



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

# Enhanced Entity-Relationship Models (EER)

LECTURE 3

**Dr. Philipp Leitner**



philipp.leitner@chalmers.se



@xLeitix

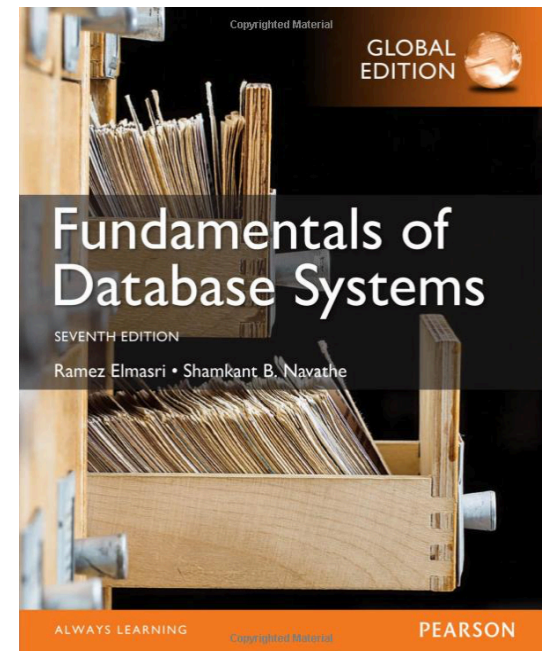
# LECTURE 3

Covers ...

Small part of Chapter 3

Chapter 4

*Please read this up until next lecture!*



# What we will be covering

**Repetition of ER**

**EER extensions (mostly generalization)**

# Reminder: the very basics of ER Diagrams

## **Entities**

Things that exist in your database

## **Attributes**


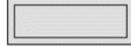







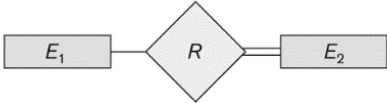

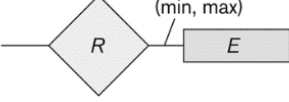
The data that makes up those things

## **Relationships**

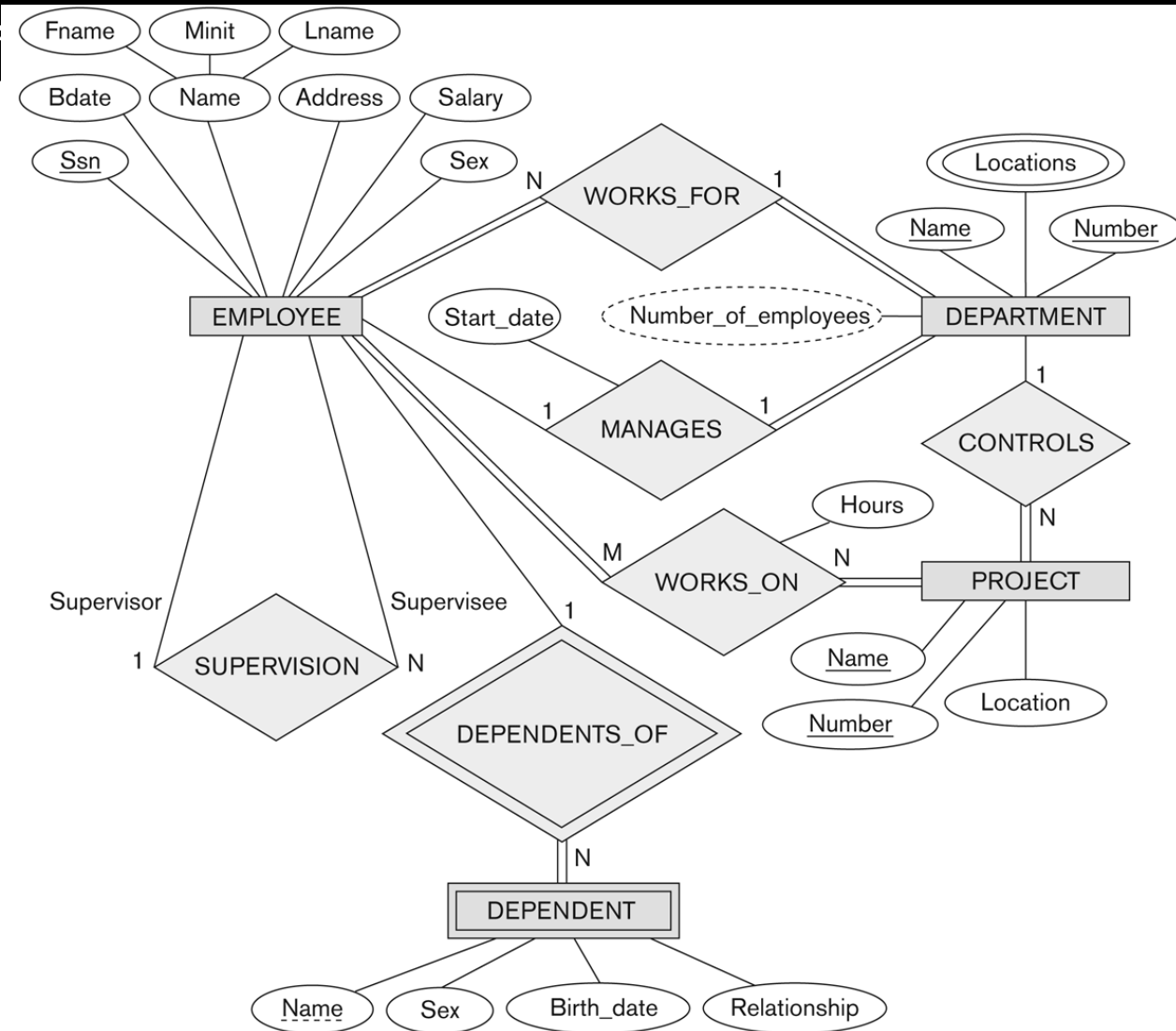
How those things relate to each other

**Figure 3.14**  
Summary of the notation for ER diagrams.

# ER notation cheat sheet

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of $E_2$ in $R$
	Cardinality Ratio 1: N for $E_1:E_2$ in $R$
	Structural Constraint (min, max) on Participation of $E$ in $R$

# Complete example from last week



**Figure 3.2**

An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

# Short Repetition from the End of Last Week

**Weak entities, partial keys**

**Relationship attributes**

**Derived attributes**

# Weak Entities

A **weak entity** does not have a key of its own (it cannot be identified in the database)

Instead it has an **owner** (a relationship with another entity that is not weak)

