CSE2DBF – CSE4DBF
Enhanced Entity-Relationship Modeling (EER) &
Transformation to Relational Tables

Reading:

Elmasri and Navathe, "Fundamentals of Database Systems, Chapters 1 & 2", Pearson, 2016.

Ebook: *https://ebookcentral-proquest-com.ez.library.latrobe.edu.au/lib/latrobe/detail.action?docID=5573709*

EER Model

- EER stands for **Enhanced ER** or **Extended ER**

- EER Model Concepts
  - Includes all modeling concepts of basic ER
  - Additional concepts:
    - Subclass/superclass *entities*
    - Specialization/generalization *relationships*
    - Union types (categories) *relationships*

- The additional EER concepts are used to model applications more completely and more accurately

- EER includes some OO concepts, such as inheritance.

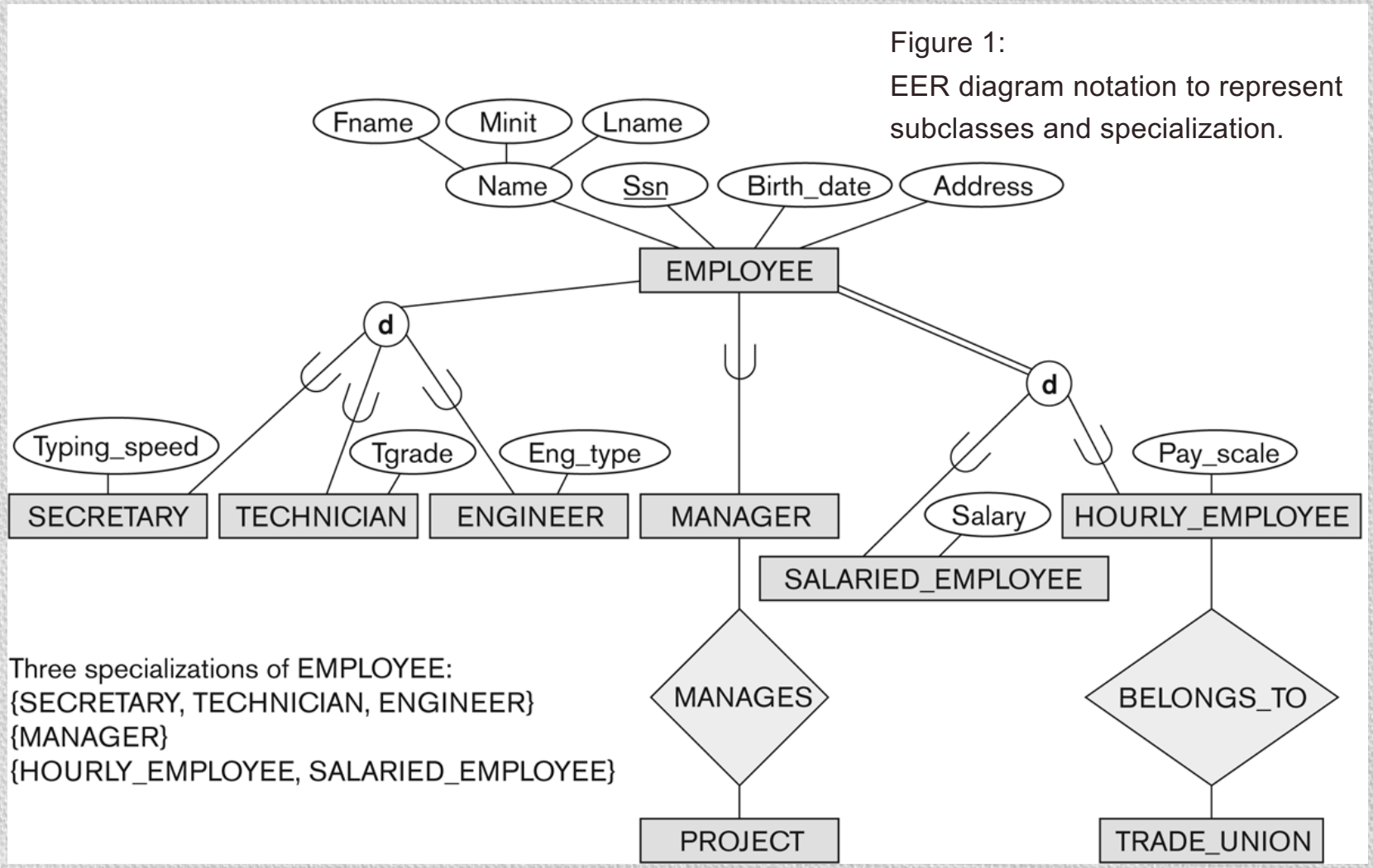
EER Model

- Transformation of EER Model Constructs to Relations
  - **STEP 8**: Transformation of Specialization/Generalization.
  - **STEP 9**: Transformation of Union Types (Categories).

