DnA Tutorial 2

Anish R Joishy

16/10/2024

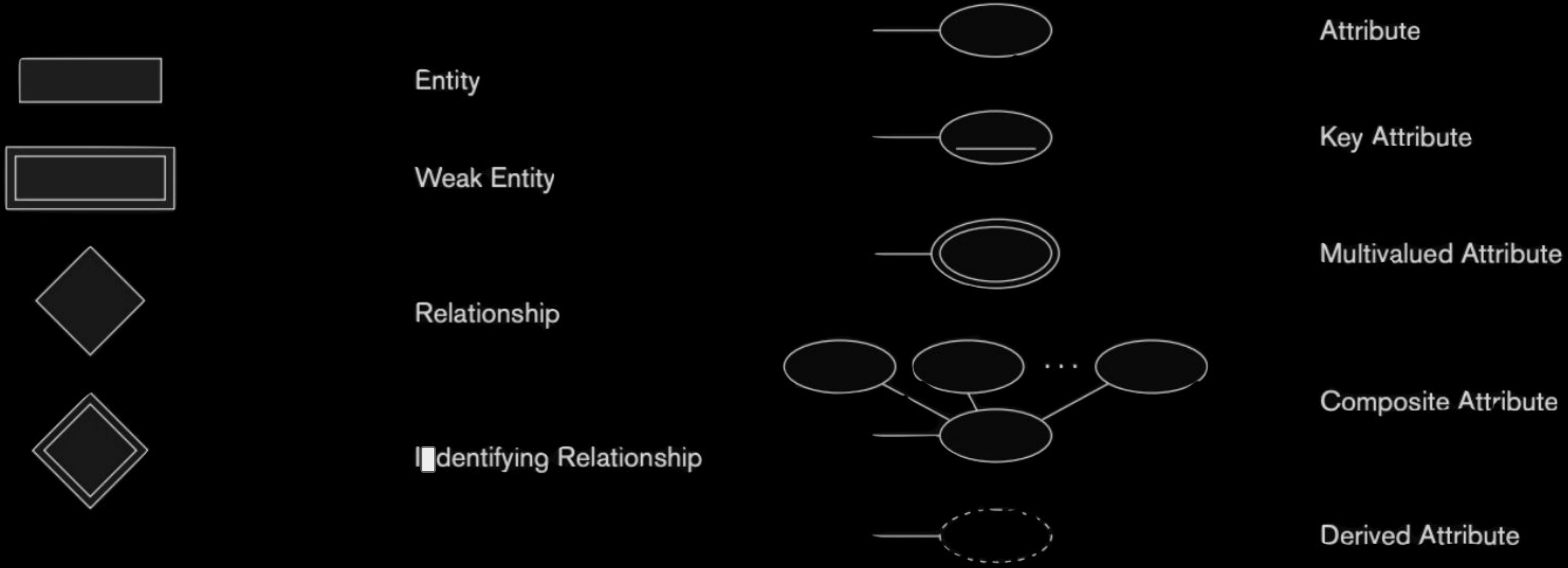
Recap (Lectures)

- An important advantage of DBMS approach is to represent complex relationships among data.
- Data Model - A set of concepts to describe the structure of a database, the operations for manipulating these structures, and certain constraints that the database should obey.
- Data models are structured in a specific way - they have groups of elements (typically called entities) and relationships among such groups.
- You would have covered categories of data models in class like conceptual, physical and implementation data models. Entity Relationship (ER) modeling comes under conceptual - High level, entity-based model that provides concepts close to the way many users perceive data.

Recap (Tutorials)

- ER Model has entities/entity types, relationships and attributes.
- Entities are instances of entity types, for example CAR is an entity type but CAR A is an entity. We have strong and weak entity types. Examples of weak entities are like DEPENDENTS (anyone financially dependent on the employee) if we have a COMPANY database, with EMPLOYEE as strong entity type.
- Entities of course have attributes. They may be simple, composite, derived, multi-valued, key attributes (unique among all entities in an entity type). Relationships can have attributes too, especially if the attributes depend uniquely on the instances from each entity taking part in the relationship.
- Relationships are what bind entity types together. In fact, whenever an attribute of one entity type refers to another entity type, some relationship exists.
- Cardinality ratios: 1 medical department has N surgeons (1:N on relationship DEPTS HAVE SURGEONS). Cardinality ratio specifies the maximum number of relationship instances that an entity can participate in. Each department (1) can have any number of surgeons (N).