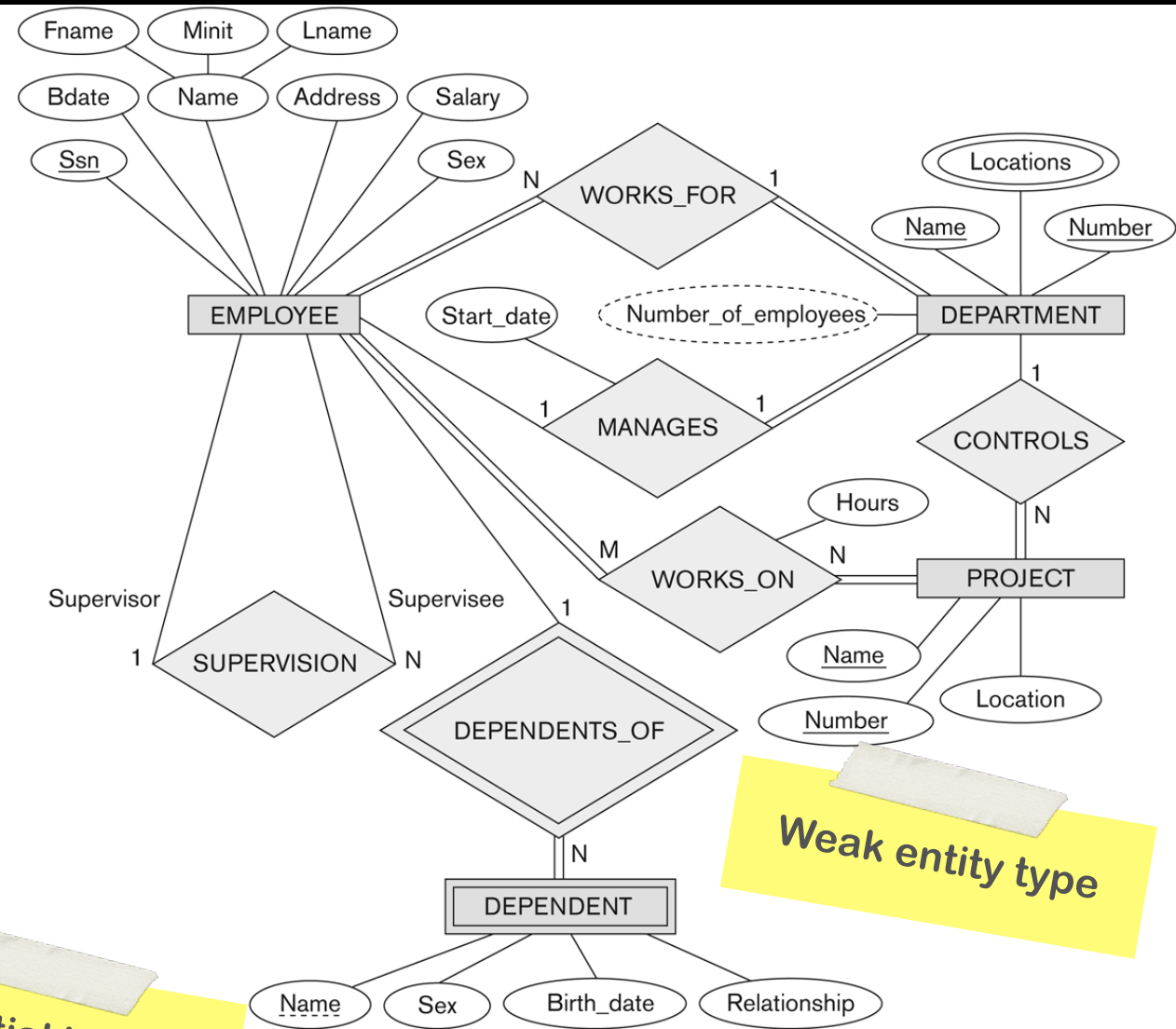
**Weak entities** are identified through a partial key and the owner

Notation:

**Double-line entity symbol and association symbol**

For partial key: **dashed underline**



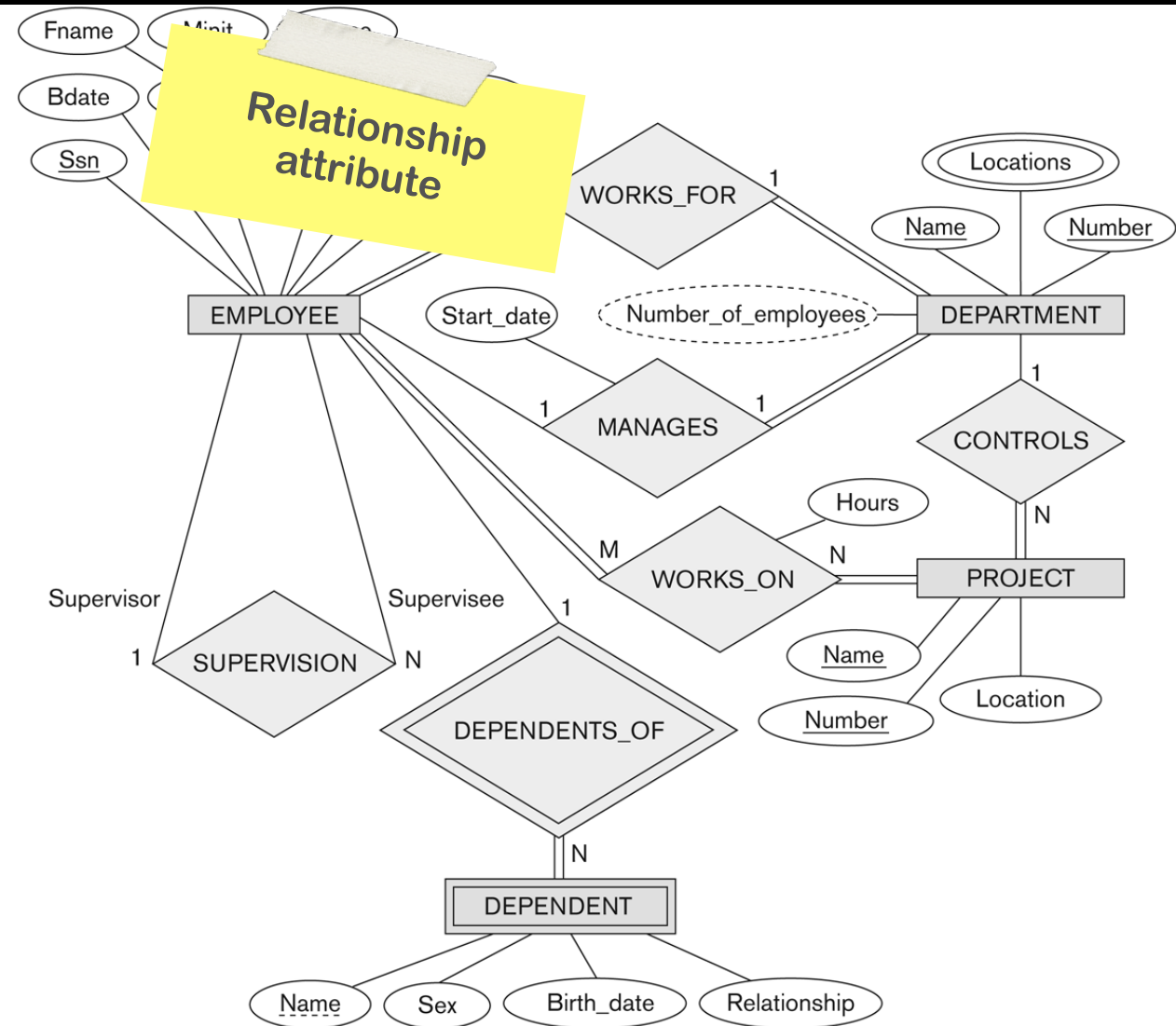


# Attributes of Relationship Types

**A relationship type can have attributes:**

Example: Start\_date of MANAGES

**Value for each relationship instance** describes the number of hours per week that an EMPLOYEE works on a PROJECT. Each value depends on a particular (employee, project) **combination**



