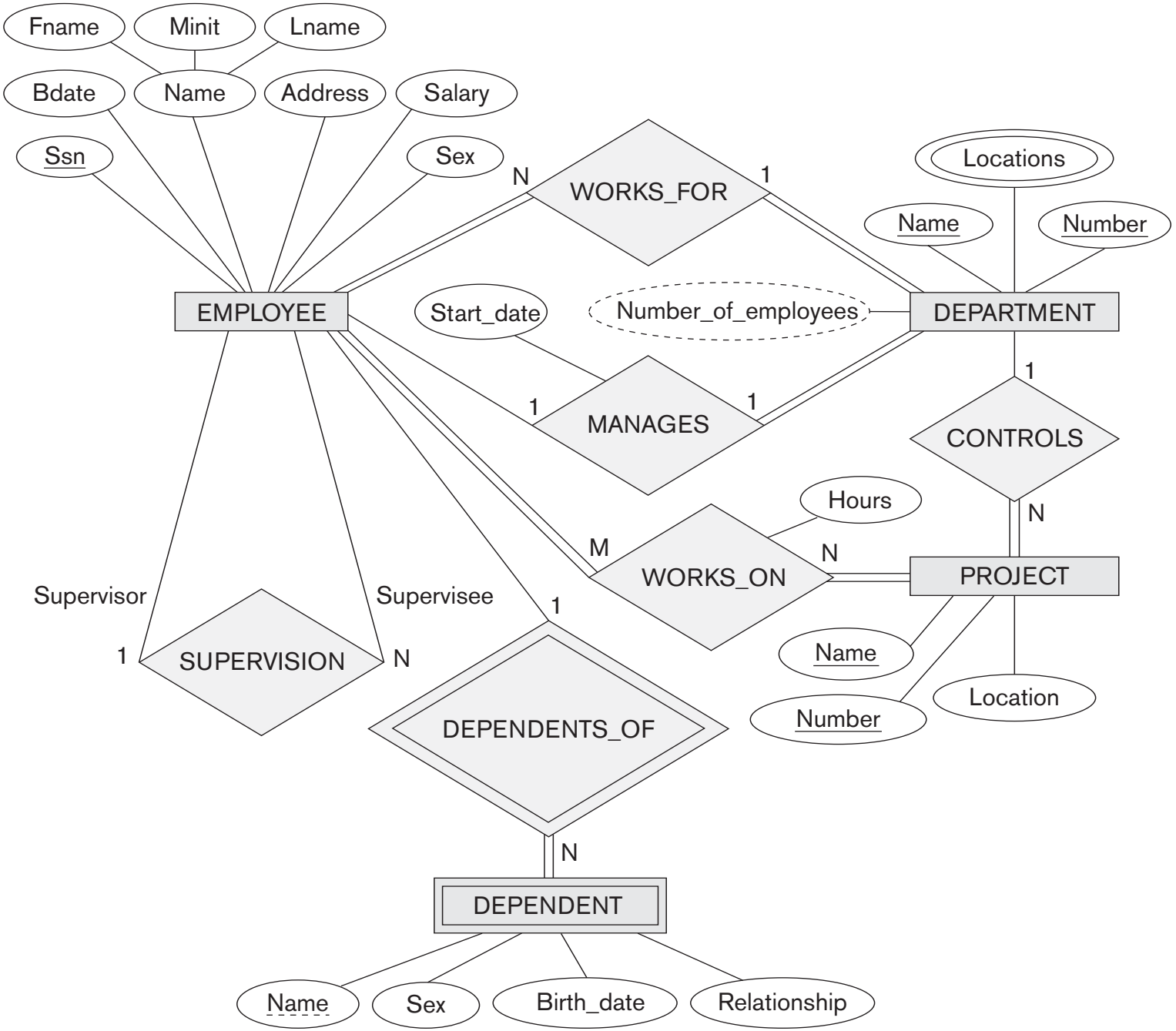
LECTURE 9:
DATABASE DESIGN (CONTINUED).

COMP2004J: Databases and Information Systems

Dr. Ruihai Dong (ruihai.dong@ucd.ie)
UCD School of Computer Science
Beijing-Dublin International College


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
WORKS_ON
N
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Location
Number
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship

Enhanced Entity Relationship Model

- In terms of data modeling, much progress has been made since the original proposal of the Entity Relationship (E-R) model.
- Particularly in the areas of **knowledge representation** and **object modeling** (e.g. for Object Oriented Programming).
- This has led to the development of the **Enhanced Entity Relationship Model** (EER).
- In particular, this adds the ability to define **supertypes** and **subtypes**.

- See Chapter 8, Elmasri & Navathe 6th ed.

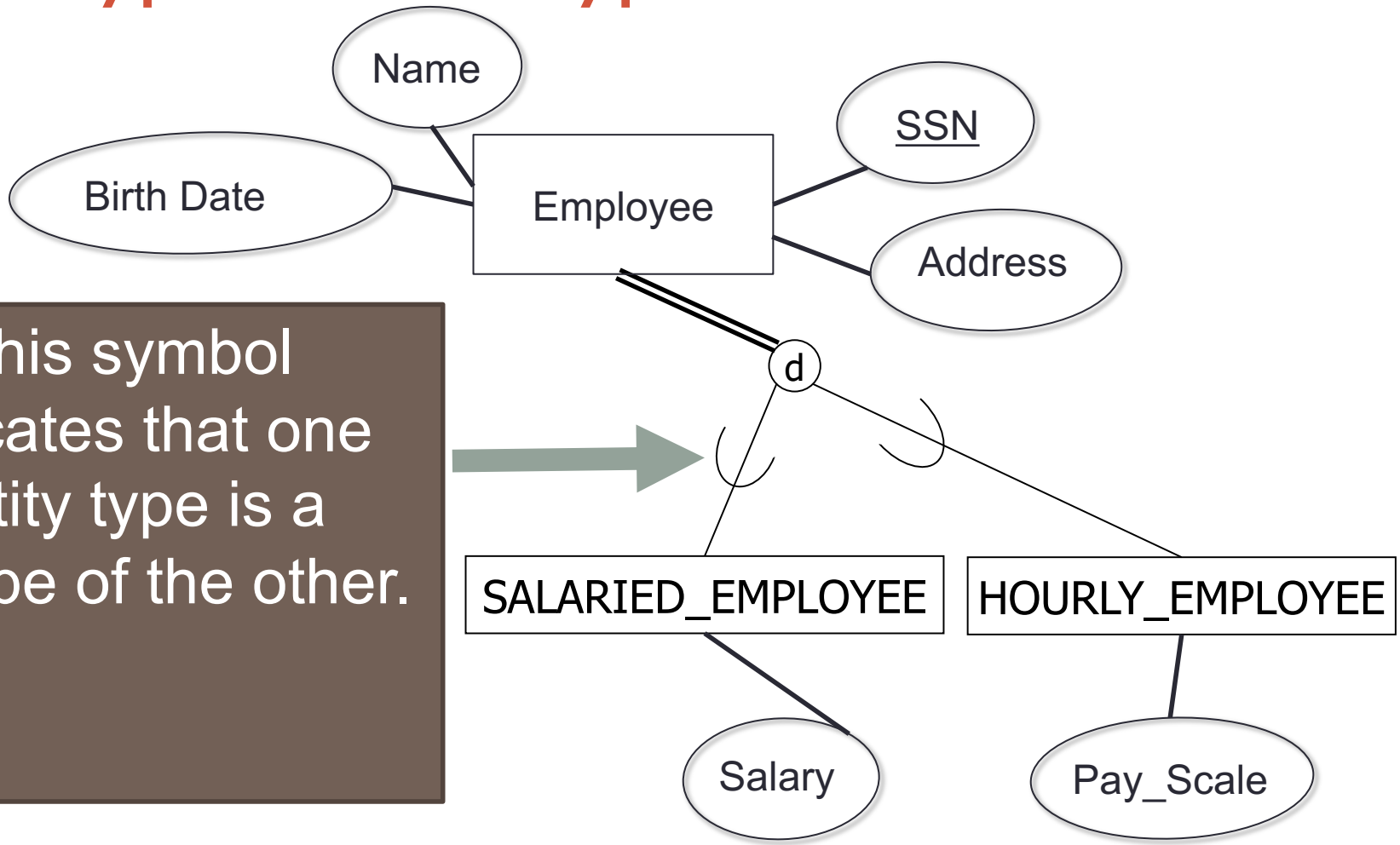
Supertypes and Subtypes

- Some entity types have subsets or subtypes that should be represented separately.
- This is most commonly seen where:
  - A subgroup/subtype has its own specific attributes that are not common to all instances of the entity type.
  - A subgroup/subtype has its own specific relationships that are not common to all instances of the entity type.