Notations


Entity
Weak Entity
Relationship
Identifying Relationship
Attribute
Key Attribute
Multivalued Attribute
Composite Attribute
Derived Attribute

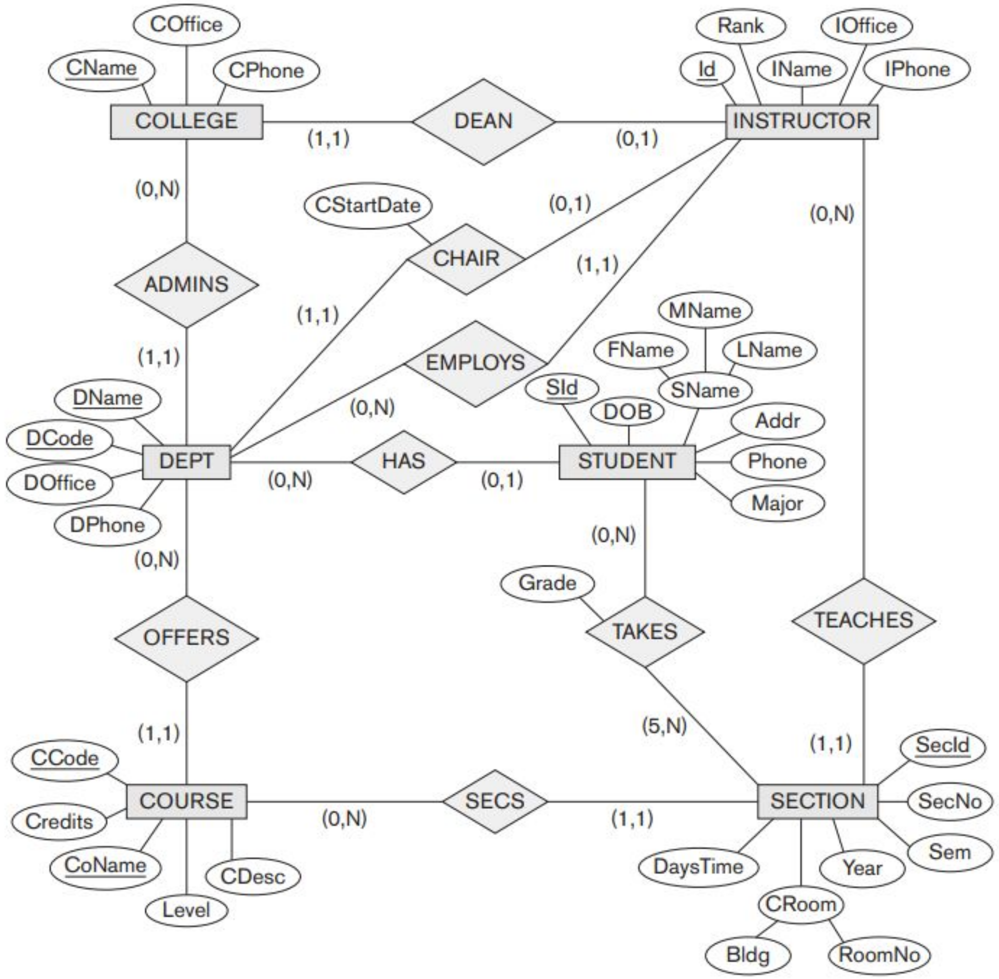


Figure 3.20
An ER diagram for a UNIVERSITY database schema.

Degree of a Relationship type

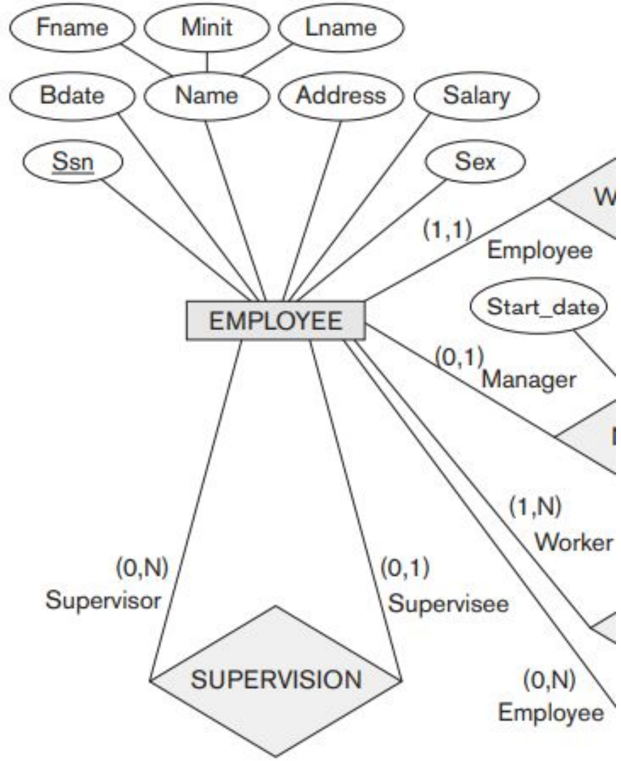
- The number of participating entity types.
- The DEAN relationship is of degree 2. This is also called binary. These are the most common.
- A relationship type of degree three is called ternary.
- You can use the term n-ary for higher degree relationship types.