**Figure 3.2**

An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

# Derived Attribute

Derived attributes are special in that they keep redundant information. Their value can be **derived** (calculated from other information)

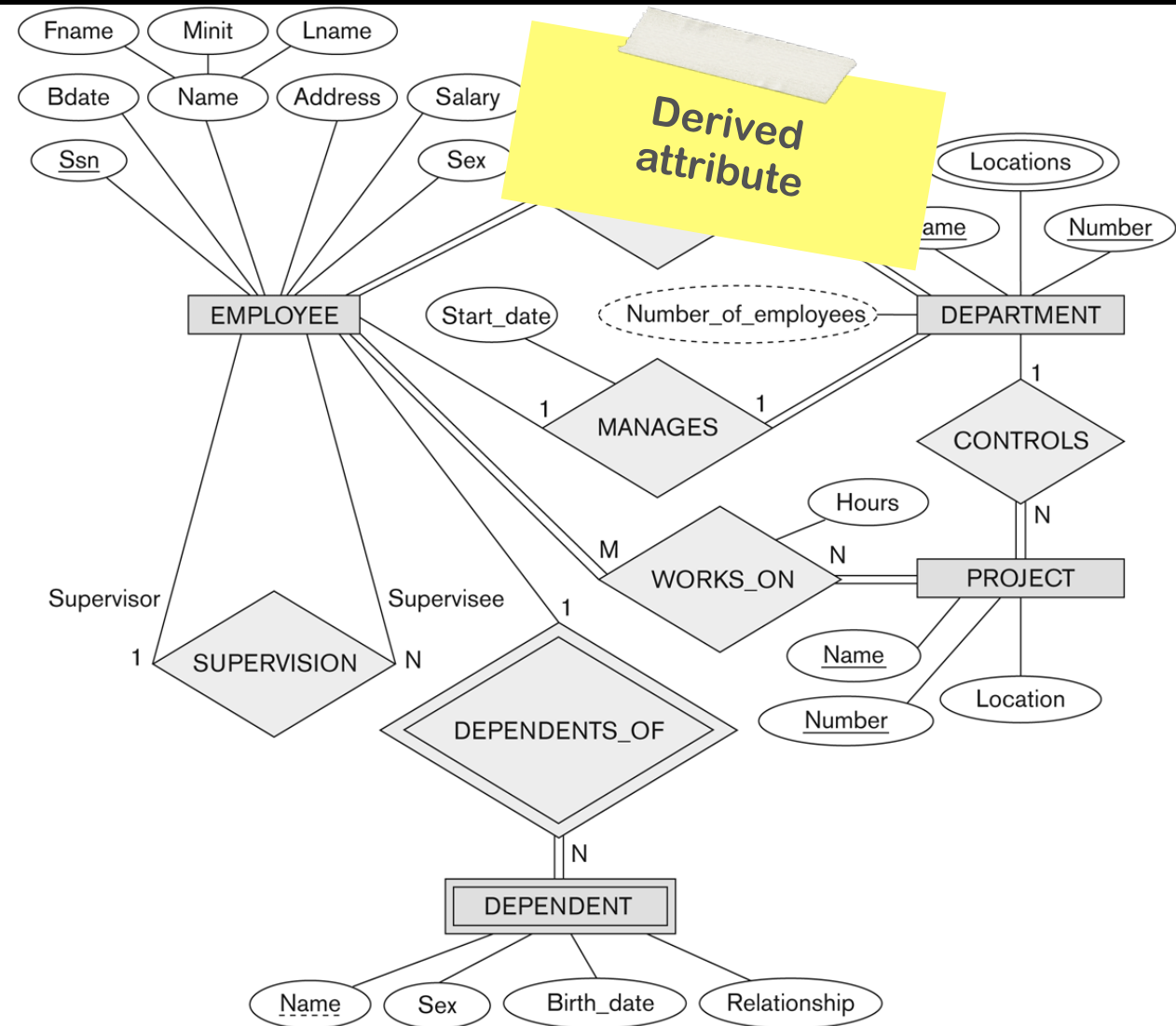
Example:

Nr\_of\_employees

Is just the number of EMPLOYEE entities in the entity set

Notation:

Attribute with **dashed line**



**Figure 3.2**

An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

# Short Quiz

# Another Example - a University Database (1)

The university is organized into colleges with an unique name, a main office, a phone, and a particular faculty member who is the dean of the college. Each college administers a few academic departments, each of which has an unique name, an unique code number, a main office, and a phone. A particular faculty member chairs each department. We also need to keep track of when this person started their chair position.

Departments offer a number of courses, each of which has a unique name, unique code number, a course level, credit hours, and a course description. We also need to keep track of course instructors, which are faculty members. Each instructor has a unique Id, name, office, phone, and rank. Each of these instructors works for exactly one primary department.

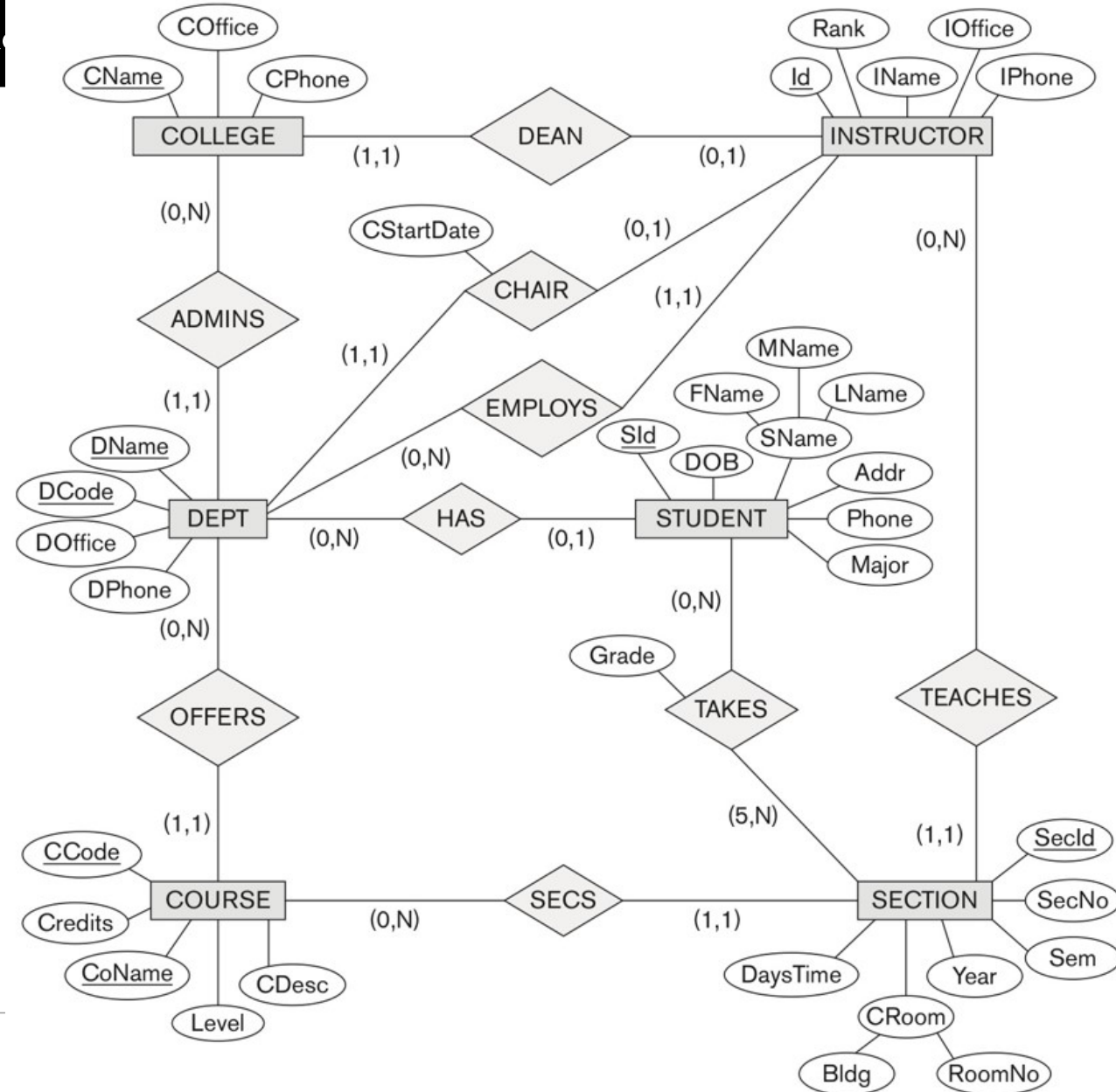
## Another Example - a University Database (2)

