an existing primary key value of the corresponding primary key PK in the **referenced relation**  $R_2$
2. a null

**Figure 5.6**

One possible database state for the COMPANY relational database schema.

**EMPLOYEE**

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

**DEPARTMENT**

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

**DEPT\_LOCATIONS**

<u>Dnumber</u>	<u>Dlocation</u>
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

**WORKS\_ON**

<u>Essn</u>	<u>Pno</u>	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**

<u>Pname</u>	<u>Pnumber</u>	<u>Plocation</u>	<u>Dnum</u>
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

**DEPENDENT**

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

**Figure 5.7**

Referential integrity constraints displayed on the COMPANY relational database schema.

**EMPLOYEE**

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

**DEPARTMENT**

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

**DEPT\_LOCATIONS**

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

**PROJECT**

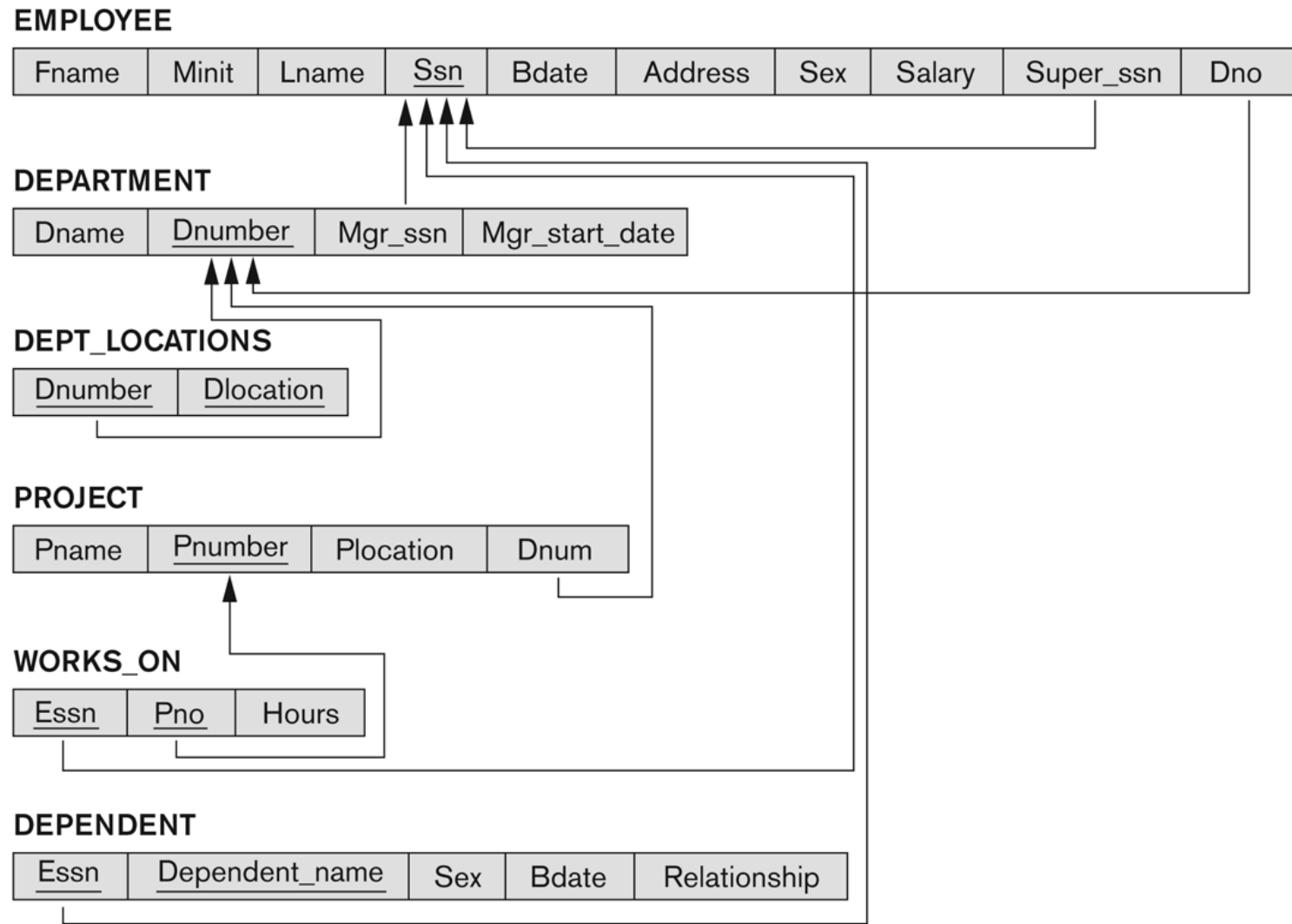
Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

**WORKS\_ON**

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

**DEPENDENT**

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------



# OTHER TYPES OF CONSTRAINTS

## Semantic Integrity Constraints

- based on application semantics and cannot be expressed by the model per se
- e.g., “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”
- *A constraint specification language* may have to be used to express these
- SQL-99: triggers and ASSERTIONS

# IN-CLASS EXERCISE

Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

STUDENT(Snumber, Name, Major, Bdate)

COURSE(Course#, Cname, Dept)

ENROLL(StudNo, Course#, Semester, Grade)

BOOK\_ADOPTION(Course#, Semester, Book\_ISBN)

TEXT(Book\_ISBN, Book\_Title, Publisher, Author)

Specify the foreign keys for this schema.

# OVERVIEW

1. Structural component of the relational model
2. Integrity component
3. **EER to relational mapping algorithm**
4. Operational component
5. Views

# OUTLINE

## ○ **ER-to-Relational Mapping Algorithm**

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types

Step 4: Mapping of Binary 1:N Relationship Types

Step 5: Mapping of Binary M:N Relationship Types

Step 6: Mapping of Multivalued attributes

Step 7: Mapping of N-ary Relationship Types

## ○ **Mapping EER Model Constructs to Relations**

Step 8: Options for Mapping Specialization or Generalization

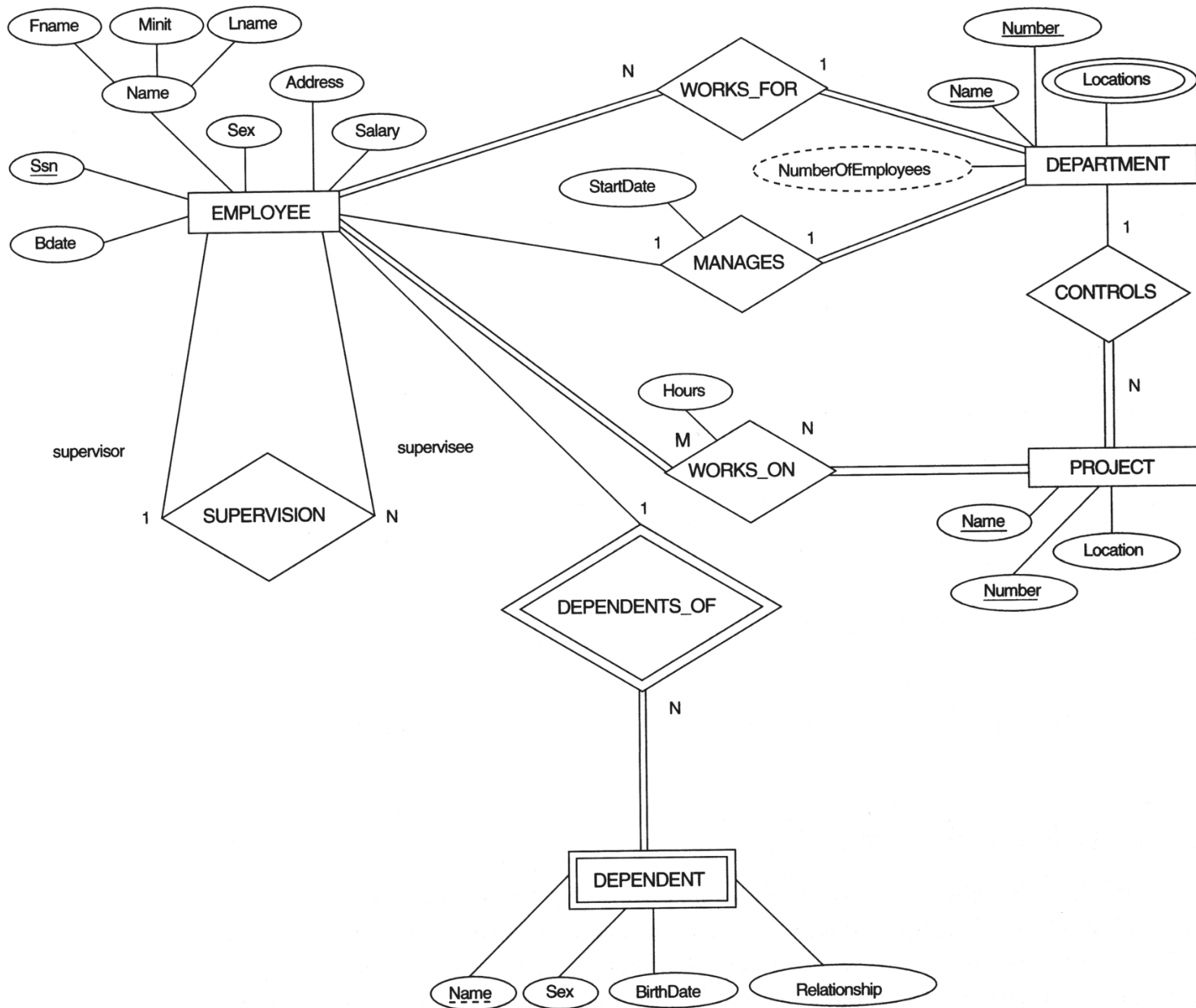
Step 9: Mapping of Union Types (Categories)