Supertypes and Subtypes

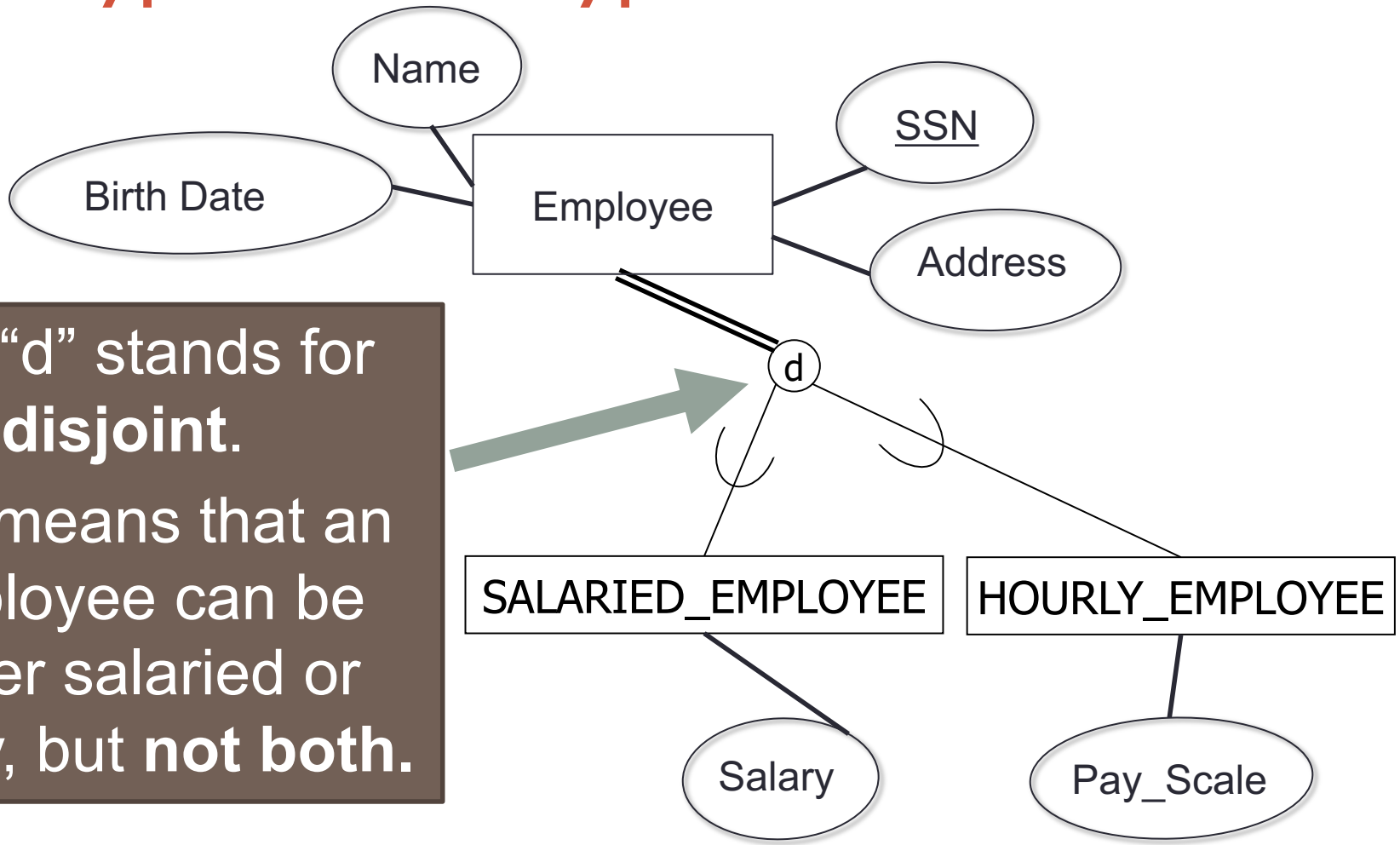
- Example:
  - In a company, we store data about an EMPLOYEE entity type.
  - Employees can be paid **per hour**, or using an **annual salary**.
    - A salaried employee has a “salary” attribute that hourly employees do not have.
    - An hourly employee has a “hourly rate” attribute that salary employees do not have.
  - This suggests that the “employee” entity type should have two **subtypes: SALARIED_EMPLOYEE** and **HOURLY_EMPLOYEE**
  - Let’s see an example:

Supertypes and Subtypes



This symbol indicates that one entity type is a subtype of the other.

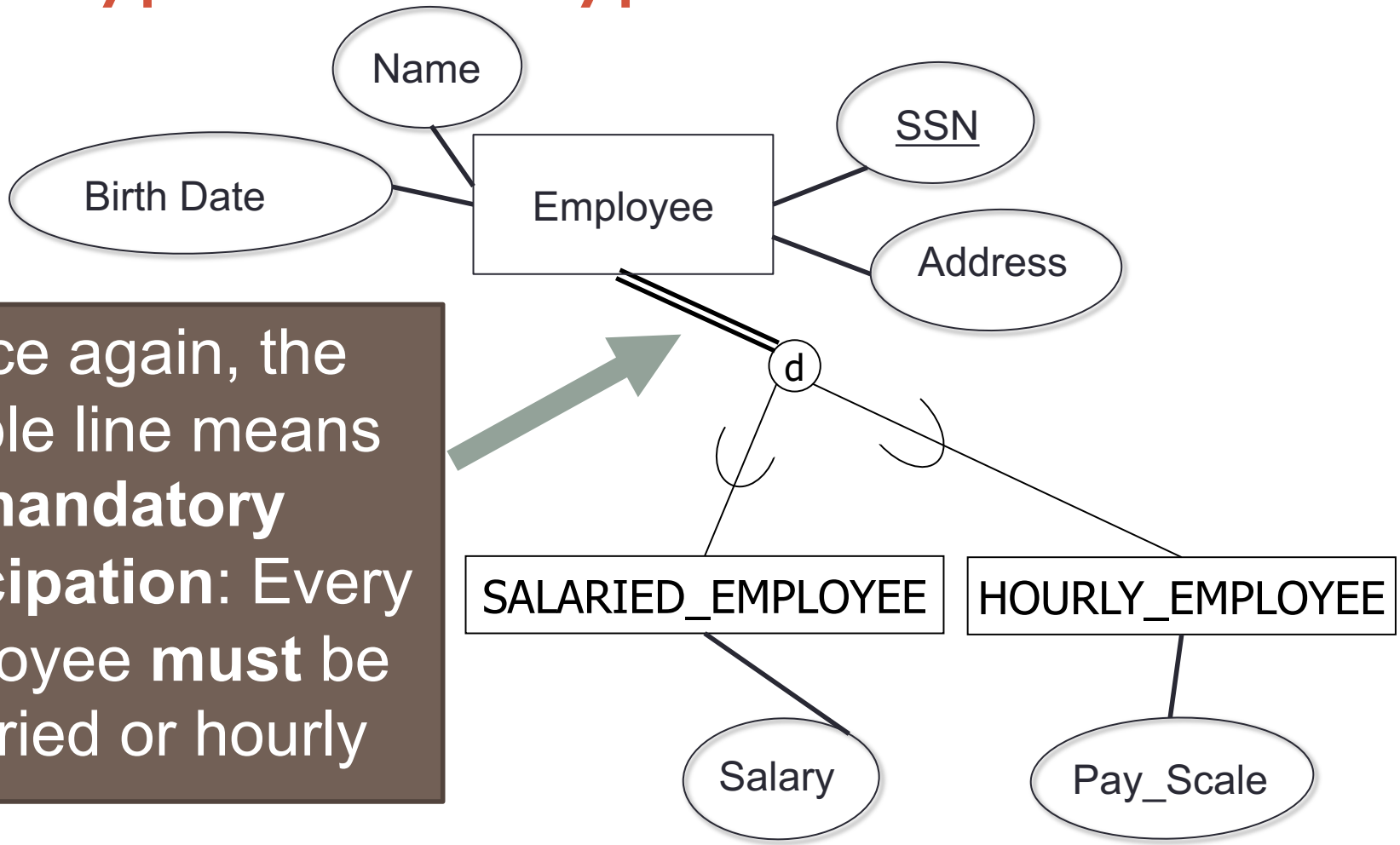
Supertypes and Subtypes



The "d" stands for **disjoint**.

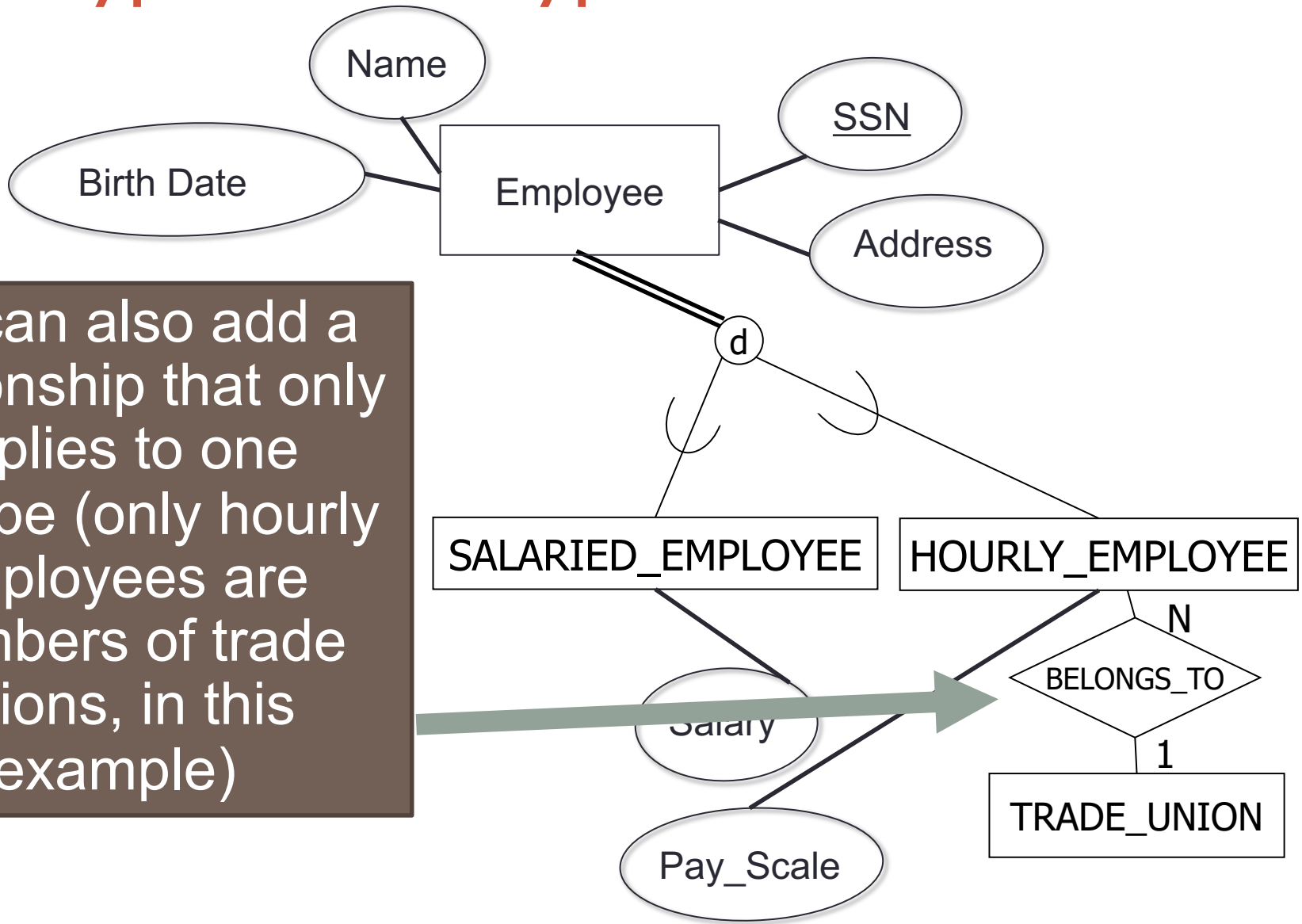
This means that an employee can be either salaried or hourly, but **not both.**