Relationships as Attributes

- We can think of binary relationships as attributes.
- Take the example of OFFERS relationship type. DEPT OFFERS COURSE.
- A DEPT offers a number of COURSEs.
- There are two points of views to making the relationship an attribute.
- One, DEPT can have a multi-valued attribute called Course for all the courses that can be offered by a particular DEPT entity. Interestingly, this can be null valued for some entities.
- Two, each COURSE entity can have an attribute called Dept representing the DEPT entity.

Role Names and Recursive Relationships

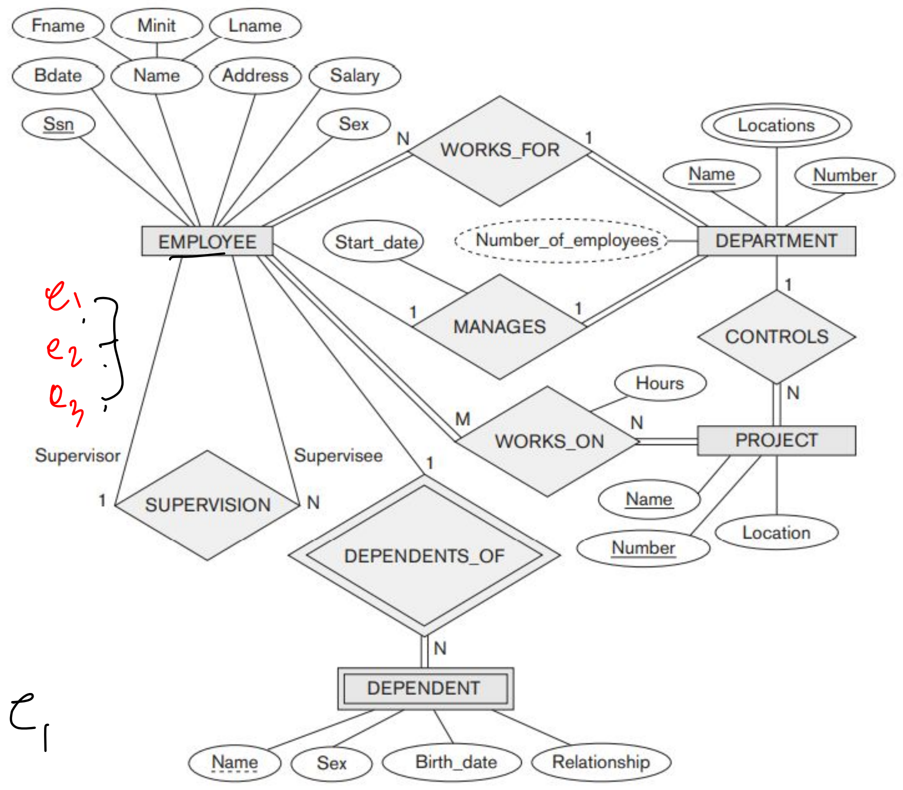
- Role names signify the role played by each entity type in the relationship. They are typically optional, if the entity types participating in a relationship are distinct.
- However, there is a notion of recursive relationships. Take for example, a relationship SUPERVISION that associates EMPLOYEEs with each other. Each relationship instance associates a Supervisor role EMPLOYEE with a Supervisee role EMPLOYEE.