# ER-TO-RELATIONAL MAPPING ALGORITHM

## ○ 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.
- Choose one of the key attributes of E as the primary key for R. If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.





# ER-TO-RELATIONAL MAPPING ALGORITHM (CONT)

## ○ 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).
- 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.

# ER-TO-RELATIONAL MAPPING (CONT)

## ○ Step 3: Mapping of Binary 1:1 Relationship Types

- Identify the relations S and T that correspond to the entity types participating in R.
- Choose one of the relations and include a foreign key in S the primary key of T. It is better to choose an entity type with *total participation* in R.
- Add attributes of the relationship type to the same relation.

# ER-TO-RELATIONAL MAPPING (CONT)

## ○ Step 4: Mapping of Binary 1:N Relationship Types

- For each regular binary 1:N relationship type R, identify the relation S that represent 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.
- Include any simple attributes of the 1:N relationship type as attributes of S.

# ER-TO-RELATIONAL MAPPING (CONT)

## ○ Step 5: Mapping of Binary M:N Relationship Types

- For each regular 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

# ER-TO-RELATIONAL MAPPING (CONT)

## ○ Step 6: Mapping of Multivalued attributes

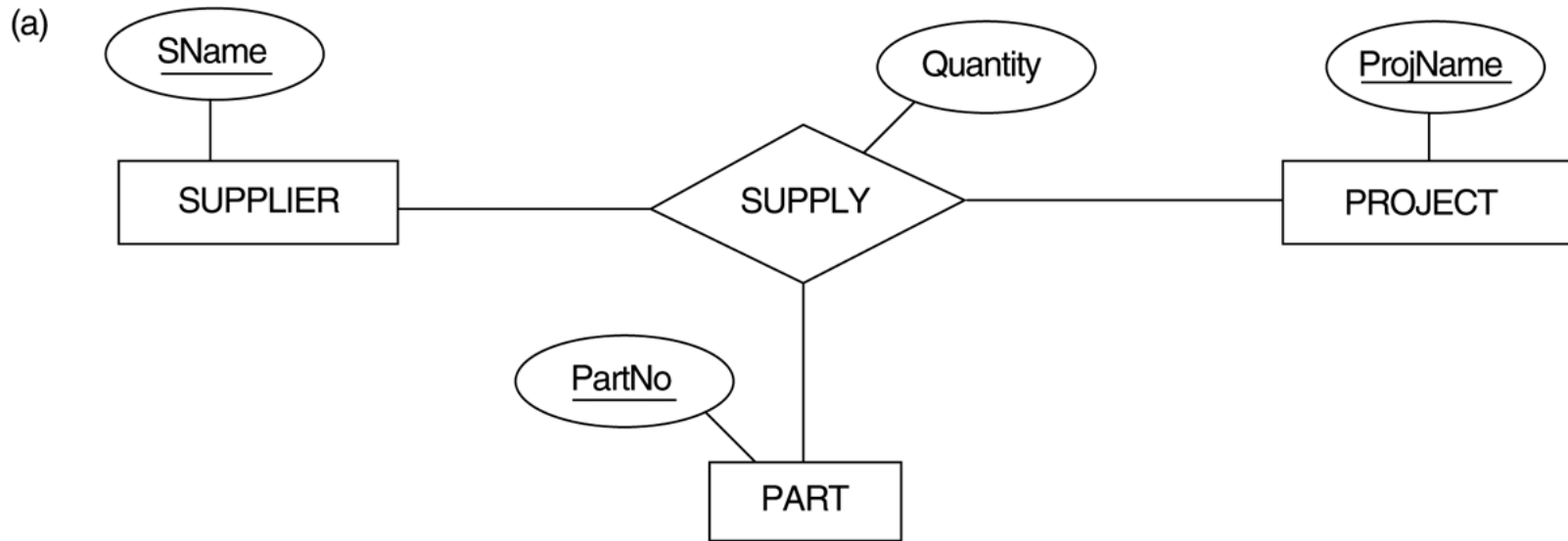
- For each multivalued attribute A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.
- The primary key of R is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

# ER-TO-RELATIONAL MAPPING (CONT)

## ○ 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.

# MAPPING OF N-ARY RELATIONSHIP TYPES



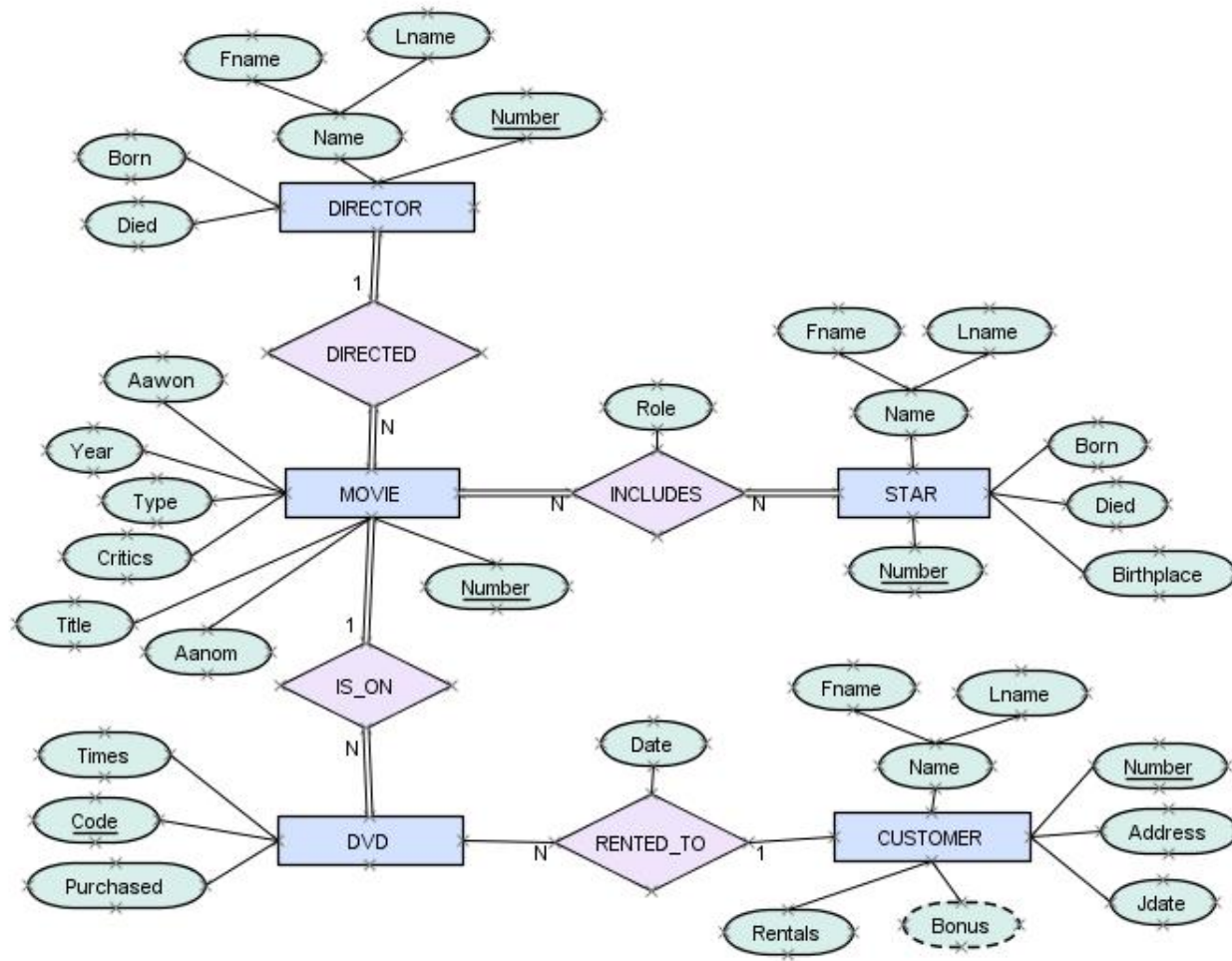


# SUMMARY OF MAPPING CONSTRUCTS AND CONSTRAINTS

*Table 8.1 Correspondence between ER and Relational Models*

ER Model	Relational Model
Entity type	“Entity” relation
1:1 or 1:N relationship type	Foreign key
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