The database will keep student data and store each student's name (which is composed of first, middle, and last name), student Id (unique for each student), address, phone, major code, and date of birth. A student is assigned to one primary academic department. It is required to keep track of the student's grades in each section (see below) the student has completed.

Courses are offered in sections, each of which is related to a single course and is taught by a single instructor. Each section has a unique identifier, a number, is taught in a semester and year, and in a classroom. Classrooms are identified through a combination of the building and room numbers. Finally, sections happen during specific times and days (e.g., TUE 13:15-14:45). The database tracks students in each section, and the grade is recorded as soon as it is available. A section must have at least 5 students.



# Complete UNIVERSITY example



# Subclasses and Superclasses

Entity type may have additional meaningful **subgroupings**:

EMPLOYEE may be further grouped into roles:

SECRETARY, ENGINEER, TECHNICIAN, ...

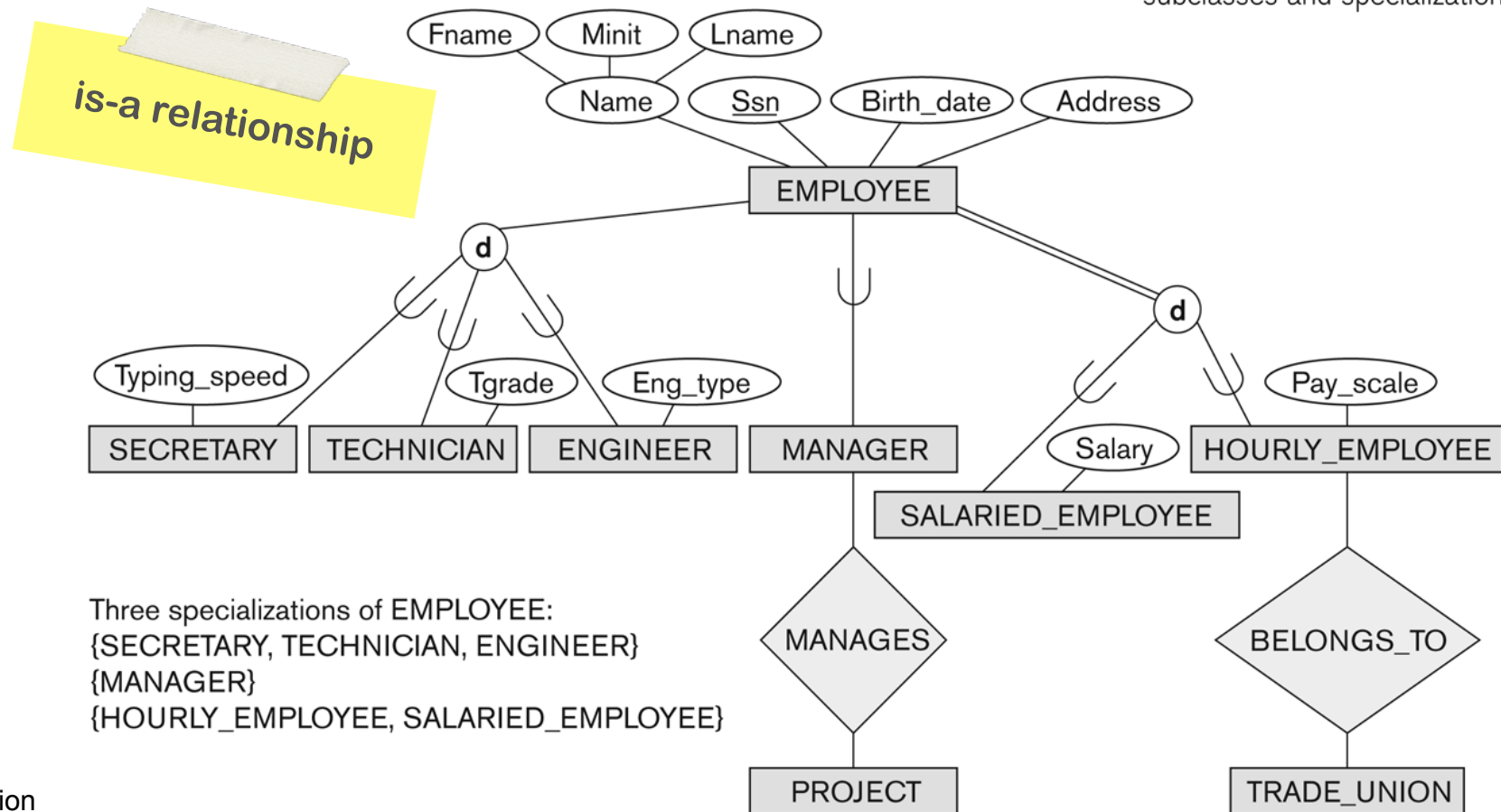
Or based on method of pay:

SALARIED\_EMPLOYEE, HOURLY\_EMPLOYEE

**EER diagrams extend ER diagrams to represent these additional subgroupings, called subclasses or subtypes**

**Figure 4.1**

EER diagram notation to represent subclasses and specialization.



This is exactly like the polymorphism  
you know from Java (with a non-  
abstract superclass)

# IS-A Relationships

SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE, ....