Supertypes and Subtypes



Once again, the double line means **mandatory participation**: Every employee **must** be salaried or hourly

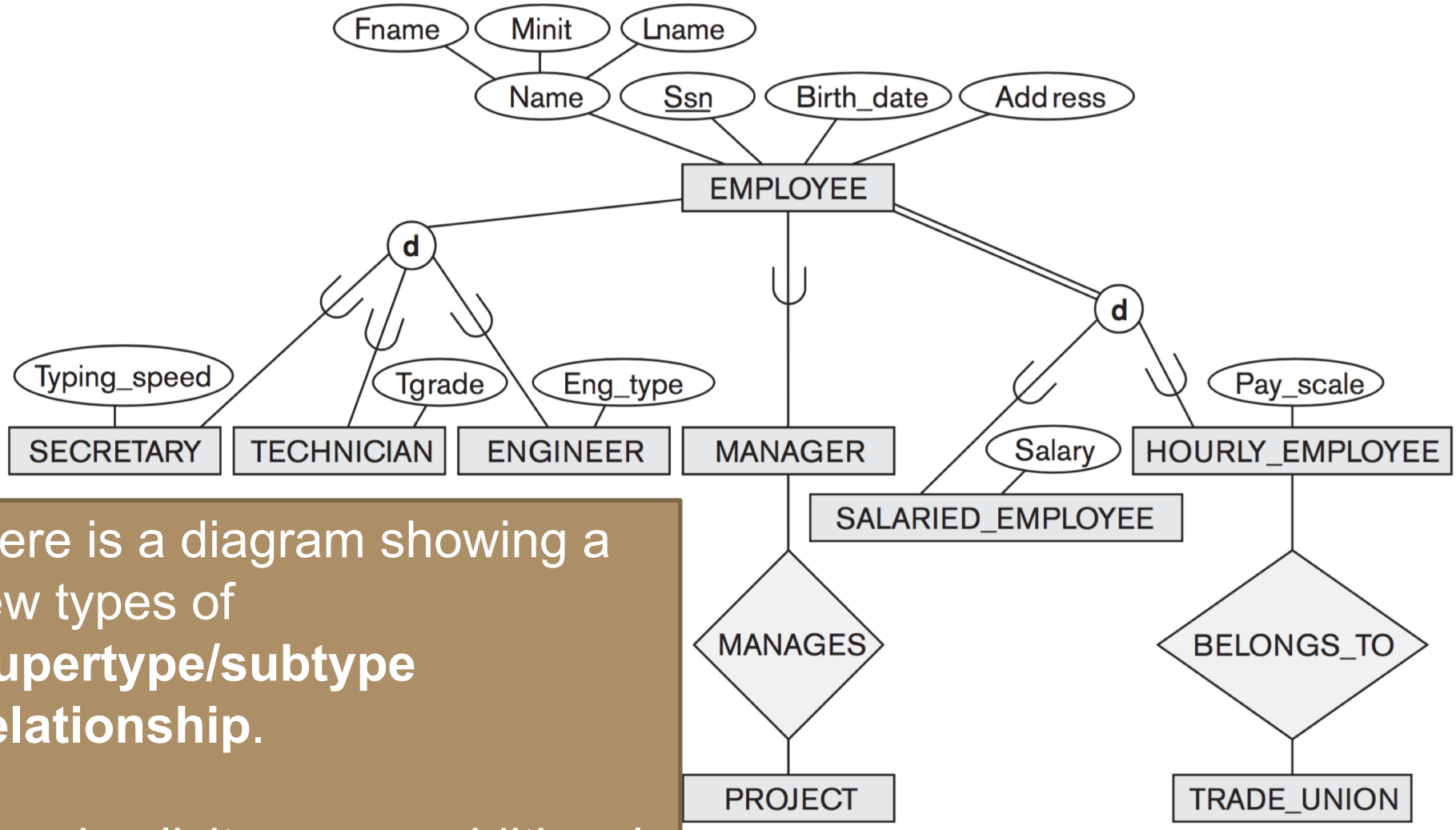
Supertypes and Subtypes



We can also add a relationship that only applies to one subtype (only hourly employees are members of trade unions, in this example)

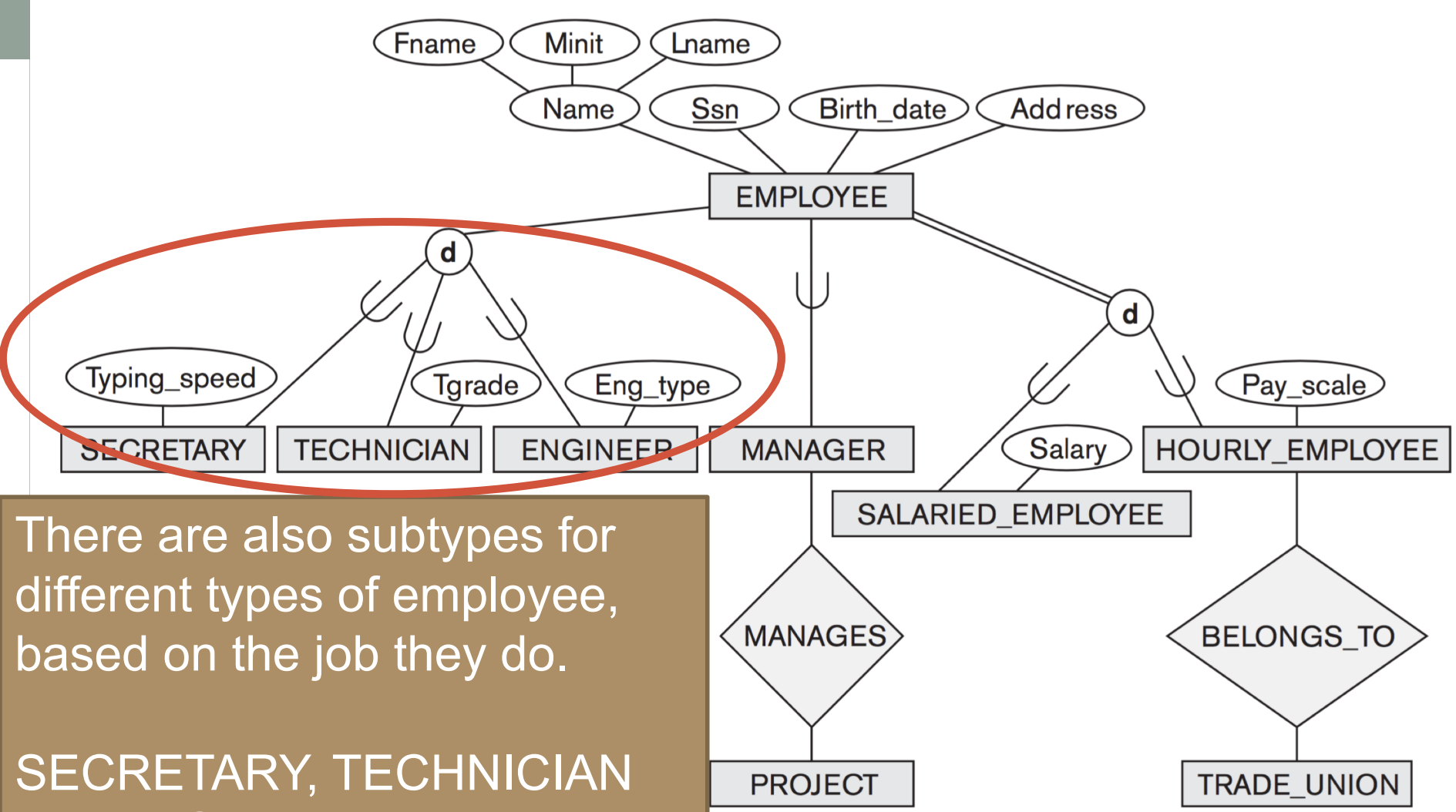
Inheritance

- Every occurrence of a subtype is also an occurrence of the supertype.
  - May seem strange: one particular person is represented both by an EMPLOYEE entity and a SALARIED_EMPLOYEE entity.
- Every property of the supertype (attributes, keys, relationships, etc.) is also a property of the subtype.
- This is known as **inheritance** (similar to in OOP).



