

# Relational Database Design by ER- and EER-to-Relational Mapping

## 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).

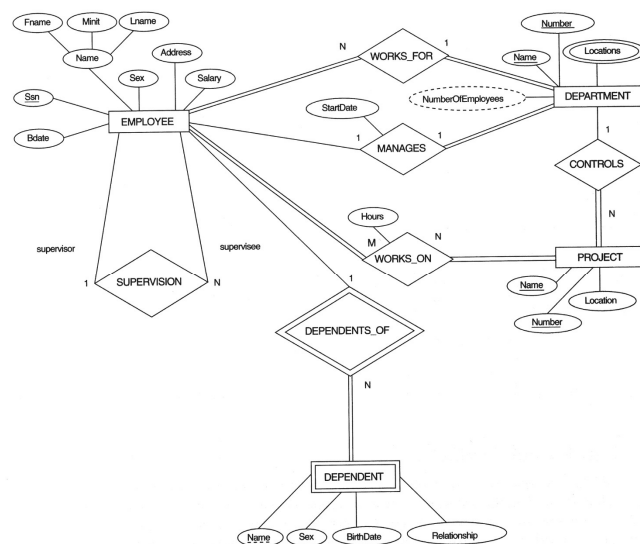
## ER-to-Relational Mapping Algorithm

### ● 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.

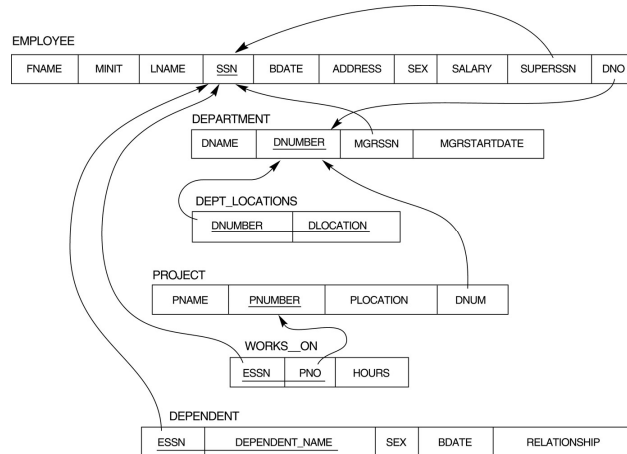
**Example:** We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in the relational schema corresponding to the regular entities in the ER diagram. SSN, DNUMBER, and PNUMBER are the primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT as shown.

**FIGURE 7.1**  
The ER  
conceptual  
schema  
diagram for  
the  
COMPANY  
database.



**FIGURE 7.2**

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



## 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.

**Example:** Create the relation **DEPENDENT** in this step to correspond to the weak entity type **DEPENDENT**. Include the primary key **SSN** of the **EMPLOYEE** relation as a foreign key attribute of **DEPENDENT** (renamed to **ESSN**).

The primary key of the **DEPENDENT** relation is the combination {**ESSN**, **DEPENDENT\_NAME**} because **DEPENDENT\_NAME** is the partial key of **DEPENDENT**.

## ER-to-Relational Mapping Algorithm (cont)

### ● Step 3: Mapping of Binary 1:1 Relation 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:

- (1) Foreign Key approach: Choose one of the relations-S, say-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 in the role of S.  
**Example**: 1:1 relation MANAGES is mapped by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total.
- (2) Merged relation option: An alternate mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when *both participations are total*.
- (3) Cross-reference or relationship relation option: The third alternative 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.

## ER-to-Relational Mapping Algorithm (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 relation type as attributes of S.

**Example**: 1:N relationship types WORKS\_FOR, CONTROLS, and SUPERVISION in the figure. For WORKS\_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO.

## ER-to-Relational Mapping Algorithm (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.

**Example:** The M:N relationship type WORKS\_ON from the ER diagram is mapped by creating a relation WORKS\_ON in the relational database schema. The primary keys of the PROJECT and EMPLOYEE relations are included as foreign keys in WORKS\_ON and renamed PNO and ESSN, respectively.

Attribute HOURS in WORKS\_ON represents the HOURS attribute of the relation type. The primary key of the WORKS\_ON relation is the combination of the foreign key attributes {ESSN, PNO}.

## ER-to-Relational Mapping Algorithm (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.

**Example:** The relation DEPT\_LOCATIONS is created. The attribute DLOCATION represents the multivalued attribute LOCATIONS of DEPARTMENT, while DNUMBER-as foreign key-represents the primary key of the DEPARTMENT relation. The primary key of R is the combination of {DNUMBER, DLOCATION}.