**An entity that is member of a subclass represents the same real-world entity as some member of the superclass**

Subclass member is the same entity in a distinct specific role

Entity **cannot** exist in the database merely by being a member of a subclass; it must also be a member of the superclass

A member of the superclass can be **optionally** included as a member of any number of its subclasses

## Examples:

A salaried employee who is also an engineer belongs to two subclasses:

ENGINEER

SALARIED\_EMPLOYEE

A salaried employee who is also an engineering manager belongs to three subclasses:

MANAGER

ENGINEER

SALARIED\_EMPLOYEE

# Attribute 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



*Again not at all different from Java*

# Specialization vs. Generalization

## **Specialization:**

Start from generic superclass

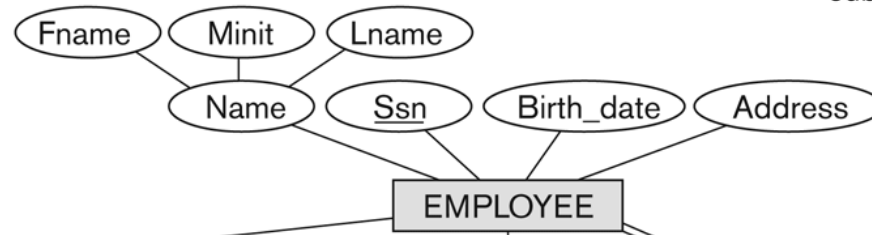
Define subclasses as special cases of superclass

## **Generalization:**

Starts with multiple entities that “have something in common”

Extract commonalities from subclasses into superclass

Copyright (c) 2011 Pearson Education



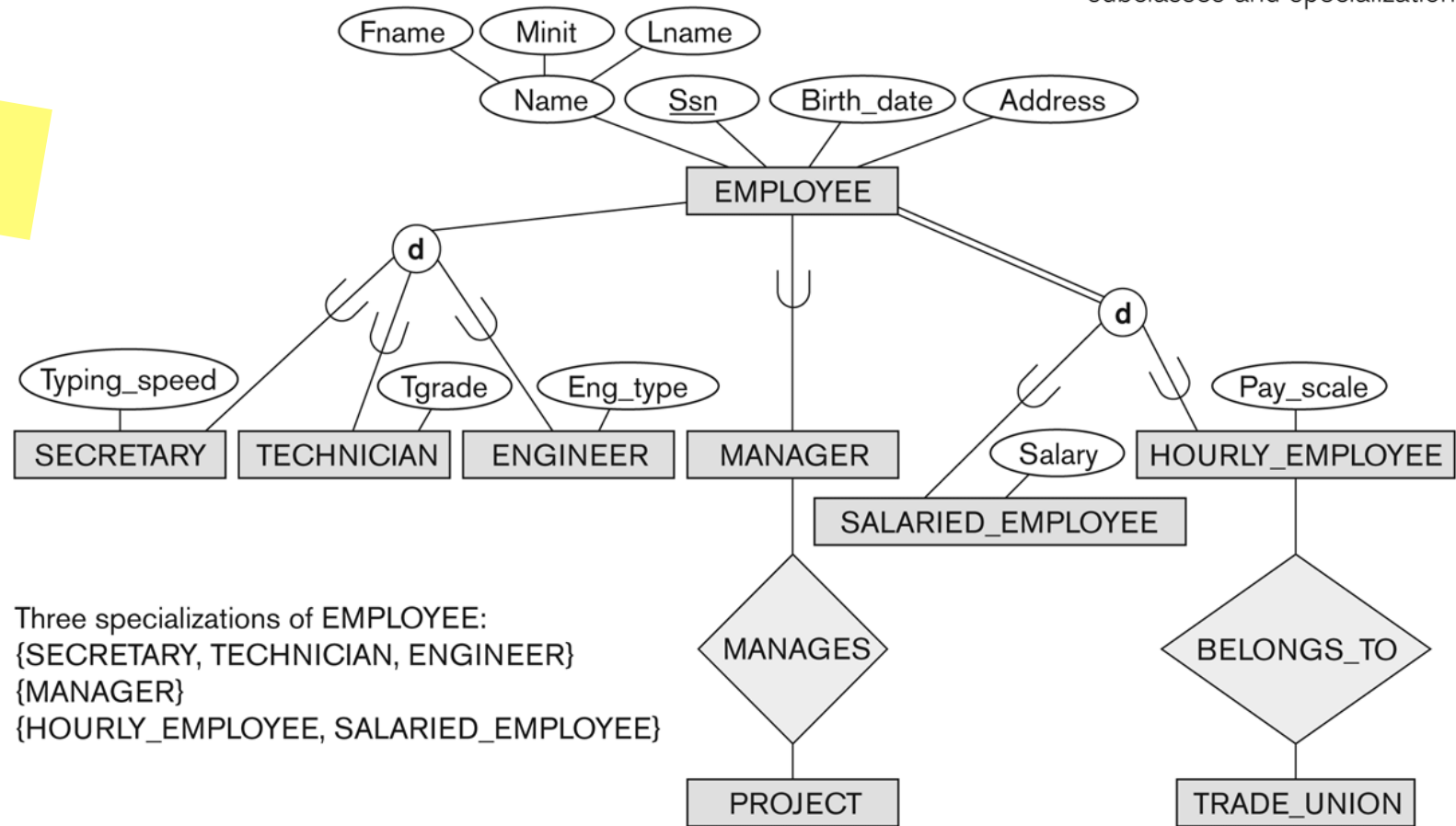
**Figure 4.1**  
EER diagram notation to represent  
subclasses and specialization.



**Figure 4.1**

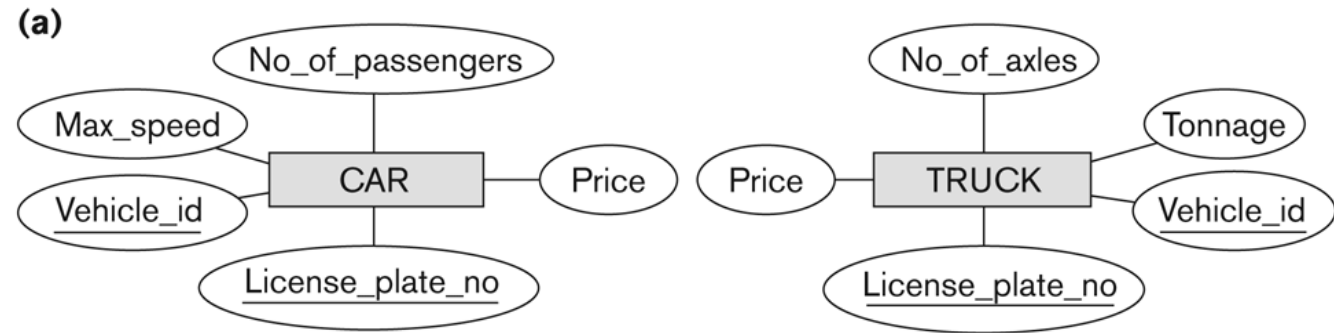
EER diagram notation to represent subclasses and specialization.