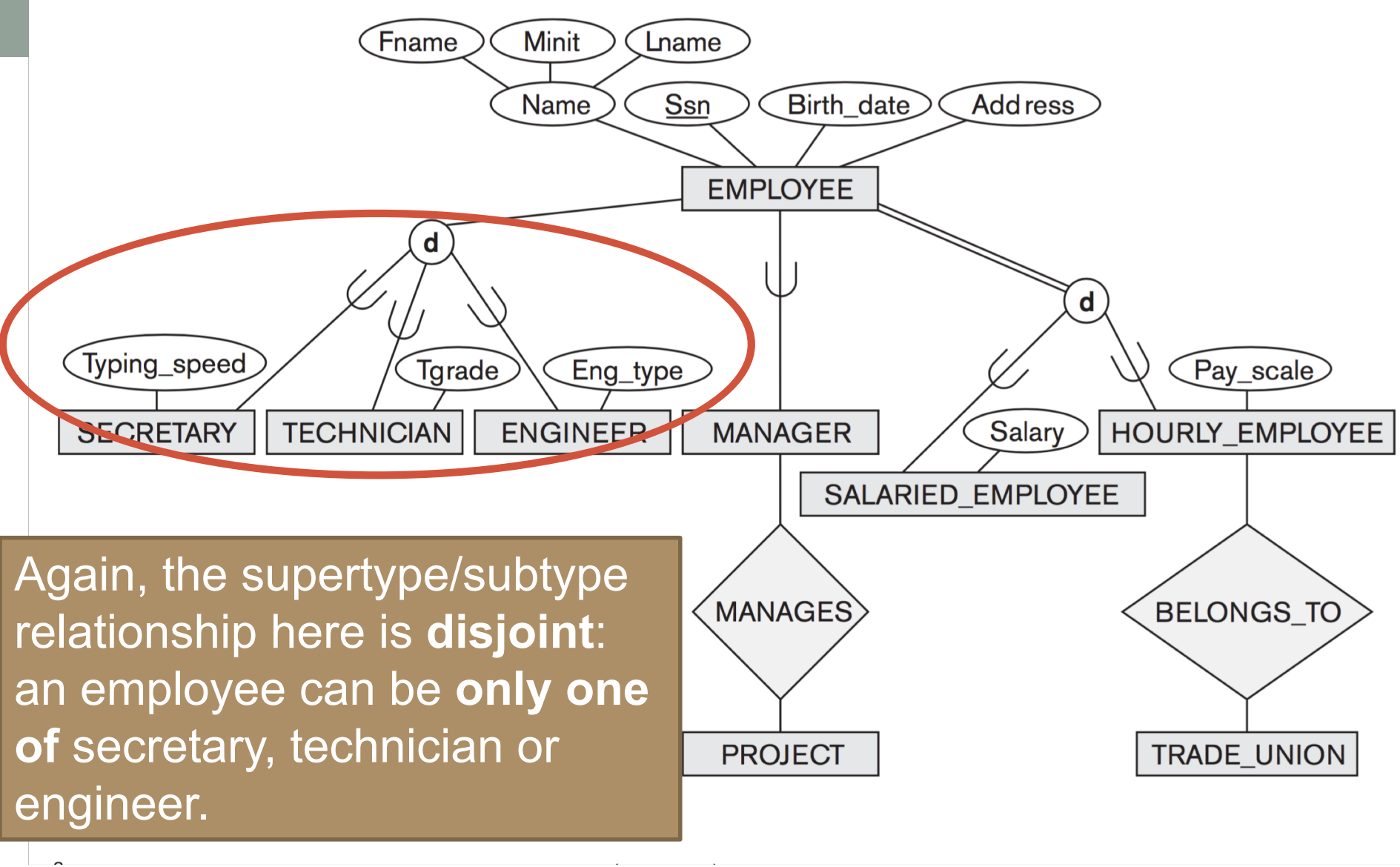
Here is a diagram showing a few types of **supertype/subtype relationship**.

For simplicity, some additional attributes, and the cardinality of relationships are not shown.

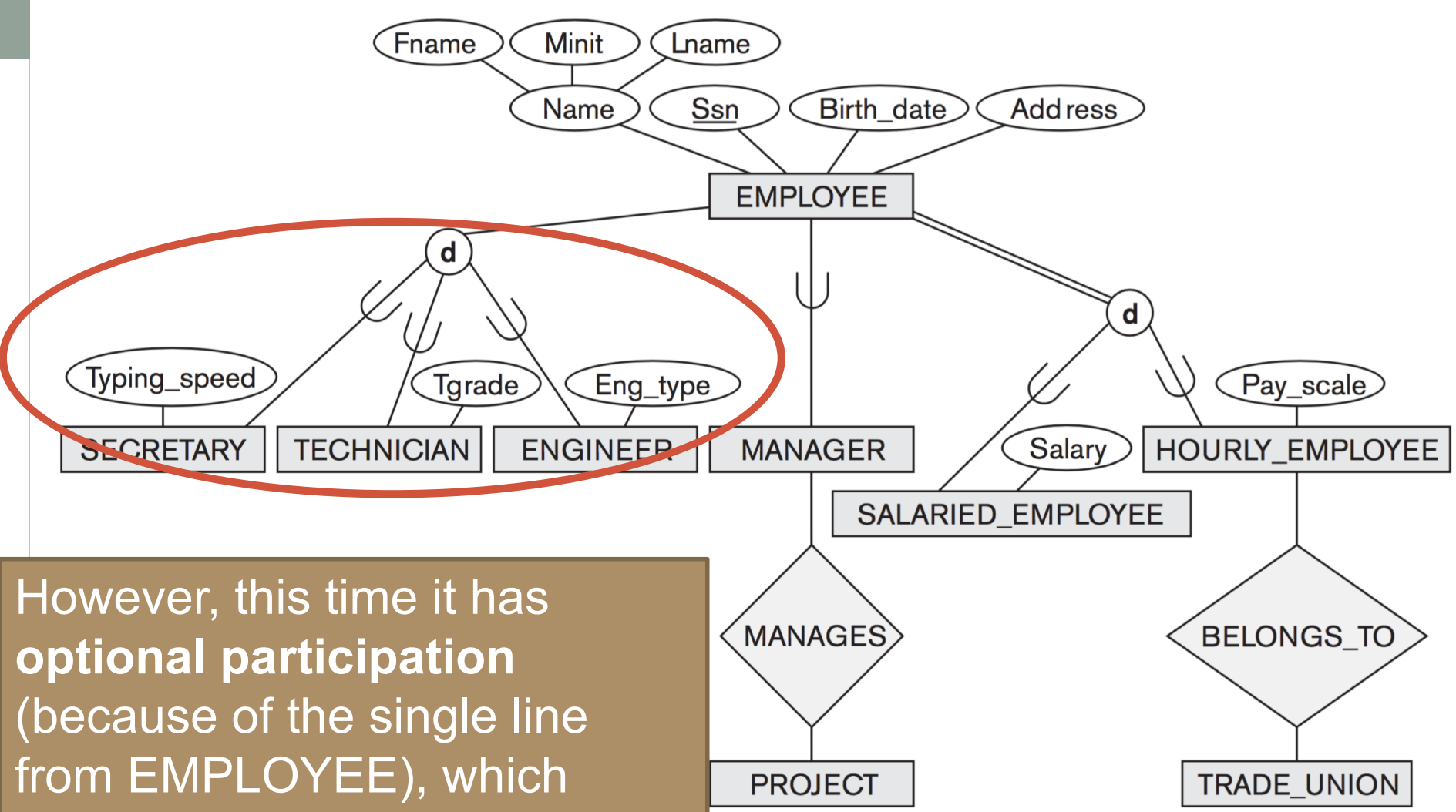


There are also subtypes for different types of employee, based on the job they do.

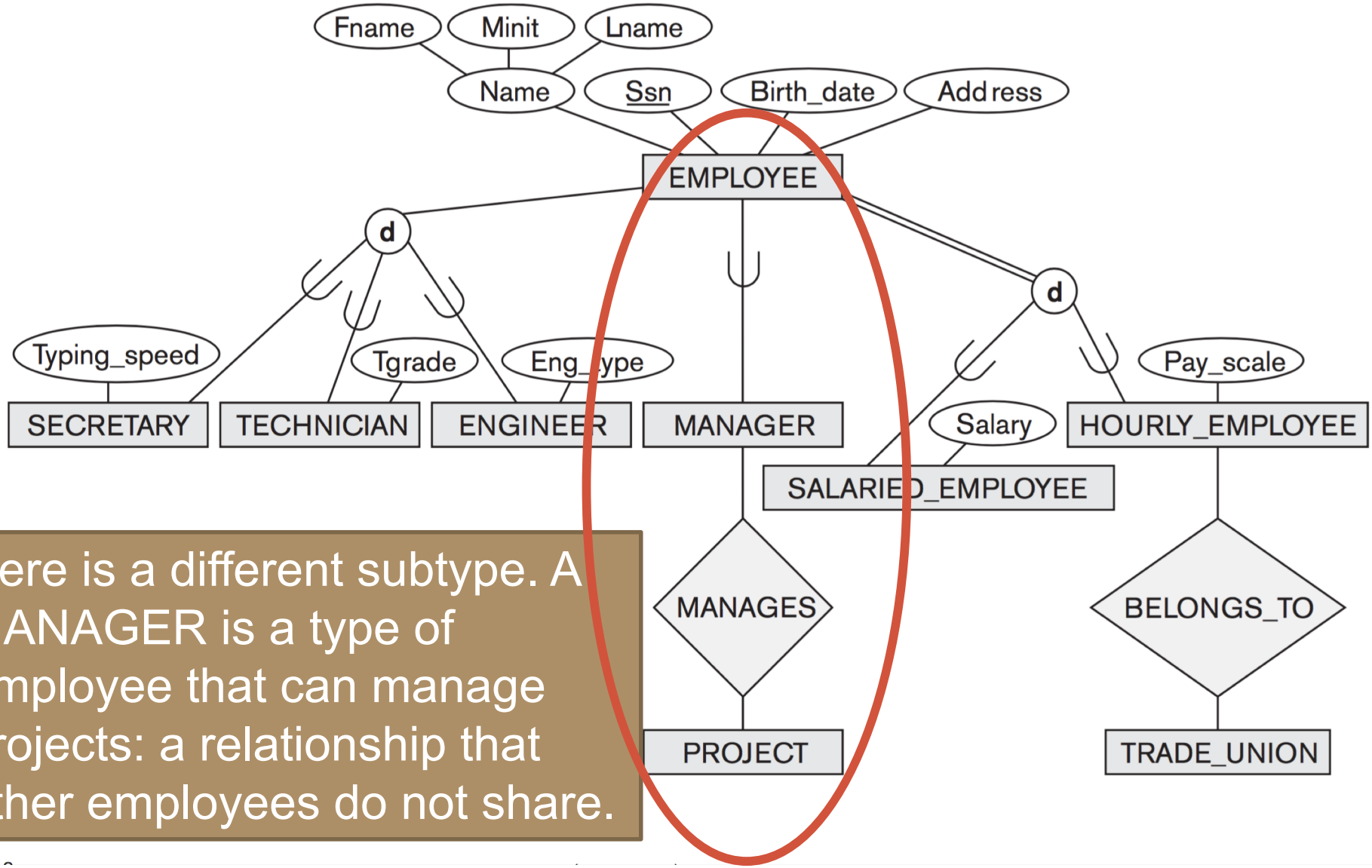
SECRETARY, TECHNICIAN and ENGINEER each have some additional attribute that not all employees have.