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.*

Subclasses and Superclasses



(Source: Elmasri and Navathe, Fundamentals of Database Systems, Figure 1, page 248)

Subclasses and Superclasses

- EMPLOYEE is a superclass entity and all subgrouping members of EMPLOYEE are the subclasses entities.

- An instance of an EMPLOYEE superclass can belong to more than one subclasses
  - 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 all instances of a superclass entity be a member of a subclass.

Inheritance

- 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

- 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

Specialization/Generalization

- ***Specialization*** is the process of defining a set of subclasses of an entity type, the latter is called the superclass of the specialization.

  Example:
  A specialization of EMPLOYEE based on *method of pay* is {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE}.

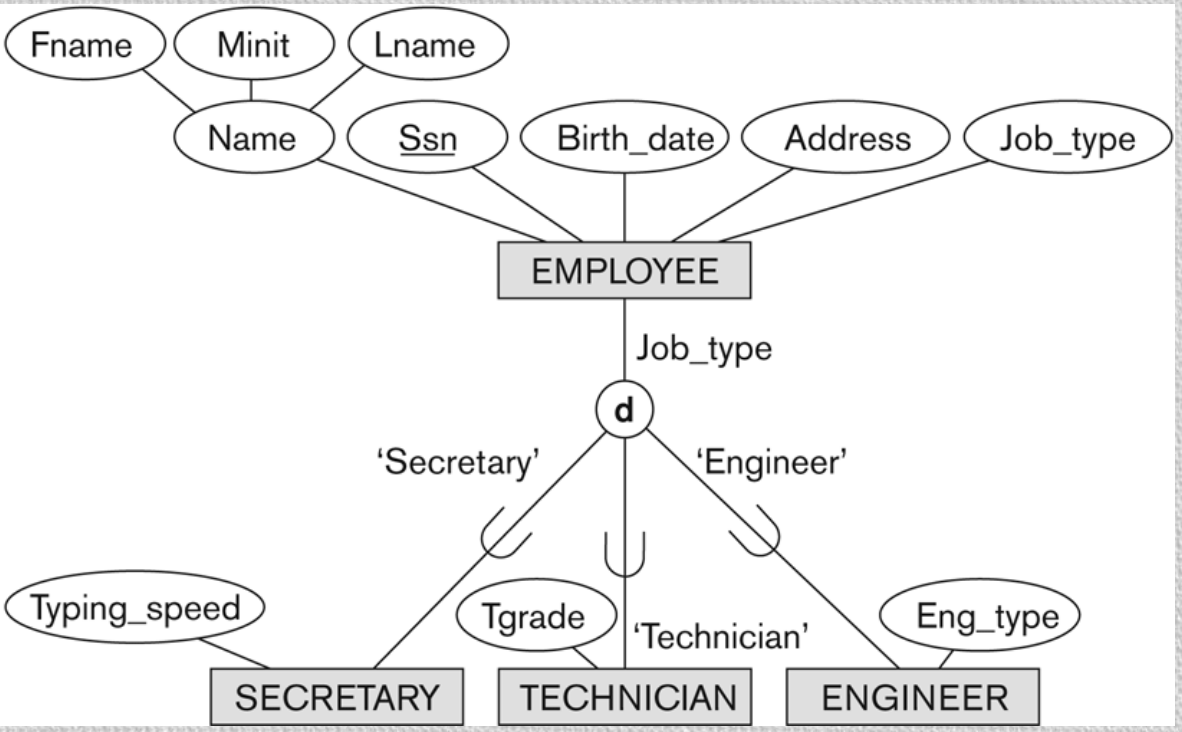
- ***Generalization*** is the process of identifying common features of existing entity types and generalize them into a single superclass.

- Specialization/ generalization can be diagrammatically represented in EER diagrams. **Note**: We do not differentiate specialization and generalization notation for this subject.