# EXAMPLE: THE MOVIES DATABASE



# MAPPING EER MODEL CONSTRUCTS TO RELATIONS

## ○ Step 8: Options for Mapping Specialization/Generalization

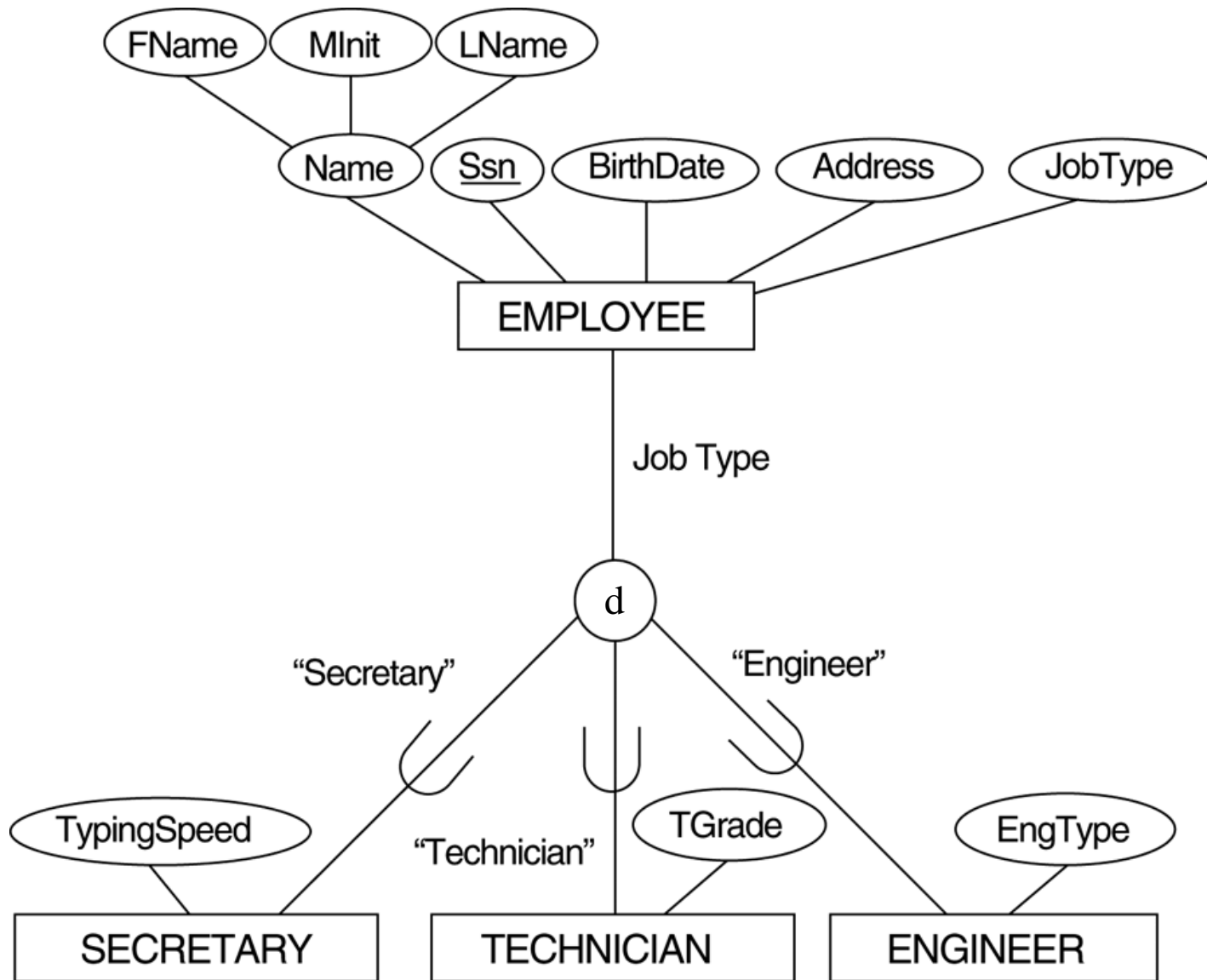
Convert each specialization with  $m$  subclasses  $\{S_1, S_2, \dots, S_m\}$  and generalized superclass  $C$ , where the attributes of  $C$  are  $\{k, a_1, \dots, a_n\}$  and  $k$  is the (primary) key, into relational schemas using one of the four following options:

### **Option 8A: Multiple relations - superclass and subclasses**

Create a relation  $L$  for  $C$  with attributes  $\text{Attrs}(L) = \{k, a_1, \dots, a_n\}$  and  $\text{PK}(L) = k$ . Create a relation  $L_i$  for each subclass  $S_i$ ,  $1 < i < m$ , with the attributes  $\text{Attrs}(L_i) = \{k\} \cup \{\text{attributes of } S_i\}$  and  $\text{PK}(L_i) = k$ . This option works **for any specialization**

### **Option 8B: Multiple relations - subclass relations only**

Create a relation  $L_i$  for each subclass  $S_i$ ,  $1 < i < m$ , with the attributes  $\text{Attr}(L_i) = \{\text{attributes of } S_i\} \cup \{k, a_1, \dots, a_n\}$  and  $\text{PK}(L_i) = k$ . This option only works for **total** specializations



## FIGURE 8.5a - Mapping specialization using option 8A

(a)