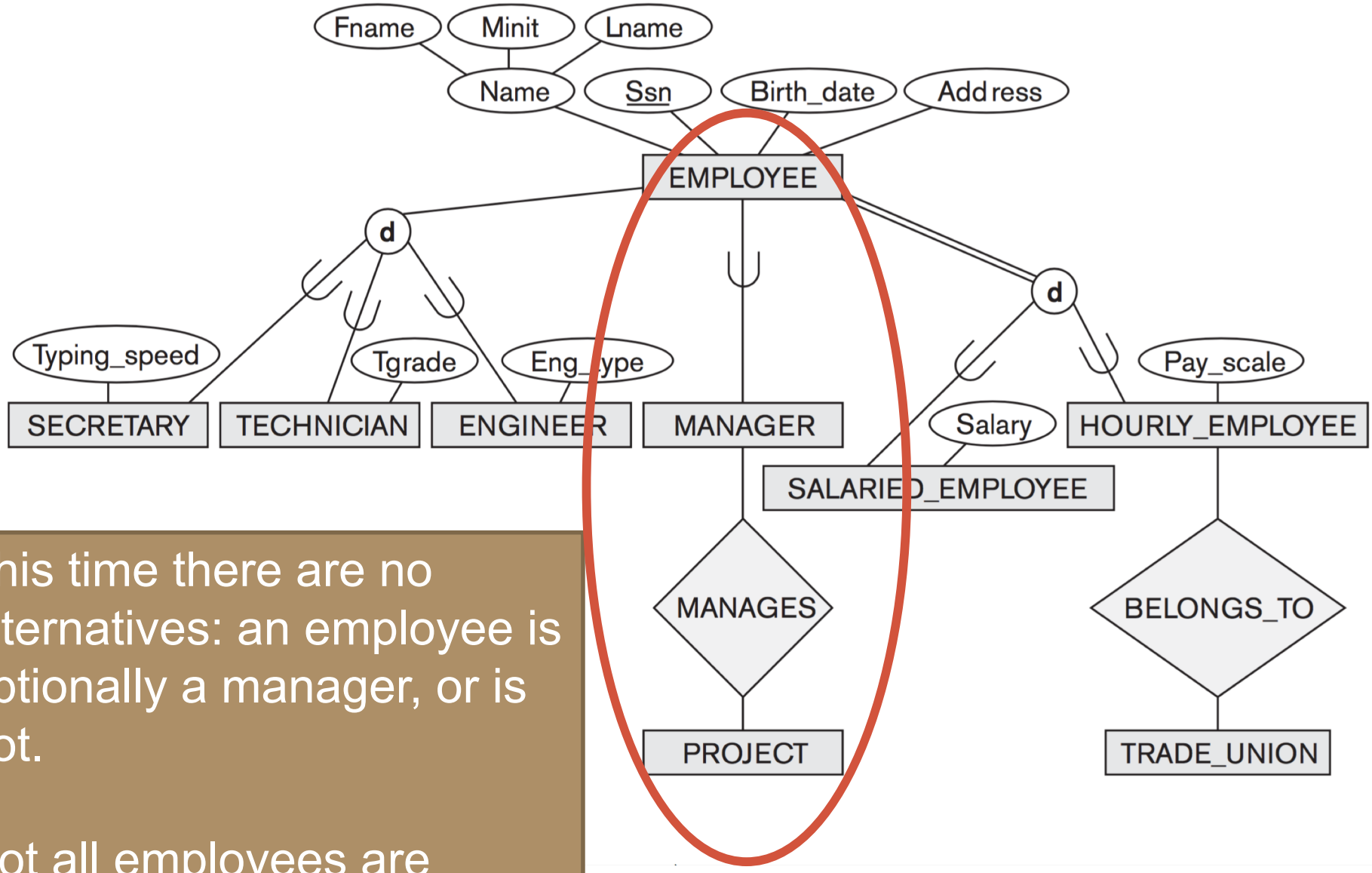
Again, the supertype/subtype relationship here is **disjoint**: an employee can be **only one of** secretary, technician or engineer.



However, this time it has **optional participation** (because of the single line from EMPLOYEE), which means that some employees might not be any of those subtypes.



Here is a different subtype. A MANAGER is a type of employee that can manage projects: a relationship that other employees do not share.



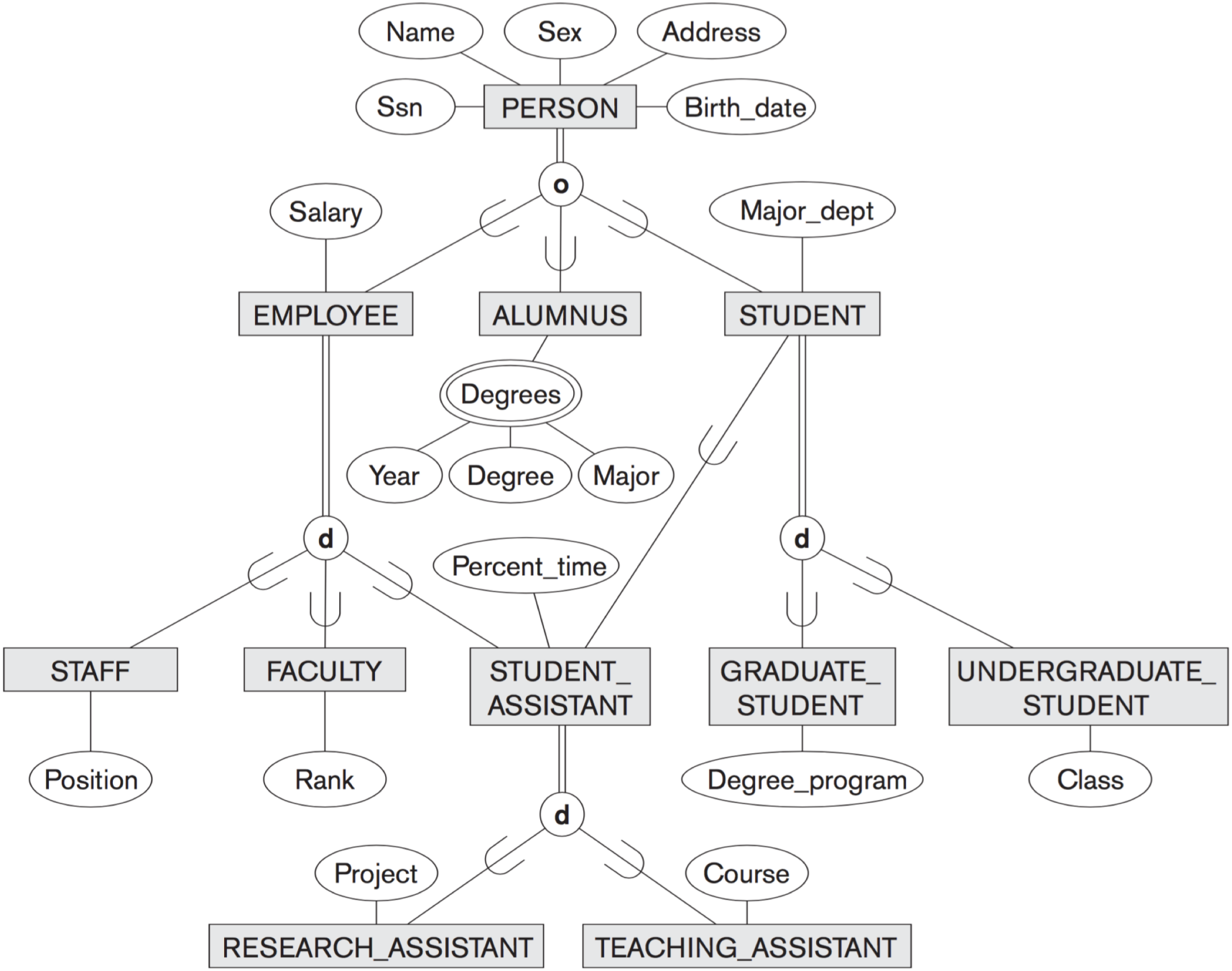
This time there are no alternatives: an employee is optionally a manager, or is not.