Specialization/Generalization with A Defining Attribute

- If all subclasses in a specialization have their membership condition on the same attribute of the superclass, the specialization is called an **attributed-defined specialization** and the attribute is called **the defining attribute** of the specialization.

Figure 2:
EER diagram notation for an attribute-defined specialization on Job_type.



(Source: Elmasri and Navathe, Fundamentals of Database Systems, Figure 4, page 253)

Constraints on Specialization/Generalization

- Two basic constraints can apply to a specialization/generalization:
  - Disjointed Constraint
  - Completeness Constraint

Disjointed Constraints on Specialization/Generalization

- Specifies that the subclasses of the specialization must be *disjoint*:
  - an entity can be a member of at most one of the subclasses of the specialization.
- In EER diagram, it is specified by ***d*** symbol in the circle that connect the superclass with the subclasses.

- If not disjoint, the specialization is *overlapping*:
  - that is the same entity may be a member of more than one subclass of the specialization.
- In EER diagram, it is specified by ***o*** symbol in the circle that connect the superclass with the subclasses.

Example of Overlapping Specialization/Generalization

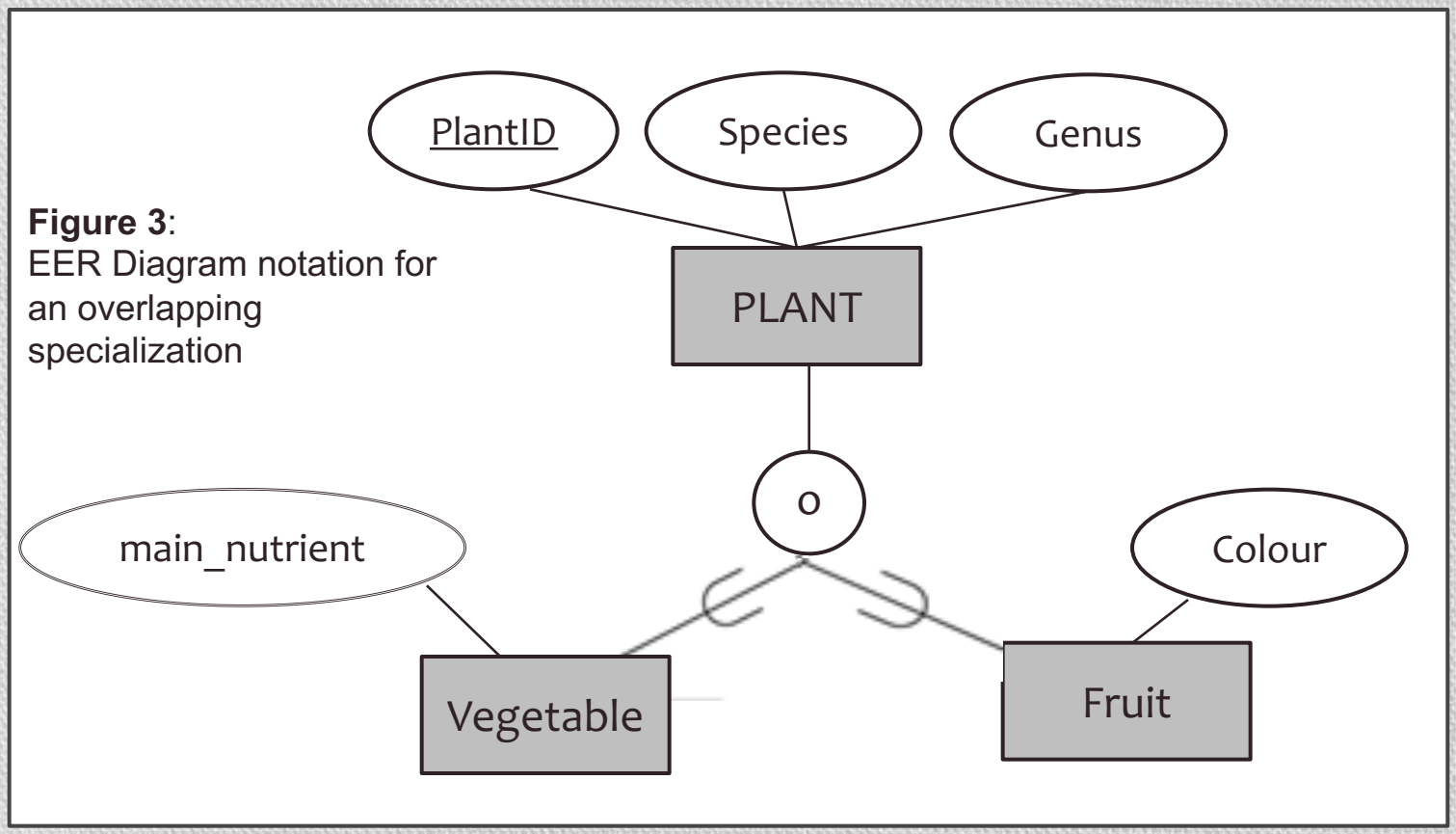


Figure 3:
EER Diagram notation for an overlapping specialization

Completeness Constraints on Specialization/Generalization

- Specifies the existence of subclass for instances of superclass entity.

- ***Total*** specifies that every instance of superclass entity must be a member of at least one subclass in the specialization/generalization.
- Shown in EER diagrams by a ***double line*** **.**

- ***Partial*** allows an instance of superclass entity not to belong to any of the subclasses .
- Shown in EER diagrams by a ***single line***.

Summary – Four Constraints on Specialization/Generalization

- Hence, we have four types of specialization/generalization:
  - Disjoint, total
  - Disjoint, partial
  - Overlapping, total
  - Overlapping, partial

Transformation of Specialization/Generalization

STEP 8: Options for the Transformation of Specialization/Generalization

- 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

Transformation of Specialization/Generalization

Option 8A: Multiple relations-Superclass and subclasses