**Specialization**

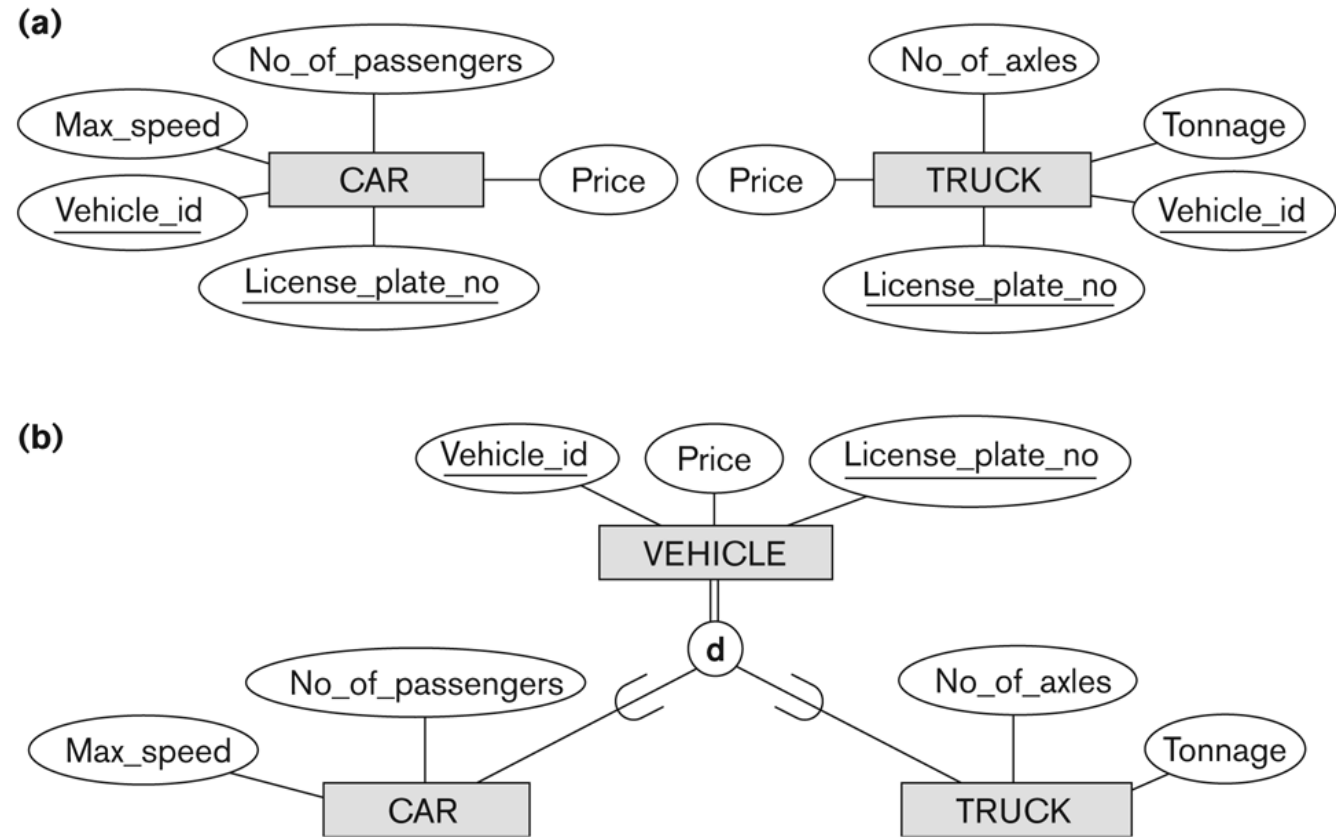


Three specializations of EMPLOYEE:  
 {SECRETARY, TECHNICIAN, ENGINEER}  
 {MANAGER}  
 {HOURLY\_EMPLOYEE, SALARIED\_EMPLOYEE}

Copyright (c) 2011 Pearson Education



Generalization



**Figure 4.3**

Generalization. (a) Two entity types, CAR and TRUCK. (b) Generalizing CAR and TRUCK into the superclass VEHICLE.

# Notation for Specialization and Generalization

**Diagrammatic notations sometimes used to distinguish between generalization and specialization:**

Arrow pointing to the generalized superclass represents a generalization

Arrows pointing to the specialized subclasses represent a specialization

We don't do this in this course

# Identifying Subclasses

Predicate-defined subclasses

based on some predicate (condition)

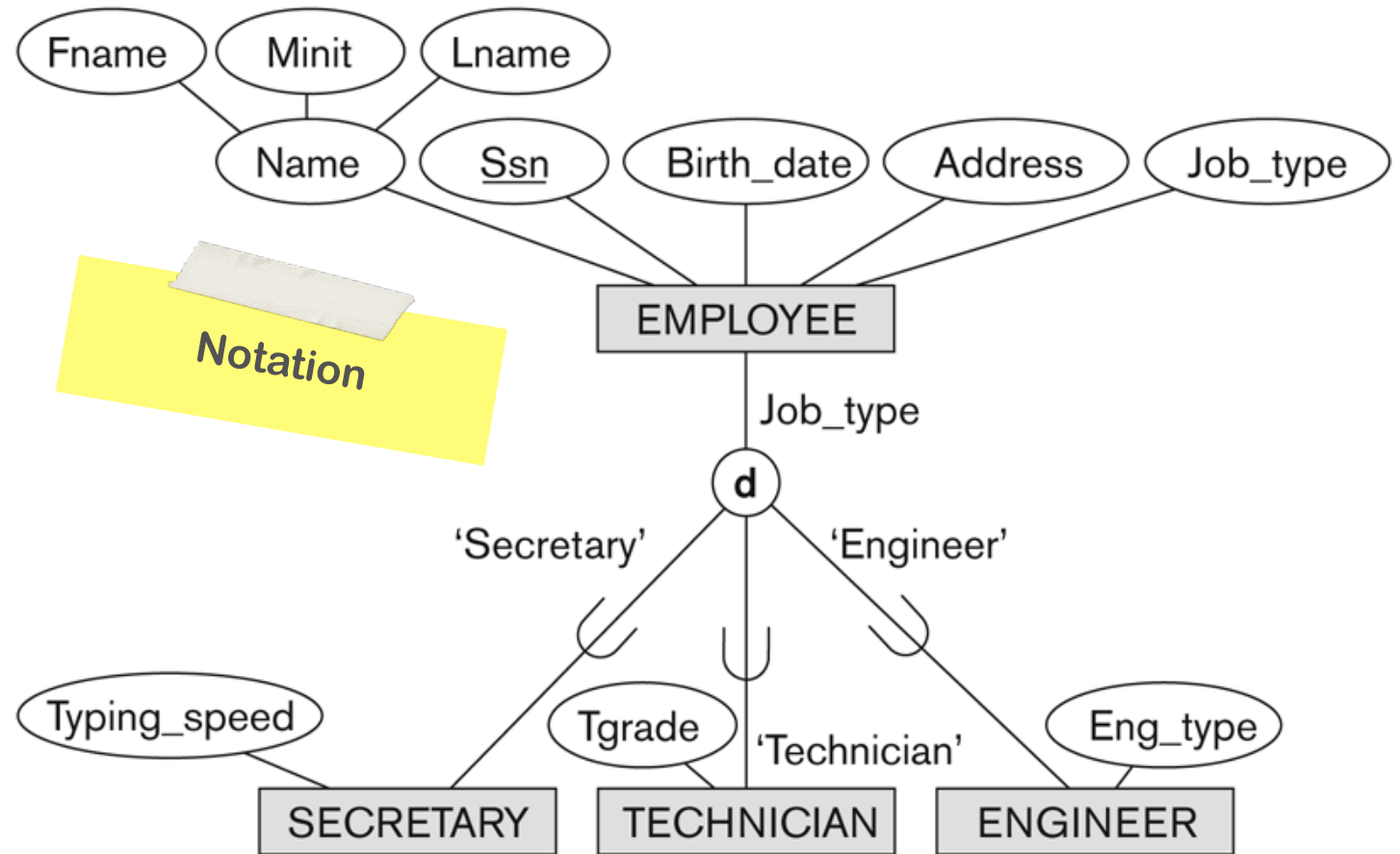
E.g., Job-type = 'Secretary'