## ER-to-Relational Mapping Algorithm (cont)

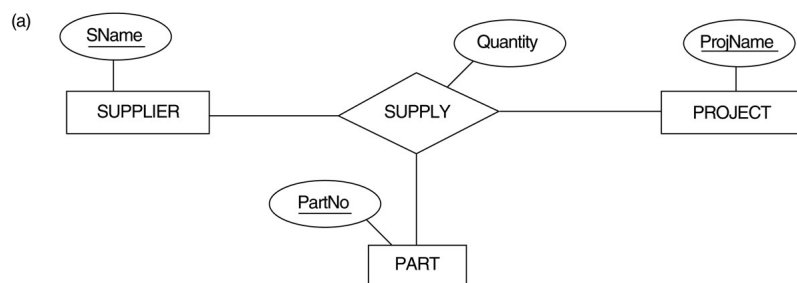
### ● Step 7: Mapping of N-ary Relationship Types.

- For each n-ary relationship type R, where  $n > 2$ , create a new relationship 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.

**Example:** The relationship type SUPPLY in the ER below. This can be mapped to the relation SUPPLY shown in the relational schema, whose primary key is the combination of the three foreign keys {SNAME, PARTNO, PROJNAME}

**FIGURE 4.11**

Ternary relationship types. (a) The SUPPLY relationship.



**FIGURE 7.3**

Mapping the  $n$ -ary relationship type SUPPLY from Figure 4.11a.

SUPPLIER

<u>SNAME</u>	...
--------------	-----

PROJECT

<u>PROJNAME</u>	...
-----------------	-----

PART

<u>PARTNO</u>	...
---------------	-----

SUPPLY

<u>SNAME</u>	<u>PROJNAME</u>	<u>PARTNO</u>	QUANTITY
--------------	-----------------	---------------	----------

## Summary of Mapping constructs and constraints

*Table 7.1 Correspondence between ER and Relational Models*

**ER Model**

Entity type  
 1:1 or 1:N relationship type  
 M:N relationship type  
 $n$ -ary relationship type  
 Simple attribute  
 Composite attribute  
 Multivalued attribute  
 Value set  
 Key attribute

**Relational Model**

“Entity” relation  
 Foreign key (or “relationship” relation)  
 “Relationship” relation and two foreign keys  
 “Relationship” relation and  $n$  foreign keys  
 Attribute  
 Set of simple component attributes  
 Relation and foreign key  
 Domain  
 Primary (or secondary) key

## Mapping EER Model Constructs to Relations

- **Step8: Options for Mapping Specialization or 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:

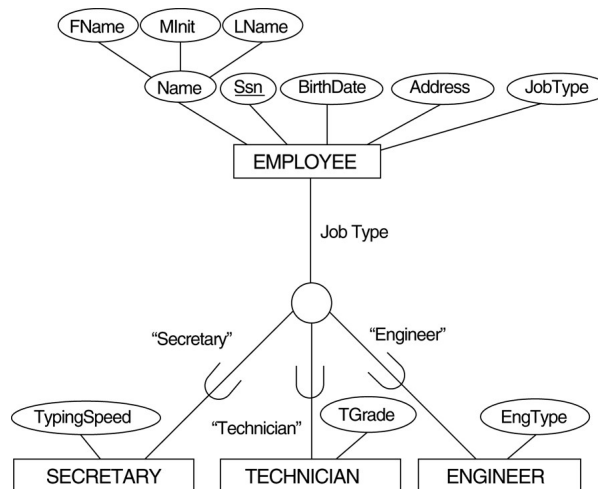
**Option 8A: Multiple relations-Superclass and subclasses.**

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** (total or partial, disjoint or over-lapping).

**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 a specialization whose subclasses are **total** (every entity in the superclass must belong to (at least) one of the subclasses).

**FIGURE 4.4**  
EER diagram notation for an attribute-defined specialization on JobType.





**FIGURE 7.4**

Options for mapping specialization or generalization.

(a) Mapping the EER schema in Figure 4.4 using option 8A.