- Create a relation for the superclass which contains the primary key and all other attributes of the superclass.
- Create a relation for each of the subclasses, which includes all attributes of that subclass and where the primary key is the primary key of the superclass.
- This option works for **any** specialization (**total or partial, disjoint or overlapping**).

Transformation of Specialization/Generalization

Option 8A: Multiple relations-Superclass and subclasses

Example:

EMPLOYEE and SECRETARY, TECHNICIAN, ENGINEER (see previous slide – Figure 1)

The relational tables:

EMPLOYEE (Ssn, Fname, Minit, Lname, Birth_date, Address)

SECRETARY (*Ssn*, Typing_speed)

TECHNICIAN (*Ssn*, Tgrade)

ENGINEER (*Ssn*, Eng_type)

Transformation of Specialization/Generalization

Option 8B: Multiple relations-Subclass relations only

- Create a relation for each subclass which contains the attributes of the superclass and the attributes of that subclass. The primary key is the primary key of the superclass.
- 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).
- If the specialization is overlapping, an entity may be duplicated in several relations.

Transformation of Specialization/Generalization

Option 8B: Multiple relations-Subclass relations only

Example:

EMPLOYEE and SALARIED_EMPLOYEE, HOURLY_EMPLOYEE (see previous slide – Figure 1)

The relational tables:

SALARIED_EMPLOYEE (Ssn, Fname, Minit, Lname, Birth_date, Address, Salary)

HOURLY_EMPLOYEE (Ssn, Fname, Minit, Lname, Birth_date, Address, Pay_scale)

Transformation of Specialization/Generalization

Option 8C: Single relation with one type attribute

- Create a single relation that has as its attributes all the attributes of the superclass, plus all the attributes of each of the subclasses, plus an additional attribute called **type**.
- The **type attribute** is the discriminating attribute that identifies the subclass to which each tuple belongs, if any. The primary key of the relation is the primary key of the superclass.
- 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.

Transformation of Specialization/Generalization

Option 8C: Single relation with one type attribute

Example:

EMPLOYEE and SECRETARY, TECHNICIAN, ENGINEER (see previous slide – Figure 2)

The relational tables:

EMPLOYEE (Ssn, Fname, Minit, Lname, Birth_date, Address, Job_type, Typing_speed, Tgrade, Eng_type)

Note: The value of **Job_type** can either be Employee, Secretary, Technician, or Engineer.