EMPLOYEE						
<u>SSN</u>	FName	MInit	LName	BirthDate	Address	JobType

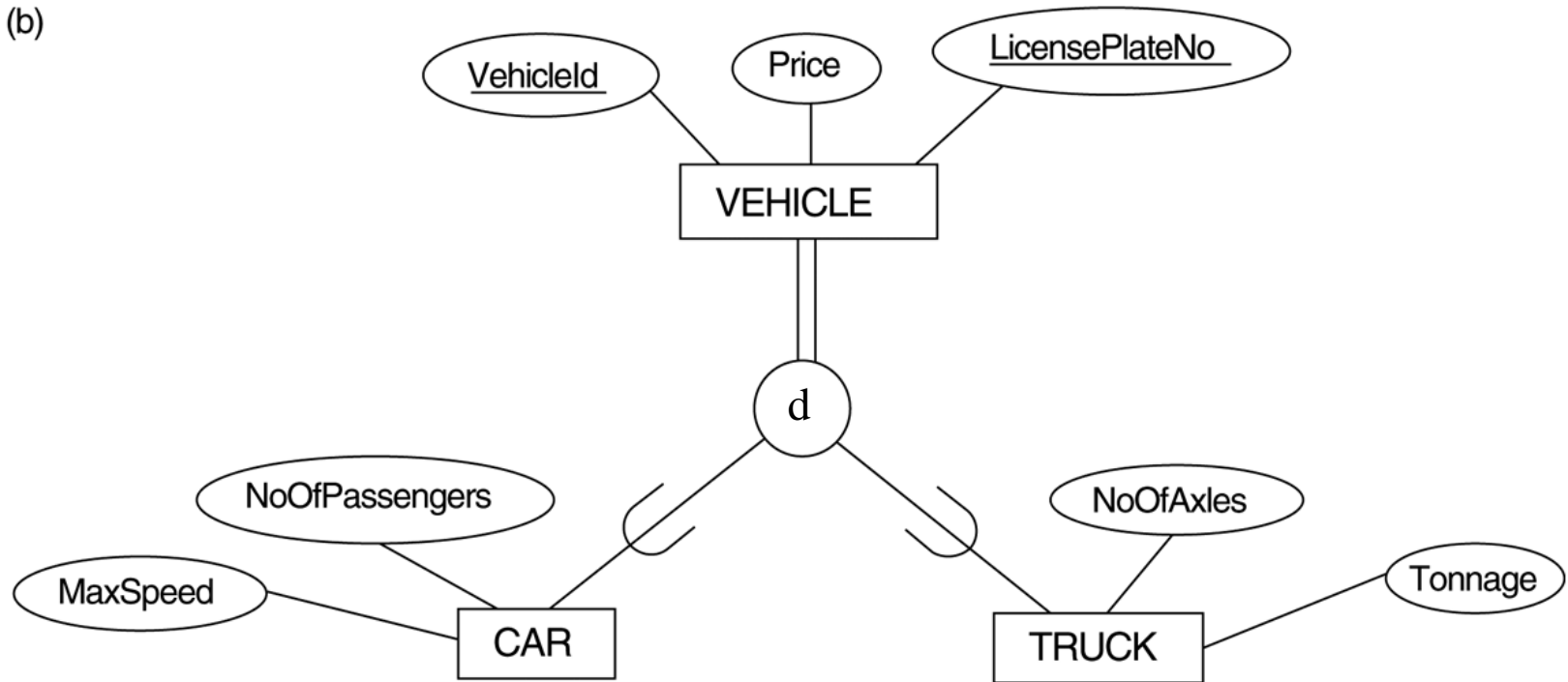
SECRETARY	
<u>SSN</u>	TypingSpeed

TECHNICIAN	
<u>SSN</u>	TGrade

ENGINEER	
<u>SSN</u>	EngType

## OPTION 8B

(b)



(b)

CAR

<u>VehicleId</u>	LicensePlateNo	Price	MaxSpeed	NoOfPassengers
------------------	----------------	-------	----------	----------------

TRUCK

<u>VehicleId</u>	LicensePlateNo	Price	NoOfAxles	Tonnage
------------------	----------------	-------	-----------	---------

# MAPPING EER MODEL CONSTRUCTS TO RELATIONS (CONT)

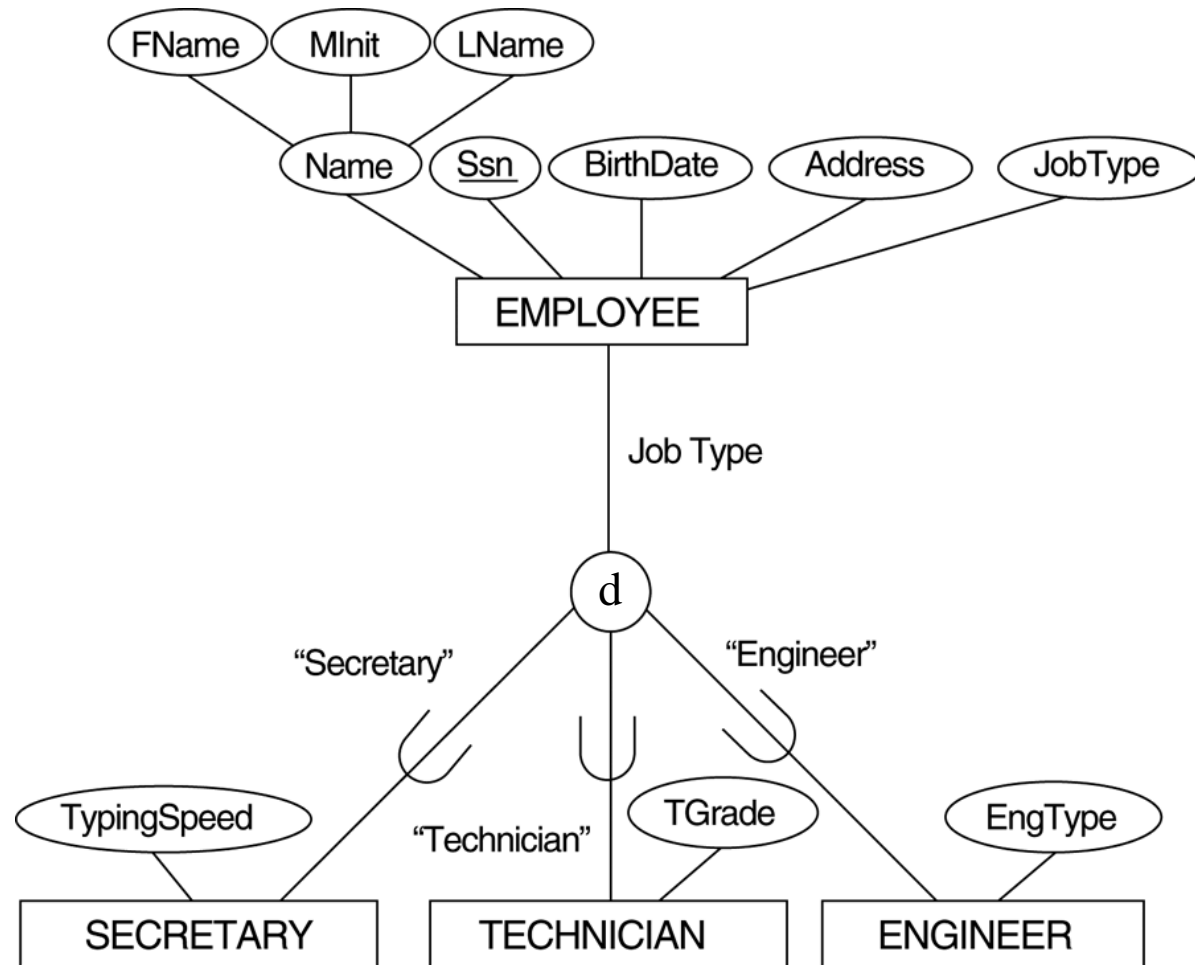
## Option 8C: Single relation with one type attribute

Create a single relation  $L$  with attributes  $\text{Attrs}(L) = \{k, a_1, \dots, a_n\} \cup \{\text{attributes of } S_1\} \cup \dots \cup \{\text{attributes of } S_m\} \cup \{t\}$  and  $\text{PK}(L) = k$ . The attribute  $t$  is called a **type** (or discriminating) attribute that indicates the subclass to which each tuple belongs

## Option 8D: Single relation with multiple type attributes

Create a single relation schema  $L$  with attributes  $\text{Attrs}(L) = \{k, a_1, \dots, a_n\} \cup \{\text{attributes of } S_1\} \cup \dots \cup \{\text{attributes of } S_m\} \cup \{t_1, t_2, \dots, t_m\}$  and  $\text{PK}(L) = k$ . Each  $t_i$ ,  $1 < i < m$ , is a Boolean type attribute indicating whether a tuple belongs to the subclass  $S_i$ .