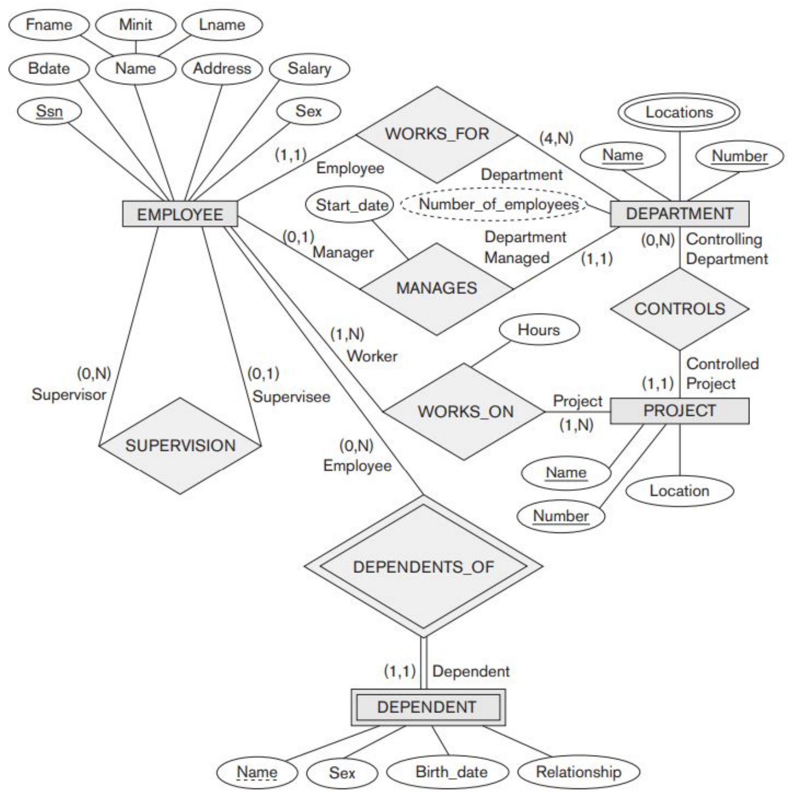


Constraints

- Cardinality Ratios: Already covered in the previous tutorial.
- Participation Constraints: It specifies the minimum number of relationship instances each entity can participate in. It can be either total or partial. Let’s say that every COLLEGE must have an INSTRUCTOR who is the DEAN. Any COLLEGE entity can only exist if it takes part in a DEAN relationship with an INSTRUCTOR. Then we say that the participation of COLLEGE in DEAN is called total participation. In contrast, we do not expect every INSTRUCTOR to be the DEAN of some COLLEGE. The participation of INSTRUCTOR in DEAN is partial participation.
- In simpler terms, no minimum is partial participation and a minimum of one is total participation.
- Total participation is also called existence dependency. It is typically displayed as a double line whereas partial participation is a single line.

From Cardinality, Participation -> (min, max)



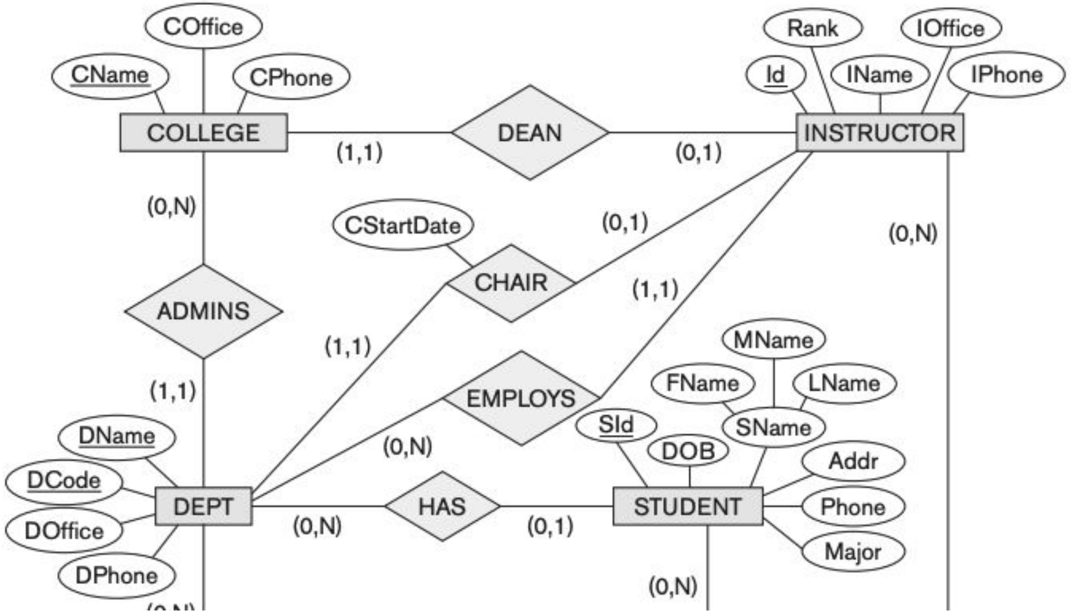


Why use (min, max) notation?