Transformation of Specialization/Generalization

Option 8D: Single relation with multiple type attributes

- Create a single relation that has as its attributes all the attributes of the superclass, plus all the attributes of each of the subclasses, plus **multiple type** attributes. Each of the type attributes is a **Boolean** indicating whether a tuple belongs to a particular subclass.
- This option works for a specialization whose subclasses are **overlapping** (but will also work for a disjoint specialisation)

Transformation of Specialization/Generalization

Option 8D: Single relation with multiple type attributes

Example:
PLANT and VEGETABLE, FRUIT (see previous slide – Figure 3)

The relational tables:
PLANT (PlantID, Species, Genus, VFlag, Main_nutrient, Fflag, colour)

Note: Boolean type fields VFlag and FFlag

Union Types

- All of the *superclass/subclass relationships* we have seen thus far have a **single superclass**.
- In some cases, we need to model a *single superclass/subclass relationship* with **more than one superclass**.
- Superclasses can represent different entity types, and therefore they may not have many common attributes even though they represent **a particular meaning**.
- The subclass of such superclasses is called a category or **UNION TYPE**.

Example of Union Types

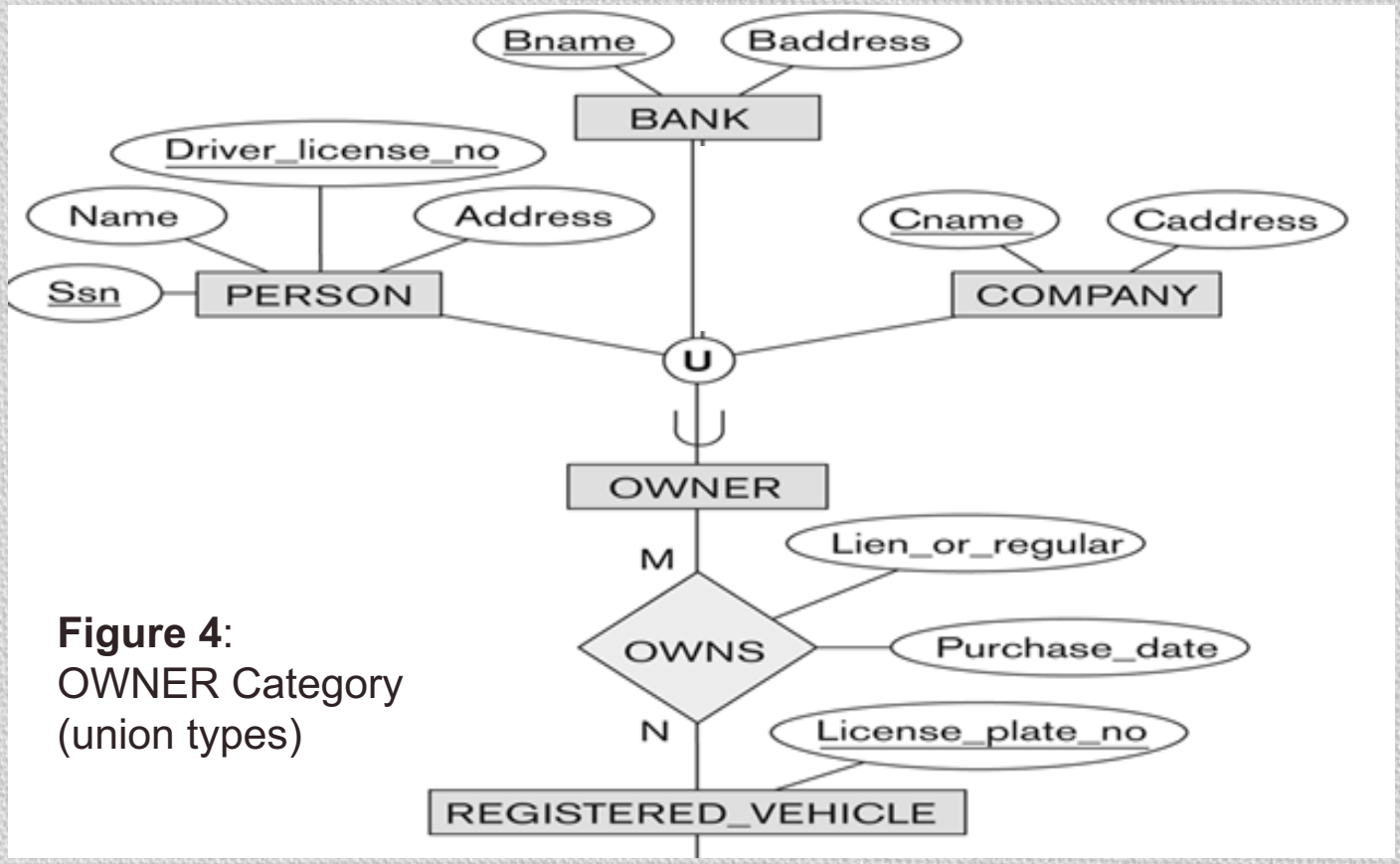


Figure 4:
OWNER Category
(union types)