## OPTION 8C



COSC265, 2019

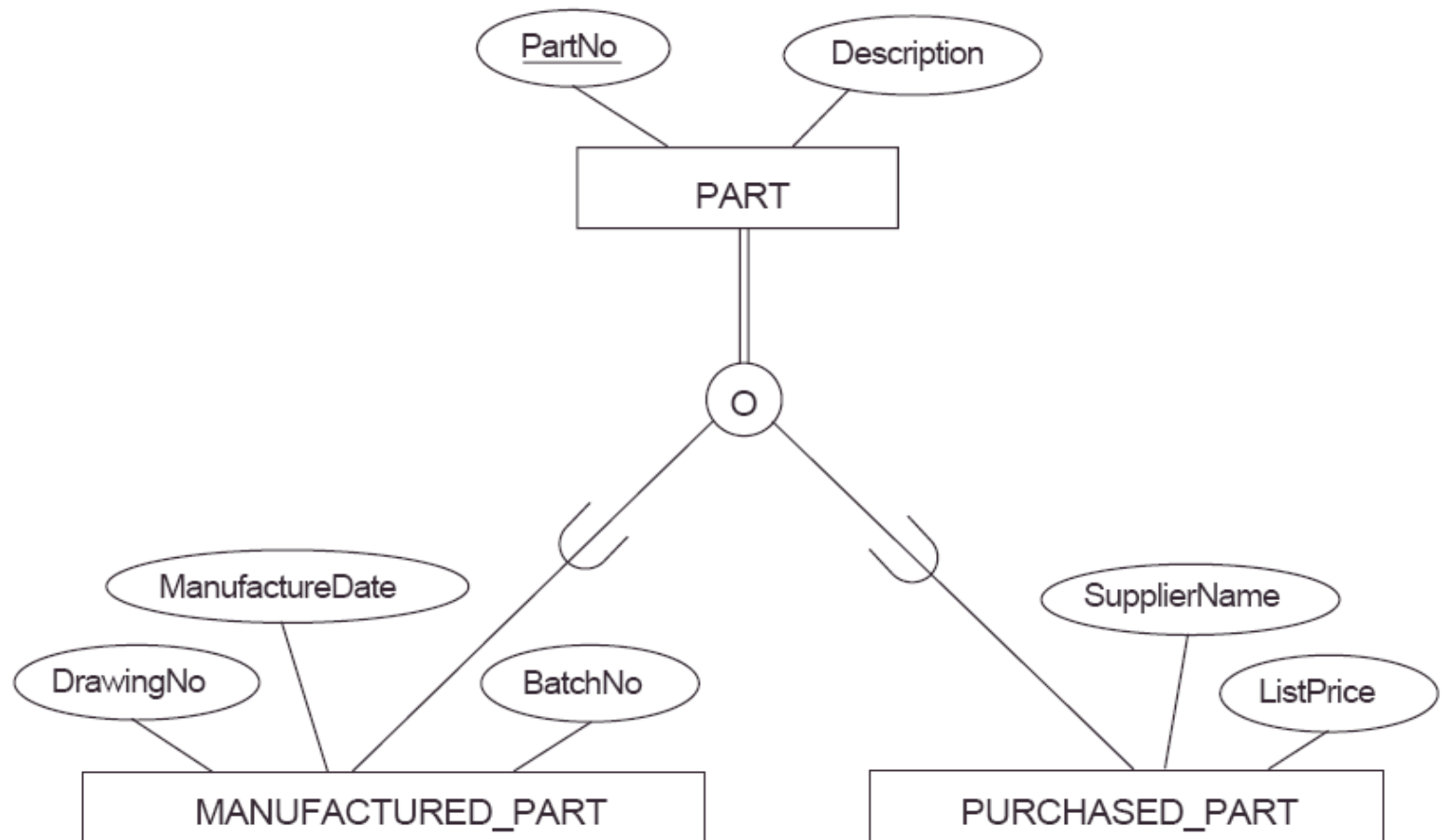
(c)

EMPLOYEE

<u>SSN</u>	FName	MInit	LName	BirthDate	Address	JobType	TypingSpeed	TGrade	EngType
------------	-------	-------	-------	-----------	---------	---------	-------------	--------	---------



## OPTION 8D



COSC265, 2019

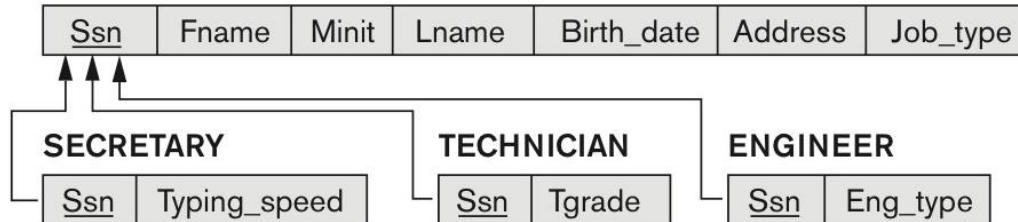
(d)

PART

<u>PartNo</u>	Description	MFlag	DrawingNo	ManufactureDate	BatchNo	PFlag	SupplierName	ListPrice
---------------	-------------	-------	-----------	-----------------	---------	-------	--------------	-----------

# FIG. 9.5: DIFFERENT OPTIONS FOR MAPPING GENERALIZATION HIERARCHIES - SUMMARY

## (a) EMPLOYEE



## (b) CAR

<u>Vehicle_id</u>	License_plate_no	Price	Max_speed	No_of_passengers
-------------------	------------------	-------	-----------	------------------

## TRUCK

<u>Vehicle_id</u>	License_plate_no	Price	No_of_axles	Tonnage
-------------------	------------------	-------	-------------	---------

## (c) EMPLOYEE

<u>Ssn</u>	Fname	Minit	Lname	Birth_date	Address	Job_type	Typing_speed	Tgrade	Eng_type
------------	-------	-------	-------	------------	---------	----------	--------------	--------	----------

## (d) PART

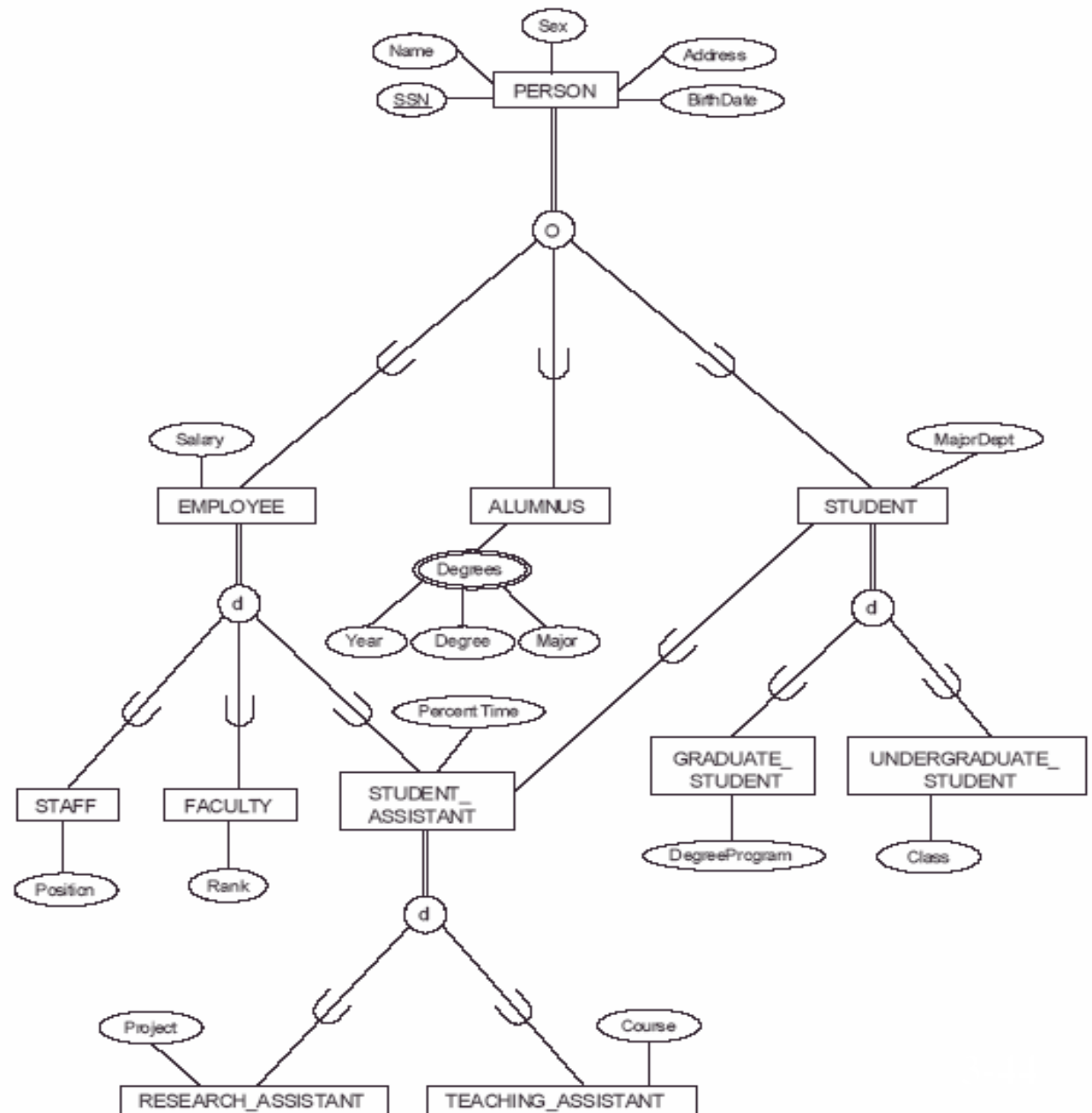
<u>Part_no</u>	Description	Mflag	Drawing_no	Manufacture_date	Batch_no	Pflag	Supplier_name	List_price
----------------	-------------	-------	------------	------------------	----------	-------	---------------	------------

# MAPPING OF SHARED SUBCLASSES

A shared subclass is a subclass of several classes, indicating multiple inheritance. These classes must all have the same key attribute; otherwise, the shared subclass would be modeled as a category.