Special case: **Attribute-defined subclasses**

all subclasses use the same attribute

E.g., Job-type = {'Secretary', 'Technician', 'Engineer'}

**Figure 4.4**  
EER diagram notation  
for an attribute-  
defined specialization  
on Job\_type.



# Basic Constraints

Two **basic constraints** can apply to a specialization/generalization:

## **Completeness Constraint / Partial**

Every entity must be a member of a subclass

Cp.: abstract / concrete superclasses in Java

## **Disjointness Constraint / Non-Overlapping**

Subclasses of the specialization must be disjoint

Entity can be member of at most one of the subclasses

# Combinations

Leads to **4 combinations**:

**Disjoint / total**

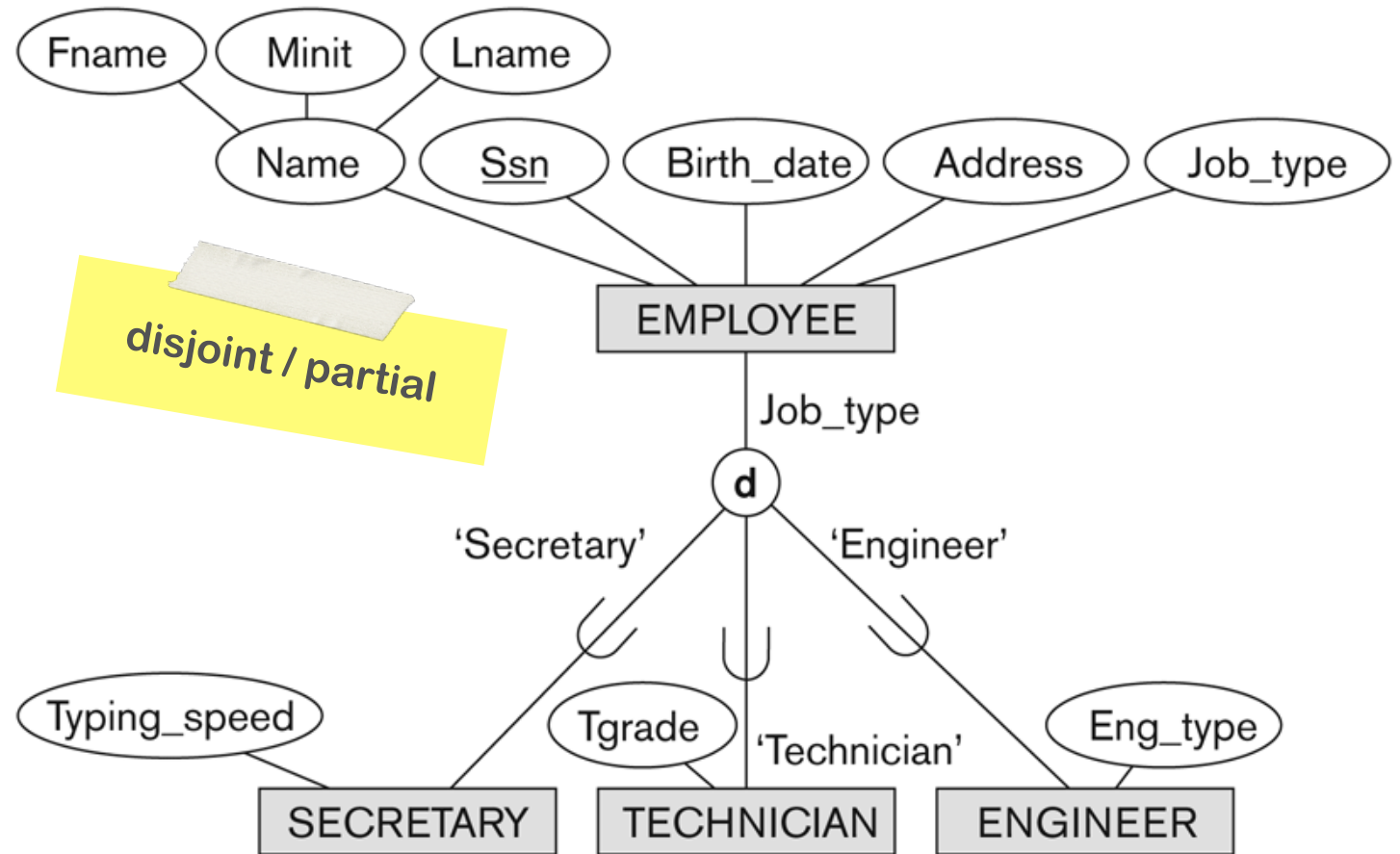
**Disjoint / partial**

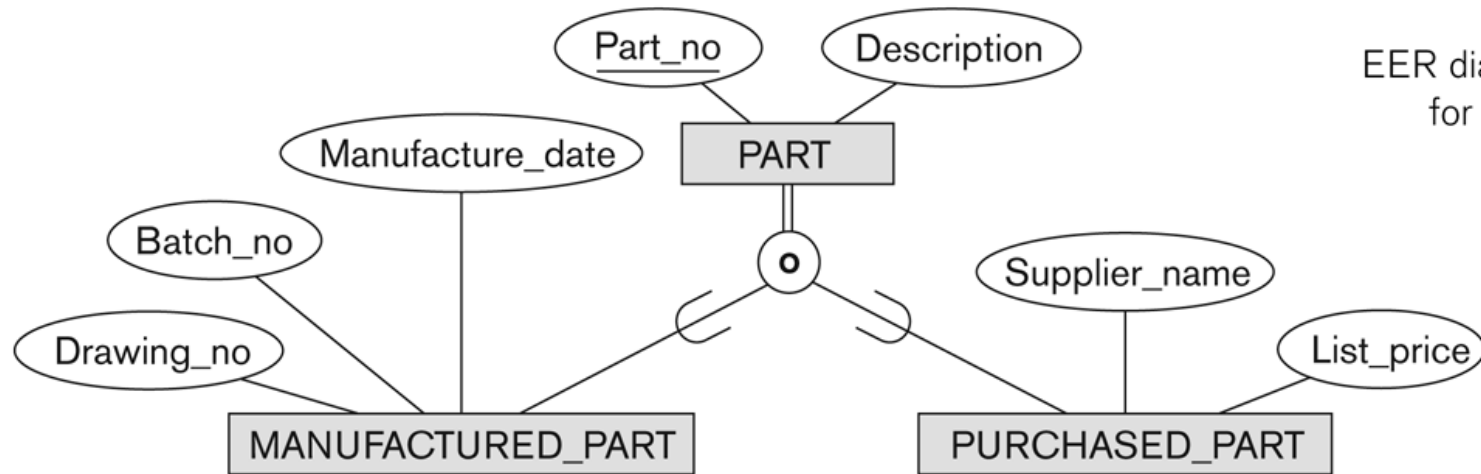
**Overlapping / total**

**Overlapping / partial**



**Figure 4.4**  
EER diagram notation  
for an attribute-  
defined specialization  
on Job\_type.





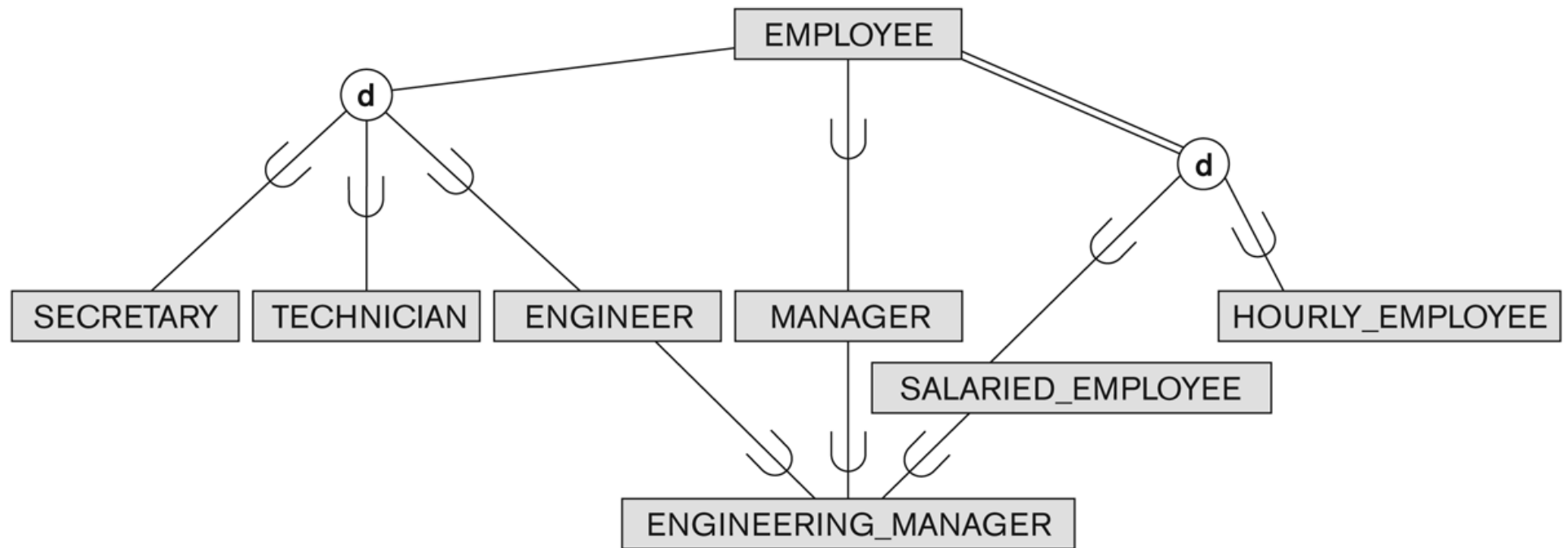
**Figure 4.5**  
EER diagram notation  
for an overlapping  
(nondisjoint)  