(Source: Elmasri and Navathe, Fundamentals of Database Systems, Figure 8, page 260)

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 (see previous slide - Figure 4)

  - 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

Transformation of Union Types

STEP 9: Transformation of Union Types (Categories)

- For mapping a category whose defining superclasses 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.
- The category becomes an entity with the surrogate key and the surrogate key becomes a foreign key in the defining superclasses.
- If the defining superclasses have the same key, that key becomes the primary key of the category.

Transformation of Union Types

STEP 9: Transformation of Union Types (Categories)

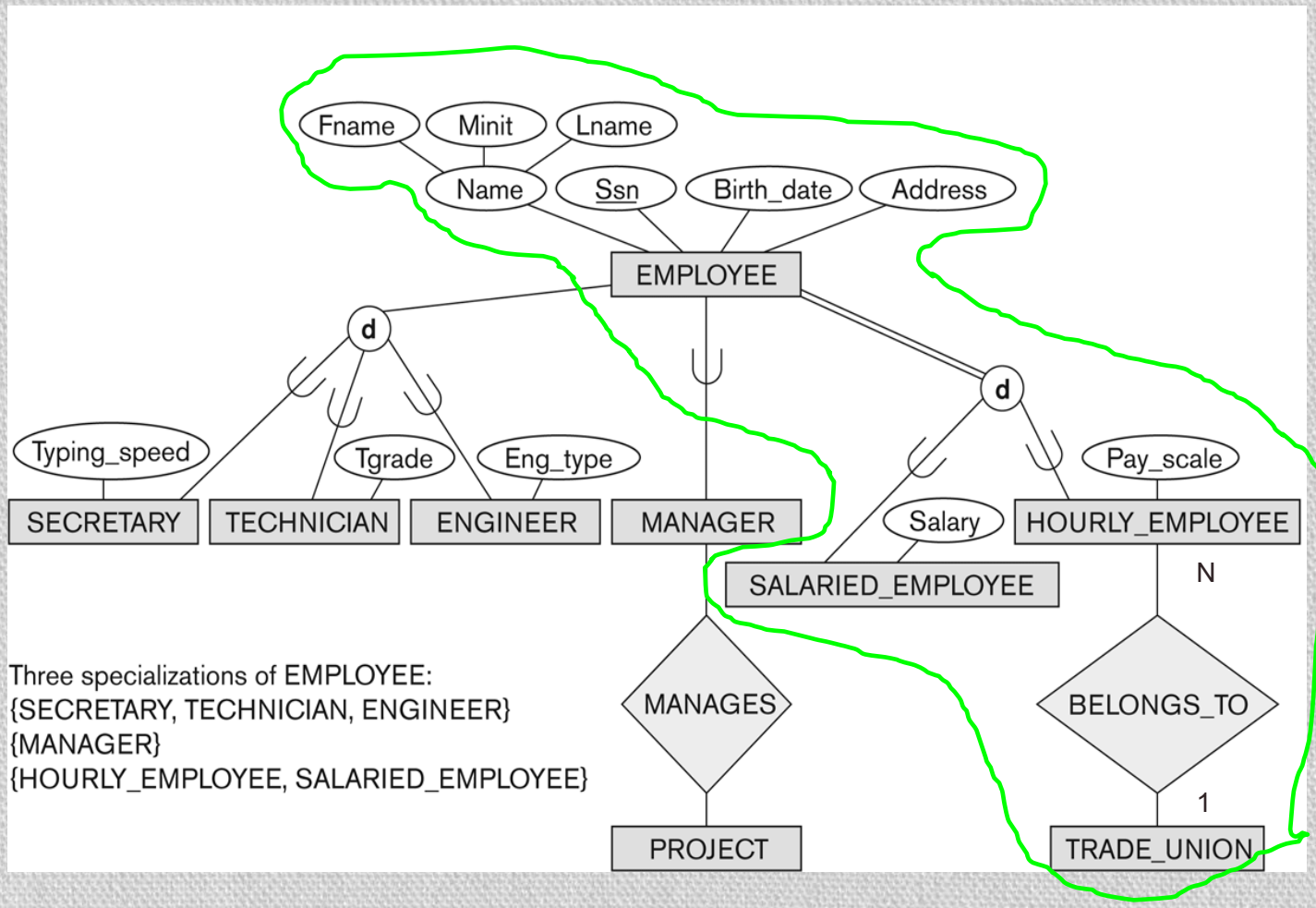
Example:
PERSON, BANK, COMPANY and OWNER (see previous slide – Fig.4)