(a) EMPLOYEE

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

SECRETARY

<u>SSN</u>	TypingSpeed
------------	-------------

TECHNICIAN

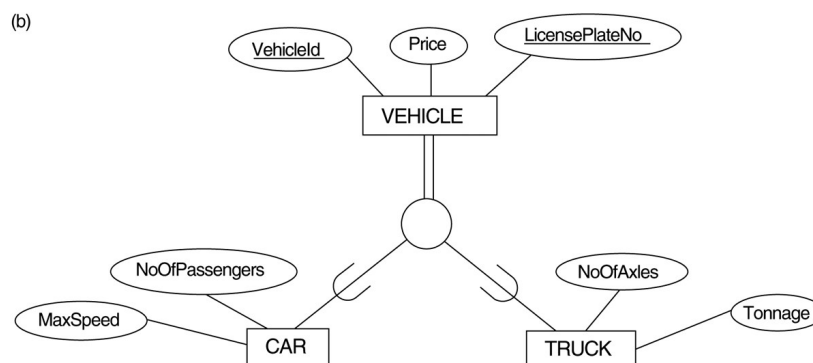
<u>SSN</u>	TGrade
------------	--------

ENGINEER

<u>SSN</u>	EngType
------------	---------

**FIGURE 4.3**

Generalization. (b) Generalizing CAR and TRUCK into the superclass VEHICLE.



**FIGURE 7.4**

Options for mapping specialization or generalization.  
 (b) Mapping the EER schema in Figure 4.3b using option 8B.

(b) CAR

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

TRUCK

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

## Mapping EER Model Constructs to Relations (cont)

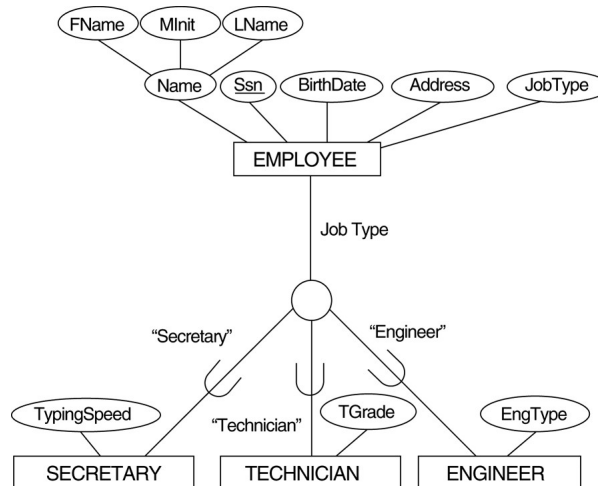
### Option 8C: Single relation with one type attribute.

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  $S_i$ .

**FIGURE 4.4**  
EER diagram  
notation for an  
attribute-  
defined  
specialization  
on JobType.



**FIGURE 7.4**

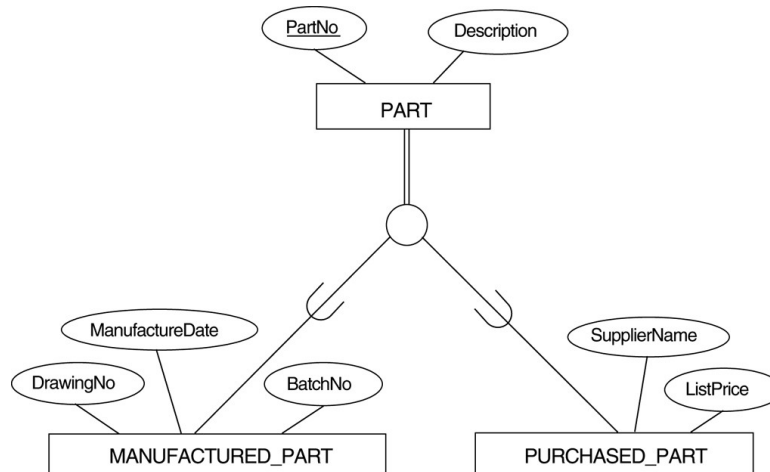
Options for mapping specialization or generalization.  
(c) Mapping the EER schema in Figure 4.4 using option 8C.

(c) EMPLOYEE

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

**FIGURE 4.5**

EER diagram notation for an overlapping (nondisjoint) specialization.

**FIGURE 7.4**

Options for mapping specialization or generalization.  
(d) Mapping Figure 4.5 using option 8D with Boolean type fields Mflag and Pflag.