We can apply any of the options discussed in Step 8 to a shared subclass, subject to the restriction discussed in Step 8 of the mapping algorithm.

# MULTIPLE INHERITANCE



## FIGURE 8.6

### MAPPING THE SPECIALIZATION LATTICE USING MULTIPLE OPTIONS

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

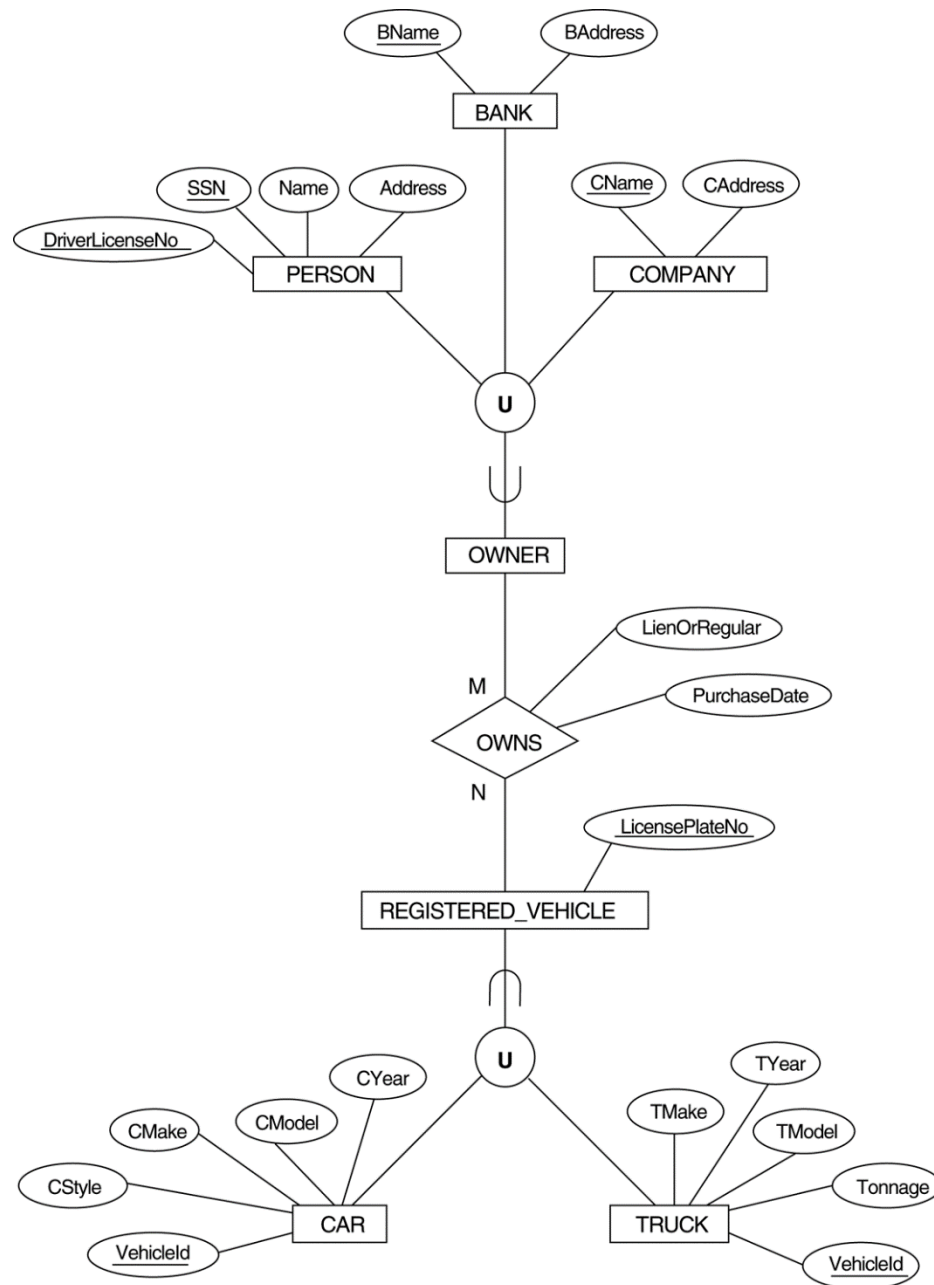
#### STUDENT

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

# STEP 9: MAPPING OF CATEGORIES

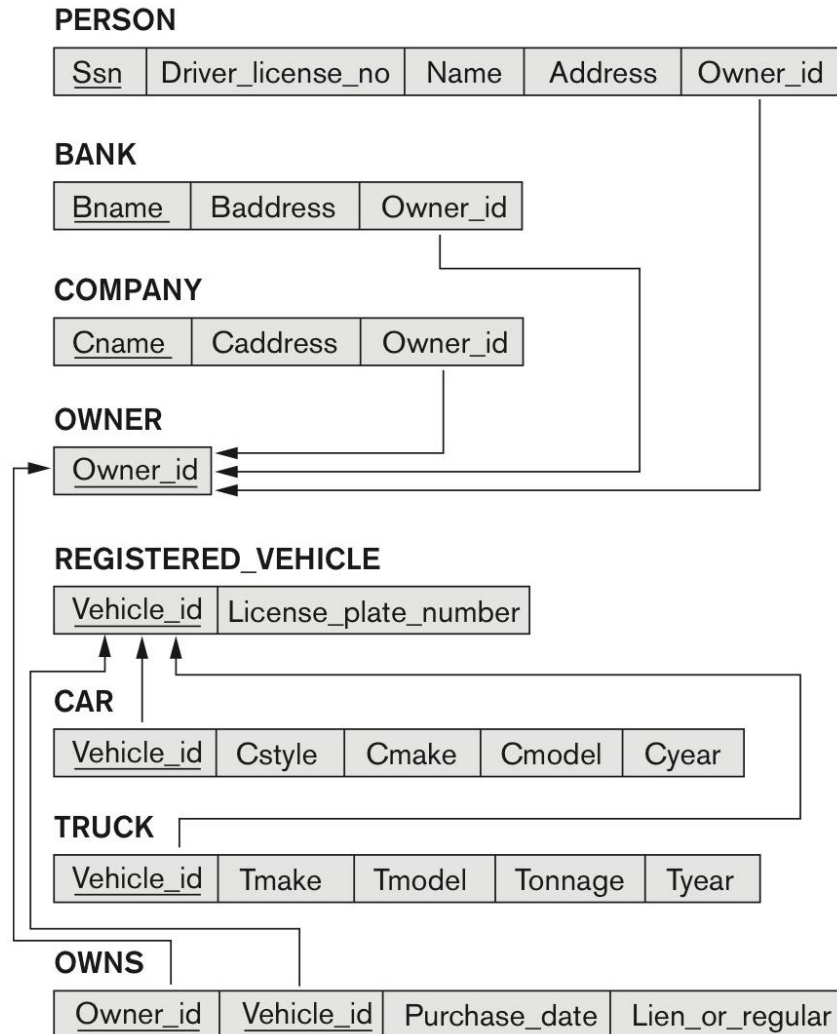
- Create a new relation for each category. Specify a new key attribute, called a **surrogate key**. This surrogate key is added to each relation representing a superclass (as a foreign key)
- In the example, the OWNER relation corresponds to the OWNER category and includes attributes of the category. The primary key of the OWNER relation is the OwnerId surrogate key.