The relational tables:
OWNER (Owner_id)
PERSON (Ssn, Driver_license_no, Name, Address, *Owner_id*)
BANK (Bname, Baddress, *Owner_id*)
COMPANY (Cname, Caddress, *Owner_id*)

Note: Owner_id is the surrogate key because the Owner superclasses has different primary keys.

FINAL STEP: Repeat STEPS 2-7 for SUBCLASSES TABLES

Let's consider the following encircled entities (EMPLOYEE, SALARIED_EMPLOYEE, HOURLY_EMPLOYEE, and TRADE_UNION)



(**Updated** **Source:** Elmasri and Navathe, Database Systems, Figure 6, page 255)

Transformation of Union Types

Assume we us option 8B for specialization, we have created the following tables:

SALARIED_EMPLOYEE and HOURLY_EMPLOYEE

The relational tables:
SALARIED_EMPLOYEE (Ssn, Fname, Minit, Lname, Birth_date, Address, Salary)
HOURLY_EMPLOYEE (Ssn, Fname, Minit, Lname, Birth_date, Address, Pay_scale)

Because of the N TO 1 relationship between HOURLY_EMPLOYEE and TRADE_UNION, we now have to repeat STEP 4 (see previous lecture) and apply it to the subclass table (ie. HOURLY_EMPLOYEE).

Following the transformation rule for many to one relationships, the HOURLY_EMPLOYEE has to include the primary key of TRADE_UNION as a foreign key:

HOURLY_EMPLOYEE (Ssn, Fname, Minit, Lname, Birth_date, Address, Pay_scale, *TradeUnionID*)

(assume *TradeUnionID* is the PK of TRADE_UNION).

Final Tables

Summary of Steps (Combines ER/EER)

- **Step 1: Transform all (strong) entities to relational tables (including superclass entities but NOT subclasses)**
- **Step 2: Transform weak entities**
- **Step 3: Transform 1 to 1 relationships**
- **Step 4: Transform 1 to many relationships**
- **Step 5: Transform many to many relationships**
- **Step 6: Transform multi-valued attributes**
- **Step 7: Transform n-ary relationships**

- **Step 8: Transform specialization/generalization relationships (8a, 8b, 8c or 8d)**
- **Step 9: Transform union type relationships**

- **Final Step: Repeat Steps 2 to 7 for Subclasses tables created in Step 8 and 9**

EER Diagram and Transformation Example

The university keeps information of Lecturers who work for them. The property of Lecturer includes the staff number, name and title. Information of students is also maintained. Students can be grouped into undergraduate (UG), coursework postgraduate (CPG), and research postgraduate (RPG). For UG, the degree major is recorded. For CPG the previous degrees are required. For all students, their student ID and personal details are required. A student can also be a research assistant (RA). For research assistant, their level and time percentage of research work are recorded.

RPG can work in a specific research area. The information of research area is stored, which includes the research area id and description.

The university also maintains information about research groups that exist in campus. The research group ID, name and head/managers is part of the research group properties. Each research group has members of researchers and each researcher can be part of multiple research group.

To be categorised as researcher, someone has to be either a Lecturer or a RA. However, not all Lecturers are researchers. The same is applicable to RAs.

Develop an **EER model** for the above problem definition. Any assumptions should also be stated. Following the EER model, transform the diagram into a set of final relations using the 9 transformation steps.

EER Diagram and Transformation Example

Solution to be discussed during lecture.

Next Lecture

More on EER

Reading:
Elmasri and Navathe, "Fundamentals of Database Systems, Chapters 1 & 2", Pearson, 2016.
Ebook: *https://ebookcentral-proquest-com.ez.library.latrobe.edu.au/lib/latrobe/detail.action?docID=5573709*