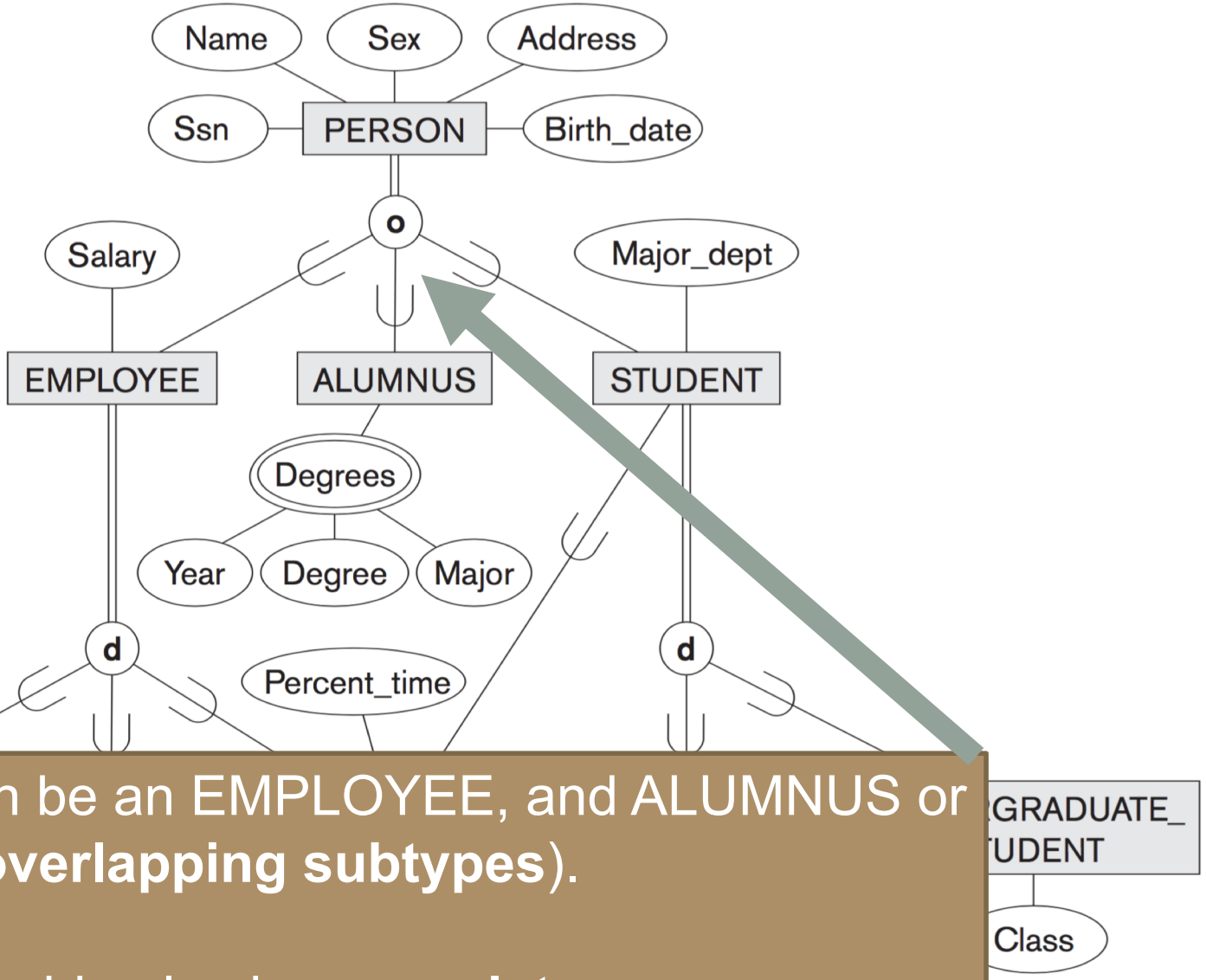
Not all employees are managers.

Disjoint and Overlapping Subtypes

- The subtypes we have seen so far have all been **disjoint**: meaning that a particular entity (e.g. an employee) can only belong to one of a group of subclasses.
- An alternative is to have **overlapping** subtypes, where an entity could be a member of multiple related entity types.
- For example, a student who does work in a university may be a member of a PERSON supertype, but be both a STUDENT and an EMPLOYEE.

- Here's an example…


Name
Sex
Address
Ssn
PERSON
Birth_date
o
Salary
Major_dept
EMPLOYEE
ALUMNUS
STUDENT
Degrees
Year
Degree
Major
d
Percent_time
d
STAFF
FACULTY
STUDENT_
ASSISTANT
GRADUATE_
STUDENT
UNDERGRADUATE_
STUDENT
Position
Rank
Degree_program
Class
d
Project
Course
RESEARCH_ASSISTANT
TEACHING_ASSISTANT

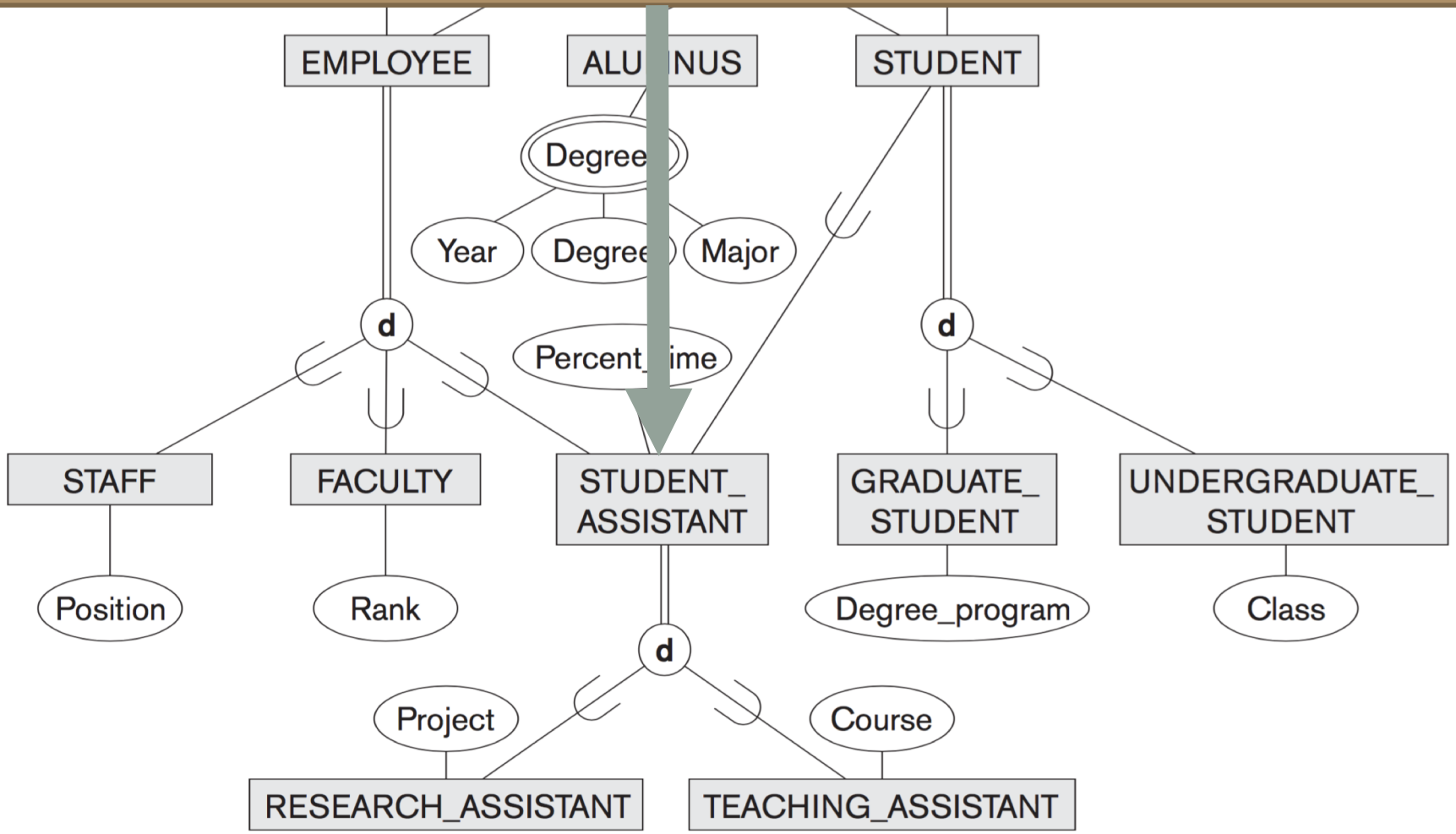


A PERSON can be an EMPLOYEE, and ALUMNUS or a STUDENT (**overlapping subtypes**).

But this relationship also has **mandatory participation** (double line), so every PERSON must be at least one of these.

Also note that the Enhanced E-R diagram allows for **multiple inheritance**.

A STUDENT_ASSISTANT is both a STUDENT and an EMPLOYEE.



Specialisation and Generalisation

- **Specialisation** and **Generalisation** are **processes** that can be used to identify supertypes and subtypes.
- For **specialisation**, we begin with a supertype, and find any attributes/relationships that are particular to certain members. Groups of members with common attributes/relationships become subtypes.
- **Generalisation** is the opposite, where we begin with specific types, and identify attributes and relationships they have in common, so as to create a supertype.

CREATING AN E-R MODEL

Example of Requirements for a Database

We wish to create a database for a company that runs training courses. For this, we must store data about the trainees and the instructors. For each course participant (about 5000), identified by a code, we want to store the social security number, surname, age, sex, place of birth, employer' s name, address and telephone number, previous employers (and period employed), the courses attended (there are about 200 courses) and the final assessment of each course. We need also to represent the seminars that each participant is attending at present and, for each day, the places and times the classes are held. Each course has a code and a title and any course can be given any number of times. Each time a particular course is given, we will call it an 'edition' of the course. For each edition, we represent the start date, the end date, and the number of participants. If a trainee is a self-employed professional, we need to know his or her area of expertise, and, if appropriate, his or her title. For somebody who works for a company, we store the level and position held. For each instructor (about 300), we will show the surname, age, place of birth, the edition of the course taught, those taught in the past and the courses that the tutor is qualified to teach. All the instructors' telephone numbers are also stored. An instructor can be permanently employed by the training company or can be freelance.

It can be useful to turn this natural language description into a **glossary of terms**, which helps to describe the concepts we want to store data about, and their relationships to one another.