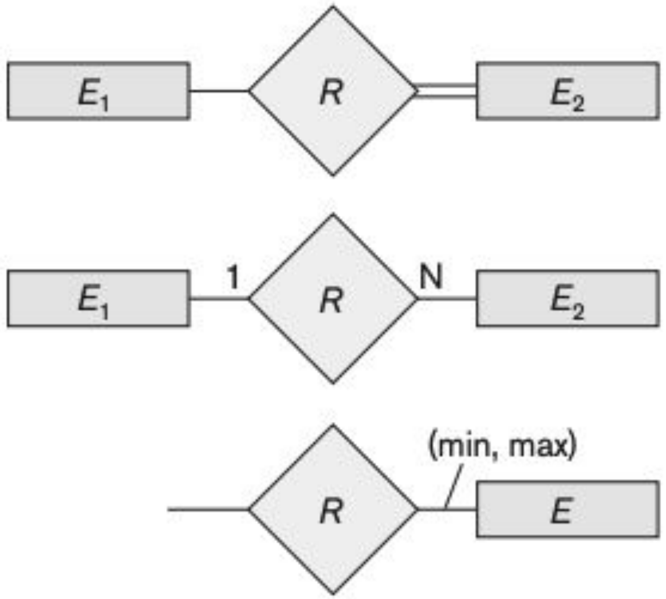
- Clearly the DEAN relationship can be represented as 1:1 using cardinality ratios. What this means is that for every relationship instance between COLLEGE and INSTRUCTOR there is only one COLLEGE and one INSTRUCTOR.
- What this fails to tell us is the participation constraints. We don’t know whether every COLLEGE has a DEAN or whether every INSTRUCTOR is the DEAN of a COLLEGE.
- This is where (min, max) notation is more handy as it encapsulates the ideas of participation constraints and cardinality ratios.
- COLLEGE is (1,1). This means each COLLEGE entity (instance) belongs in exactly one relationship instance. In other words, the participation of COLLEGE is total and only with one INSTRUCTOR.
- INSTRUCTOR is (0, 1). This means each INSTRUCTOR entity (instance) can either not be part of any relationship instance or at most 1. The participation of INSTRUCTOR is partial.

Attributes of Relationship Types

- Just as attributes of an entity type describe the entity type, attributes of a relationship type describe the relationship type
- CStartDate is an attribute of the CHAIR relationship type



Expanded Notation



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

EER Model

- **Enhanced** ER
- Introduces **subclasses** and **superclasses** into the ER model
- Introduces the concepts of **specialisation** and **generalisation**
- Also includes idea of **category** or **union** (out of scope)
- We will only cover specific parts of the EER diagram