# TWO CATEGORIES



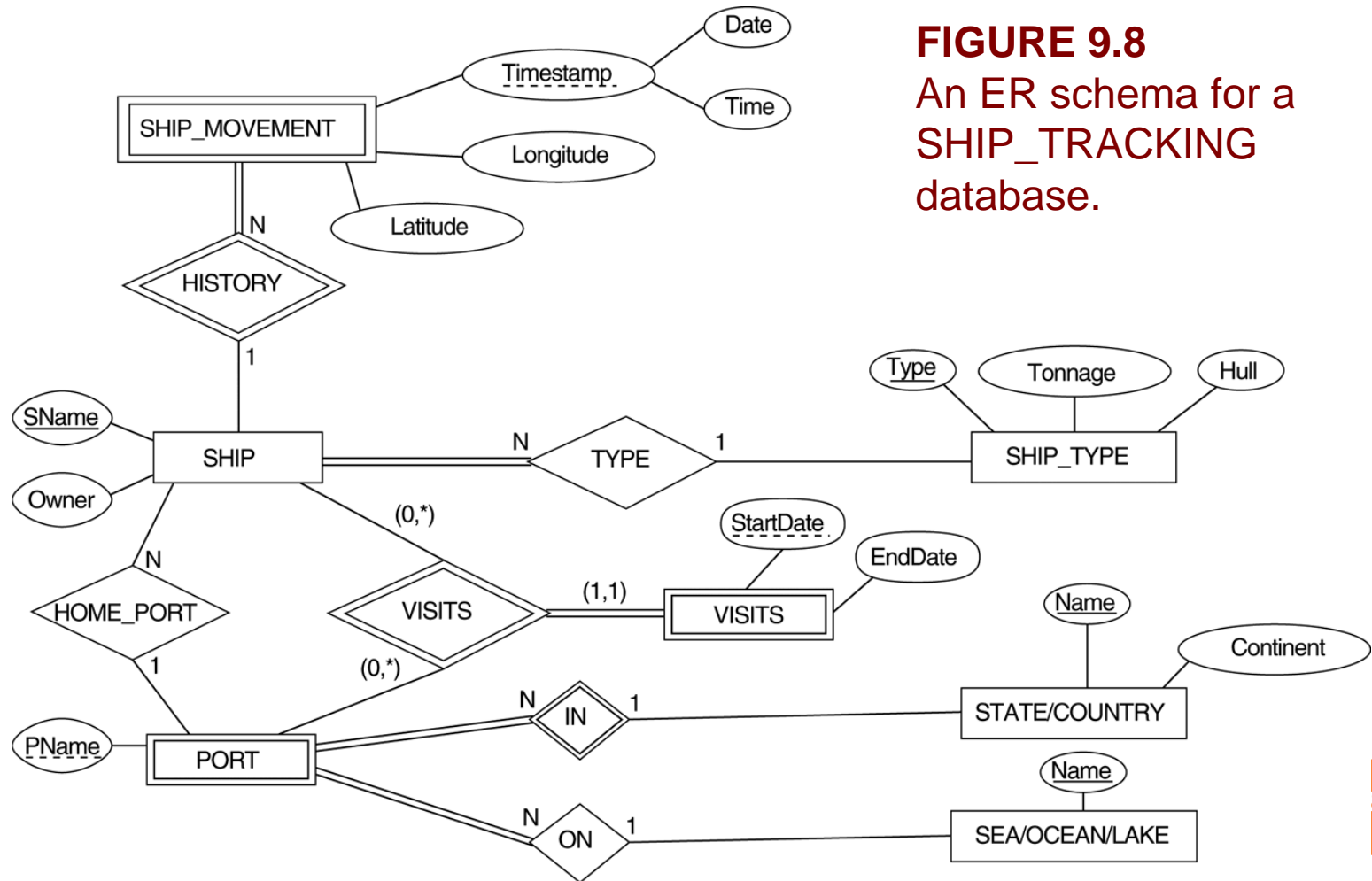
**FIGURE 9.7**

**MAPPING THE EER CATEGORIES (UNION TYPES) IN FIGURE 4.8 TO RELATIONS.**





# MAPPING EXERCISE-1

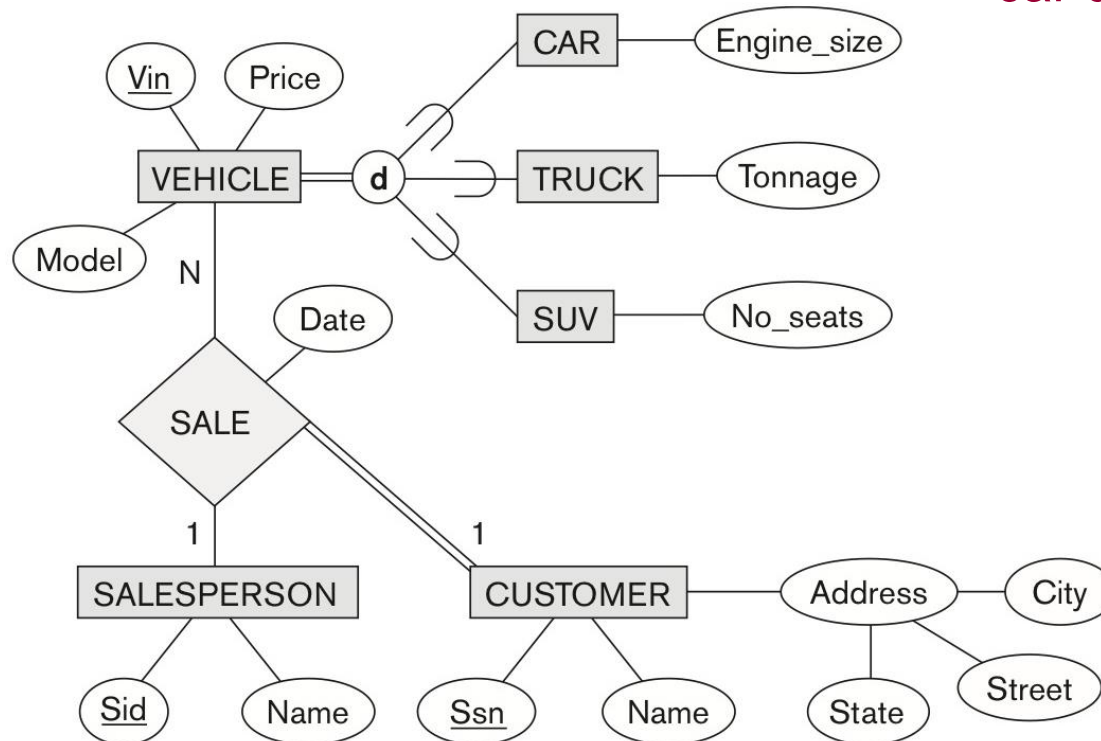


**FIGURE 9.8**  
An ER schema for a  
**SHIP\_TRACKING**  
database.

# MAPPING EXERCISE-2

Exercise 9.9 : Map this schema into a set of relations

**FIGURE 9.9**  
EER diagram for a  
car dealer



# OVERVIEW

1. Structural component
2. Integrity component
3. EER to relational mapping algorithm
4. Operational component
  - Updates
  - Retrievals - Relational algebra
5. Views

# UPDATE OPERATIONS

- INSERT
  - DELETE
  - MODIFY
- 
- Integrity constraints should not be violated by the update operations.
  - Several update operations may have to be grouped together.
  - Updates may *propagate* to cause other updates automatically. This may be necessary to maintain integrity constraints.

# UPDATE OPERATIONS

- In case of integrity violation, several actions can be taken:
  - Cancel the operation that causes the violation (REJECT option)
  - Perform the operation but inform the user of the violation
  - Trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)
  - Execute a user-specified error-correction routine

# RELATIONAL ALGEBRA

- Chapter 8
- Operations over relations
  - Unary: Select, Project
  - Binary: Join, Division
- Set operations
  - Union, Intersection, Set difference, Cartesian product
- Additional operations
  - Outer joins
  - Grouping
  - Aggregate functions

# SELECT

- $\sigma_{\langle \text{selection condition} \rangle} (\langle \text{relation} \rangle)$
- Unary operation
- $\sigma_{\text{DNO} = 4} (\text{EMPLOYEE})$
- Selection condition can be:
  - simple:  $\langle \text{attribute} \rangle \langle \text{operator} \rangle \langle \text{value} \rangle$
  - composite: simple conditions connected by AND, OR and NOT

# SELECT

- Example 1: *Find all information about the director whose number is 100.*
- Example 2: *Select the tuples for all stars who were born in New York or who were born in 1950.*



# PROJECT

- $\pi_{\langle \text{attribute list} \rangle} (\langle \text{relation} \rangle)$
- Unary operation
- Example 3: Generate *a list of all movies by title*
- Example 4: *List each customer's first and last name and address.*

R

<u>A</u>	B	C
1	1	2
2	2	1
3	3	2
4	1	2
5	2	1
6	3	3

1.  $\pi_{A,C} (R)$

2.  $\pi_{B,C} (R)$

3.  $\sigma_{B \geq 2} (R)$

# RELATIONAL ALGEBRA EXPRESSIONS

- We may want to apply several relational algebra operations one after the other
  - Either we can write the operations as a single **relational algebra expression** by nesting the operations, or
  - We can apply one operation at a time and create **intermediate result relations**.
- In the latter case, we must give names to the relations that hold the intermediate results.