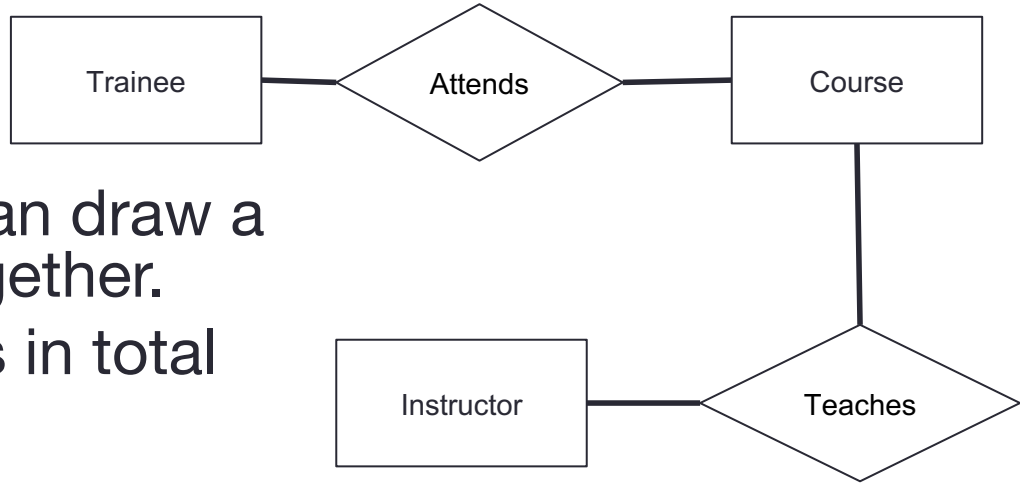
Term	Description	Synonyms	Links
Trainee	Participant in a course. Can be an employee or self-employed	Participant, Student	Course, Employer
Instructor	Course tutor. Can be freelance.	Tutor, Teacher	Course
Course	Course offered. Can have various editions.	Seminar	Instructor, Trainee
Employer	Company by which a trainee is employed or has been employed.		Trainee

General criteria for data representation

- If a concept has significant properties and/or describes classes of objects with an autonomous existence, it is appropriate to represent it by an entity.
- If a concept has a simple structure, and has no relevant properties associated with it, it is convenient to represent it by an attribute of another concept to which it refers.
- If the requirements contain a concept that provides a logical link between two (or more) entities, it is convenient to represent this concept by a relationship.
- If one or more concepts are particular cases of another concept, it is convenient to represent them by means of a supertype/subtype relationship.

Basic schema and strategy for a training company

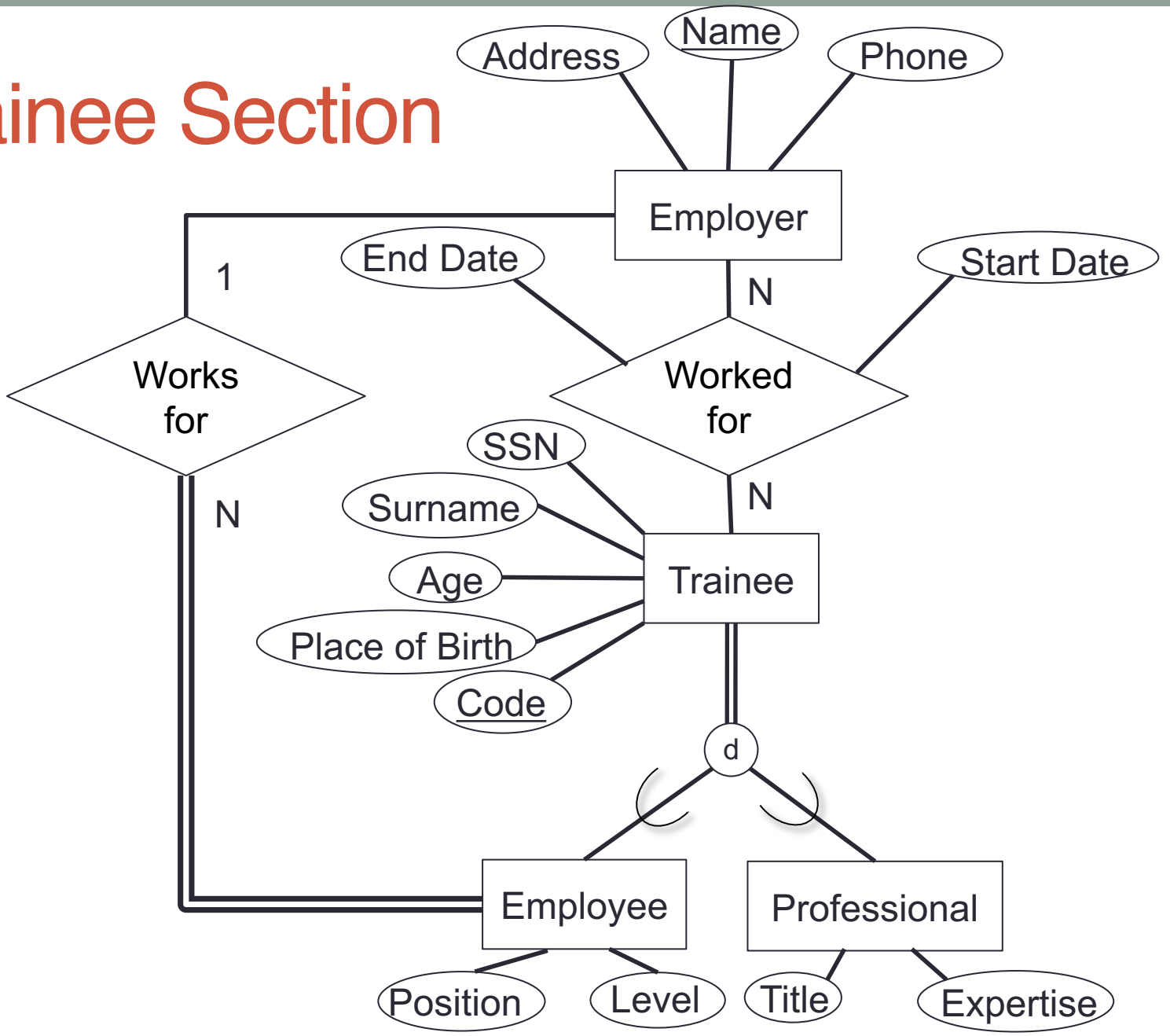
- Because E-R diagrams can get quite large, we do not need to squeeze everything into one diagram.
- We can divide our work into three sections:
  - Trainee section.
  - Course section.
  - Instructor section.
- After completing these, we can draw a diagram to connect these together.
- We will finish with 4 diagrams in total



Trainee Section

- For each course participant (about 5000), identified by a code, we want to store the **social security number**, **surname**, **age**, **sex**, **place of birth**, **employer' s name**, **address** and **telephone number**, **previous employers** (and **period employed**), the courses attended (there are about 200 courses) and the final assessment of each course.
- If a trainee is a **self-employed professional**, we need to know his or her **area of expertise**, and, if appropriate, his or her **title**. For somebody who **works for a company**, we store the **level** and **position** held.

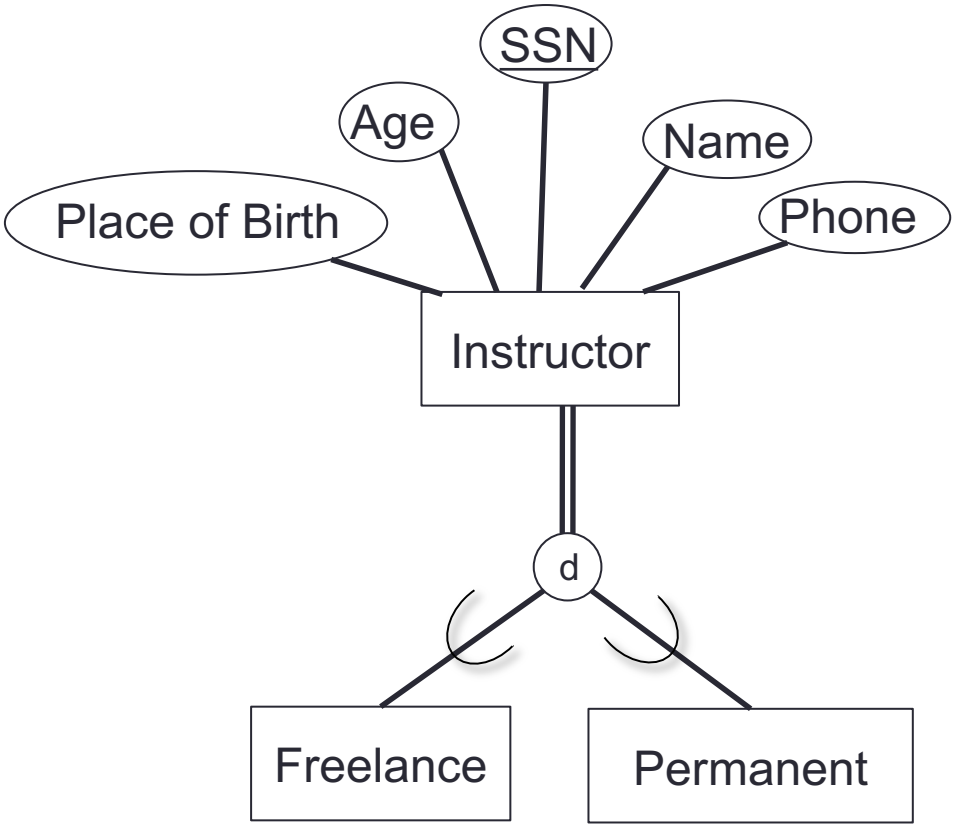
Trainee Section



Instructor Section

For each instructor (about 300), we will show the **name**, **age**, **place of birth**, the **edition** of the course taught, those **taught in the past** and the courses that the tutor is **qualified to teach**. All the instructors' **telephone numbers** are also stored. An instructor can be **permanently employed** by the training company or can be **freelance**.