(d) PART

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

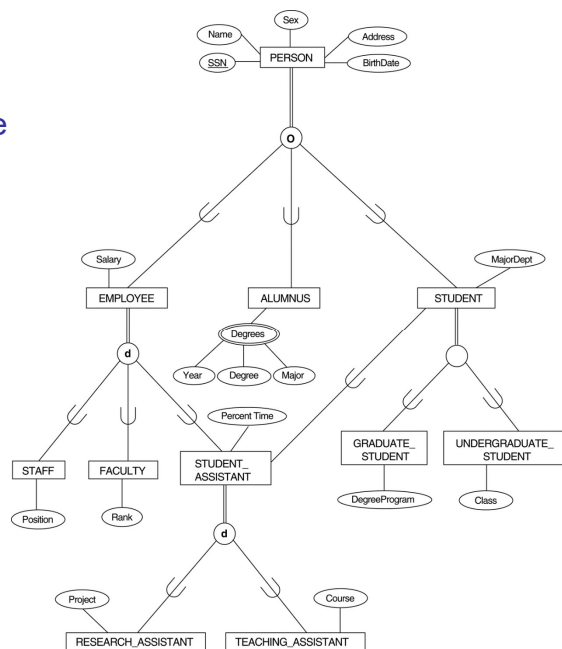
## Mapping EER Model Constructs to Relations (cont)

- **Mapping of Shared Subclasses (Multiple Inheritance)**

A shared subclass, such as STUDENT\_ASSISTANT, 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. Below both 8C and 8D are used for the shared class STUDENT\_ASSISTANT.

**FIGURE 4.7**  
A specialization lattice with multiple inheritance for a UNIVERSITY database.



**FIGURE 7.5**

Mapping the EER specialization lattice in Figure 4.6 using multiple options.

**PERSON**

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

**EMPLOYEE**

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

**ALUMNUS**

<u>SSN</u>
------------

**ALUMNUS\_DEGREES**

<u>SSN</u>	Year	Degree	
------------	------	--------	--

**STUDENT**

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

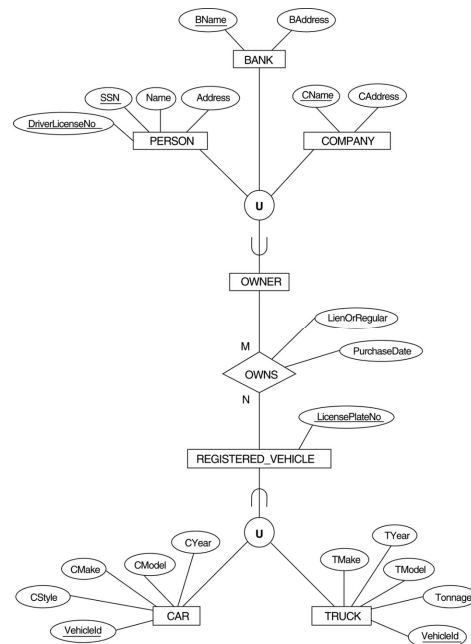
## Mapping EER Model Constructs to Relations (cont)

- **Step 9: Mapping of Union Types (Categories).**

- For mapping a category whose defining superclass 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.
- In the example below we can create a relation OWNER to correspond to the OWNER category and include any attributes of the category in this relation. The primary key of the OWNER relation is the surrogate key, which we called OwnerId.

**FIGURE 4.8**

Two categories (union types): OWNER and REGISTERED\_VEHICLE.

**FIGURE 7.6**

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

**PERSON**

<u>SSN</u>	DriverLicenseNo	Name	Address	
------------	-----------------	------	---------	--

**BANK**

<u>BName</u>	BAddress	OwnerId
--------------	----------	---------

**COMPANY**

<u>CName</u>	CAddress	OwnerId
--------------	----------	---------

**OWNER**

<u>OwnerId</u>
----------------

**REGISTERED\_VEHICLE**

<u>VehicleId</u>	LicensePlateNumber
------------------	--------------------

**CAR**

<u>VehicleId</u>	CStyle	CMake	CModel	
------------------	--------	-------	--------	--

**TRUCK**

<u>VehicleId</u>	TMake	TModel	Tonnage	TYear
------------------	-------	--------	---------	-------

**OWNS**

<u>OwnerId</u>	<u>VehicleId</u>	PurchaseDate	LienOrRegular
----------------	------------------	--------------	---------------

## Acknowledgement

Reference for this lecture is

- Ramez Elmasri and Shamkant B. Navathe,  
*Fundamentals of Database Systems*, Pearson  
Education.

***The authors and the publishers are  
gratefully acknowledged.***

---