# SINGLE EXPRESSION VERSUS SEQUENCE OF RELATIONAL OPERATIONS (EXAMPLE)

- To retrieve the first name, last name, and salary of all employees who work in department number 5, we must apply a select and a project operation
- We can write a *single relational algebra expression* as follows:
  - $\pi_{\text{FNAME, LNAME, SALARY}}(\sigma_{\text{DNO}=5}(\text{EMPLOYEE}))$
- OR We can explicitly show the *sequence of operations*, giving a name to each intermediate relation:
  - $\text{DEP5\_EMPS} \leftarrow \sigma_{\text{DNO}=5}(\text{EMPLOYEE})$
  - $\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME, SALARY}}(\text{DEP5\_EMPS})$

# SEQUENCES OF OPERATIONS

- Operations can be nested to any depth
- The result of one operation used as the argument of another
- Example 5:  
*List the names of stars born in 1990.*

# RENAMING TABLES AND ATTRIBUTES

- The rename operator is  $\rho$  (rho)
  - $\rho_{S(B_1, B_2, \dots, B_n)}(R)$  is a renamed relation  $S$  based on  $R$  with column names  $B_1, B_2, \dots, B_n$ .
  - $\rho_S(R)$  is a renamed relation  $S$  based on  $R$  (which does not specify column names)

# SET OPERATIONS

- Union compatible relations:
  - Same degree
  - Each pair of corresponding attributes have the same domain
- Union:  $S \cup R$
- Intersection:  $S \cap R$
- Set difference (minus):  $S - R$

# SOME PROPERTIES OF UNION, INTERSECT, AND DIFFERENCE

- Both union and intersection are *commutative*  
 $R \cup S = S \cup R$ , and  $R \cap S = S \cap R$
- Both union and intersection can be treated as n-ary operations applicable to any number of relations as both are *associative* operations; that is
  - $R \cup (S \cup T) = (R \cup S) \cup T$
  - $(R \cap S) \cap T = R \cap (S \cap T)$
- The minus operation is not commutative; that is, in general
  - $R - S \neq S - R$



# CARTESIAN PRODUCT

- Set operation
- Does not require union compatible relations
- $S \times R$
- Binary operation
- Commutative

# SET OPERATIONS

- Example 6:

*Produce all combinations of DVD codes and movie numbers.*

- Example 7:

*For each star, show his/her name and a list of all movies he/she acted in.*

# JOIN

- Binary relational operation (commutative)
- $R \bowtie_{\langle \text{join condition} \rangle} S$
- Join condition:  $R.A = S.B$
- Example 8:  
*Join the MOVIE and DVD tables*

# JOIN

- Types of join:
  - Equijoin
  - Theta join
  - Natural join  $R \bowtie S$

R

<u>A</u>	B	C
1	1	2
2	2	1
3	3	2
4	1	2
5	2	1
6	3	3

S

<u>D</u>	E	A
1	a	1
2	b	2
3	e	4
4	a	1
5	c	null
6	d	4
7	f	2

R \* S

# JOIN

- Example 9:

*Make a list of movie titles and directors.*

- Example 10:

*Generate a list showing the name of a star and the title of a movie the star acted in.*

# COMPLETE SET OF RELATIONAL ALGEBRA OPERATIONS

- $\{\sigma, \pi, \cup, -, \bowtie\}$
- How can we represent:
  - Intersection?
  - Join?

# DIVISION

- Given two relations,  $R(X)$  and  $S(Z)$
- $R \div S = T(Y)$
- $Y = X - Z$
- For a tuple  $t$  to appear in  $T$ , the values in  $t$  must appear in  $R$  in combination with *every* tuple in  $S$ .
- Equivalent to:

$$T1 \leftarrow \pi_Y (R)$$

$$T2 \leftarrow \pi_Y ((S \times T1) - R)$$

$$T1 \leftarrow T1 - T2$$



# EXAMPLE OF DIVISION

Figure 8.8 The DIVISION operation. (a) Dividing SSN\_PNOS by SMITH\_PNOS. (b)  $T \leftarrow R \div S$ .

(a)

**SSN\_PNOS**

Essn	Pno
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10
987987987	30
987654321	30
987654321	20
888665555	20

**SMITH\_PNOS**

Pno
1
2

**SSNS**

Sen
123456789
453453453

(b)

**R**

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

**S**

A
a1
a2
a3

**T**

B
b1
b4

# DIVISION

●  $R \div S$

R	A	B
	a1	b1
	a2	b1
	a3	b1
	a4	b1
	a1	b2
	a3	b2
	a2	b3
	a3	b3
	a1	b4
	a2	b4
	a3	b4

S	A
	a1
	a2
	a3

● Example 11

*Find the names of stars who acted in all movies that Henry Fonda acted in.*

# AGGREGATE FUNCTIONS

- Operate on a set of tuples
  - SUM
  - AVERAGE
  - MAXIMUM
  - MINIMUM
  - COUNT (just counts the number of rows, without removing duplicates)
- Use of the functional operator  $\mathcal{F}$
- $\mathcal{F}_{\text{MAXIMUM Salary}}$  (Employee)
- Example 12:  
*Find the total number of directors in the MOVIE database*

# GROUPING FUNCTION




- $\langle \text{grouping attributes} \rangle \mathcal{F} \langle \text{function list} \rangle (\langle \text{relation} \rangle)$

- $\text{DNO } \mathcal{F} \text{COUNT SSN, AVERAGE Salary (Employee)}$

- Example 13:

*Retrieve each director's number and the number of movies he/she directed.*

# OUTER JOIN OPERATIONS

- Left outer join     R   $\langle \text{join condition} \rangle$  S
- Right outer join     R   $\langle \text{join condition} \rangle$  S
- Full outer join     R   $\langle \text{join condition} \rangle$  S

- Example 14:

*Find a list of all employee names and also the name of the department they manage, if they happen to do so.*

# AN EXAM QUESTION

Using the given tables, show the results of the natural join, equijoin and outer join (left, right and full outer join) operations.

STUDENT

<u>ST_NO</u>	STAFF_NO
956328	
942312	2
971111	4
972548	2
953333	
962831	1

STAFF

<u>STAFF_NO</u>	DEPARTMENT
1	2
2	6
3	6
4	4
5	3

# TABLE 8.1 OPERATIONS OF RELATIONAL ALGEBRA

**Table 8.1** Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation $R$ .	$\sigma_{\langle \text{selection condition} \rangle}(R)$
PROJECT	Produces a new relation with only some of the attributes of $R$ , and removes duplicate tuples.	$\pi_{\langle \text{attribute list} \rangle}(R)$
THETA JOIN	Produces all combinations of tuples from $R_1$ and $R_2$ that satisfy the join condition.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$
EQUIJOIN	Produces all the combinations of tuples from $R_1$ and $R_2$ that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$ , OR $R_1 \bowtie_{(\langle \text{join attributes 1} \rangle), (\langle \text{join attributes 2} \rangle)} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of $R_2$ are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 \star_{\langle \text{join condition} \rangle} R_2$ , OR $R_1 \star_{(\langle \text{join attributes 1} \rangle), (\langle \text{join attributes 2} \rangle)} R_2$ OR $R_1 \star R_2$

# TABLE 8.1 OPERATIONS OF RELATIONAL ALGEBRA (CONTINUED)

**Table 8.1** Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
UNION	Produces a relation that includes all the tuples in $R_1$ or $R_2$ or both $R_1$ and $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both $R_1$ and $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in $R_1$ that are not in $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of $R_1$ and $R_2$ and includes as tuples all possible combinations of tuples from $R_1$ and $R_2$ .	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in $R_1$ in combination with every tuple from $R_2(Y)$ , where $Z = X \cup Y$ .	$R_1(Z) \div R_2(Y)$



# STRATEGY FOR BUILDING QUERIES

- Determine which tables contain the data
- Specify join conditions if there is more than one table in found in step 1
- Specify selection conditions (if necessary)
- Specify expressions to appear in the result
- Group the tuples (if necessary)

# VIEWS

- Virtual tables: data does not exist on the disk; instead, the query is stored
- Purpose
  - Security
  - Customized access to data
  - Specifications of complex operations on base tables
- Updatable views
  - Defined over a single table
  - Containing primary (or a candidate) key
  - Not involving aggregate functions or grouping