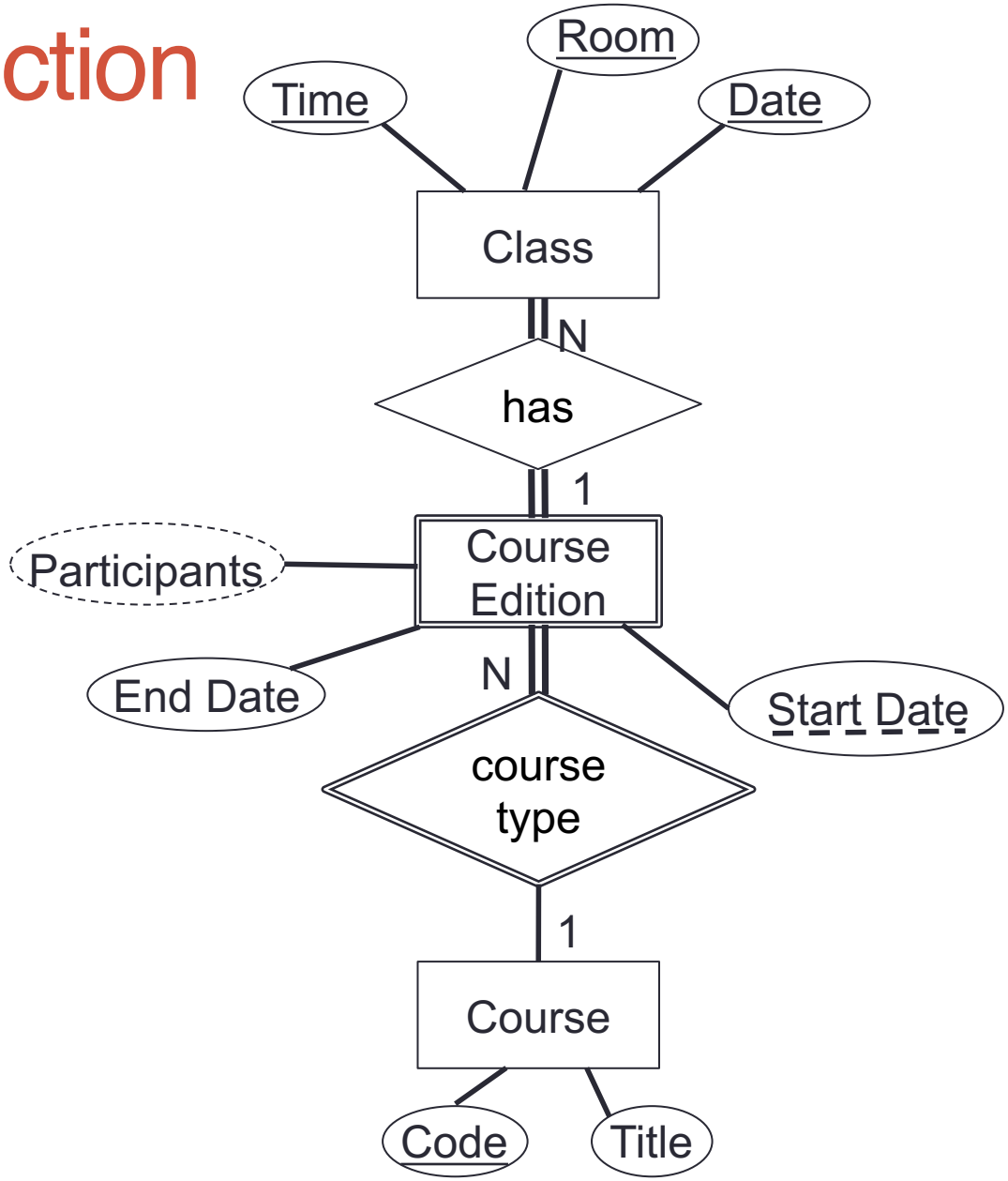
Instructor Section



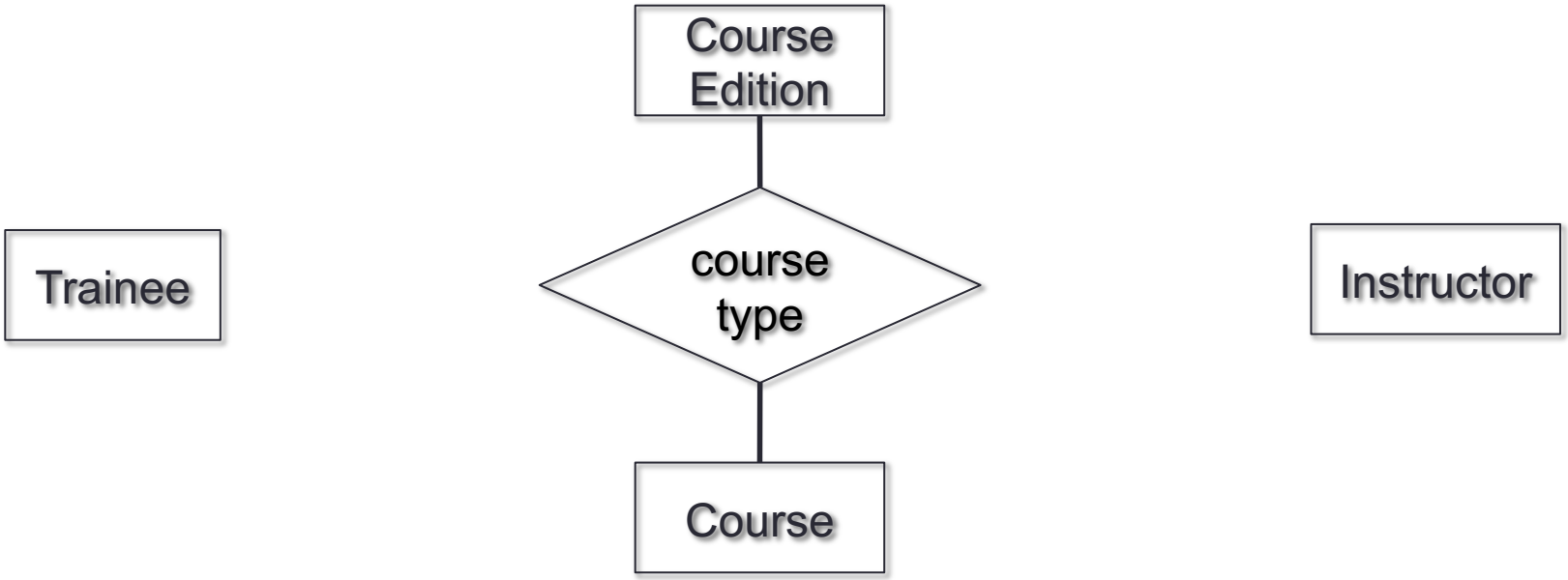
Course Section

- We need also to represent the **seminars** that each participant is attending at present and, for each **day**, the **places** and **times** the **classes** are held. Each course has a **code** and a **title** and any course can be given any number of times. Each time a particular course is given, we will call it an '**edition**' of the course. For each edition, we represent the **start date**, the **end date**, and the **number of participants**.

Course Section



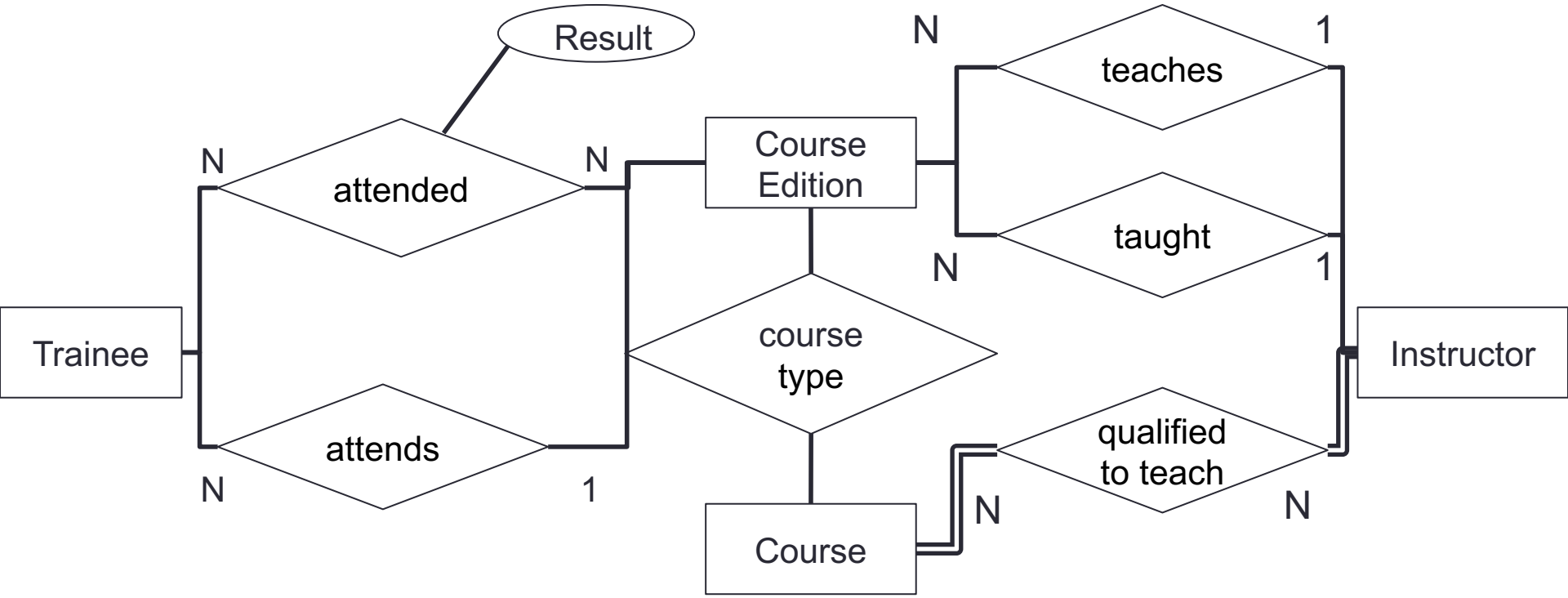
Full Schema



Full Schema

We wish to create a database for a company that runs training courses. For this, we must store data about the trainees and the instructors. For each course participant (about 5000), identified by a code, we want to store the … the **courses attended** (there are about 200 courses) and the **final assessment** of each course. We need also to represent the **seminars** that each participant **is attending** at present and, for each day, the places and times the classes are held. For each **instructor** (about 300), we will show …the **edition of the course taught**, those **taught in the past** and the courses that the tutor is **qualified to teach**.

Full Schema


Result
N
1
teaches
N
attended
N
Course
Edition
taught
N
1
Trainee
course
type
Instructor
attends
qualified
to teach
N
1
N
N
Course