specialization.

overlapping /  
total

# Hierarchies and Multiple Inheritance

A subclass may itself have **further subclasses** specified on it  
Specialization hierarchy

A subclass may have **multiple superclasses**  
Multiple inheritance  
(the book calls this lattices)



**Figure 4.6**

A specialization lattice with shared subclass ENGINEERING\_MANAGER.

# Union Types

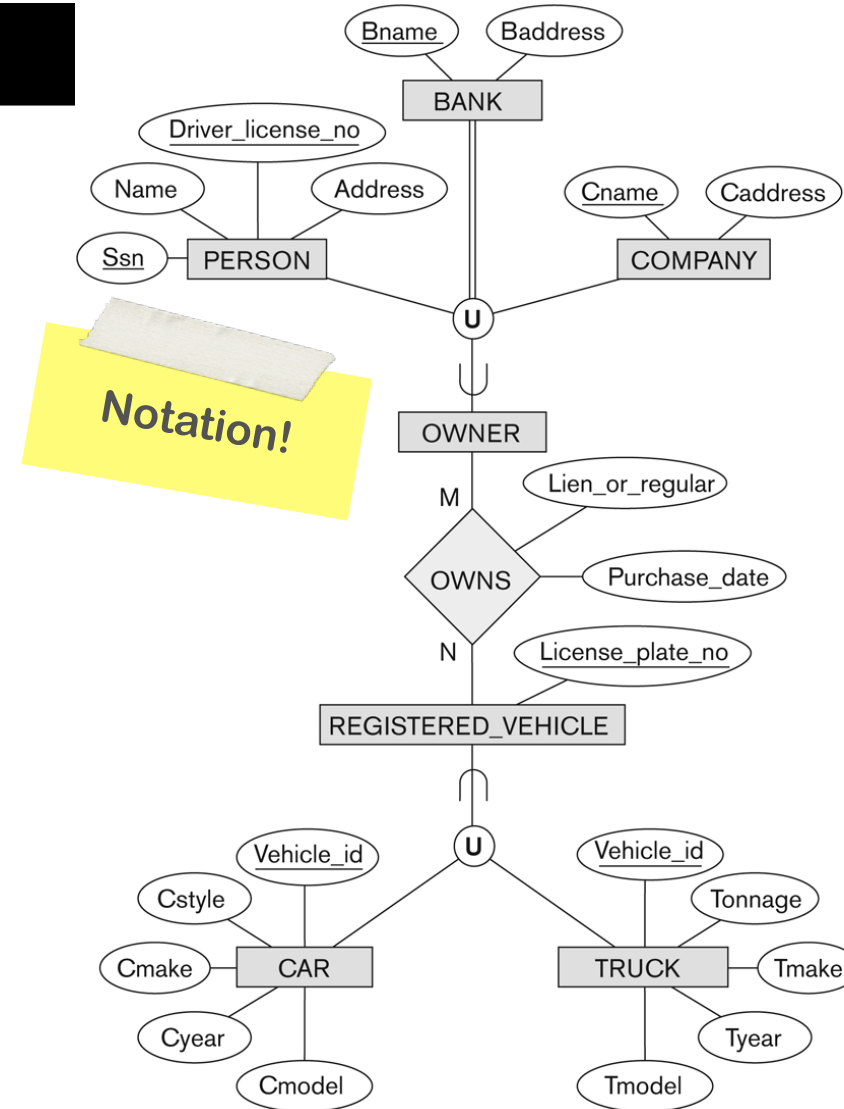
All of the subclasses so far had a **single superclass**

We may need to model a single inheritance relationship with **more than one superclass**

Such a subclass is called a **category** or **UNION TYPE**

## Example:

The owner of a registered vehicle is either a person, a bank, or a company



**Figure 4.8**  
Two categories (union types): OWNER and REGISTERED\_VEHICLE.

**Notation!**

# Another Short Quiz

# Key Takeaways

## **Basic Notation of Generalization/Specialization in EER**

IS-A Relationships

Attribute Inheritance

## **Types of Generalization/Specializations**

Total / partial

Disjoint / overlapping

## **Union types and multiple inheritance**