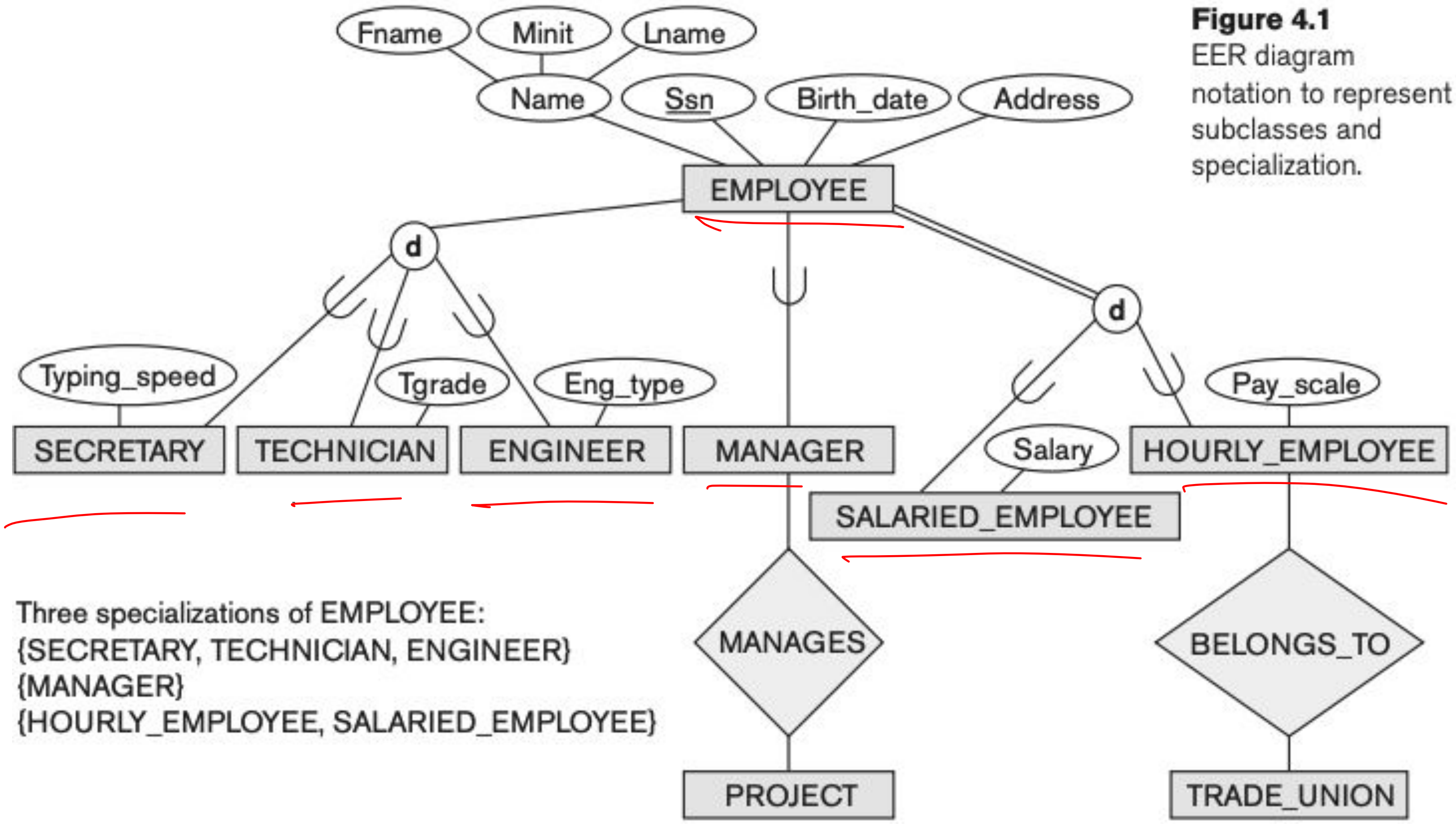
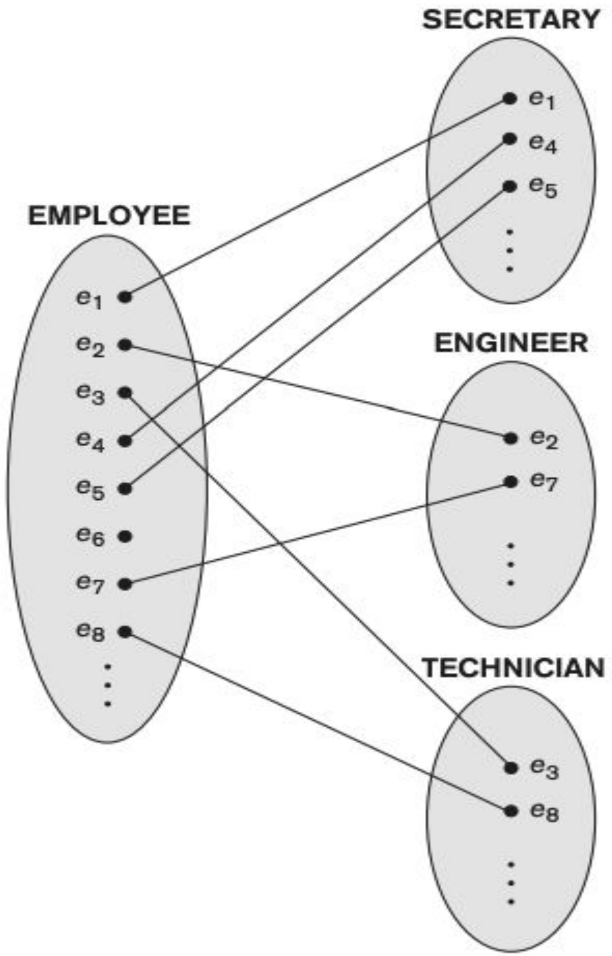


Figure 4.1
EER diagram notation to represent subclasses and specialization.

Subclasses & Superclasses

- Similar to the concept of Inheritance in OOPS
- In an ER diagram, entity types have some distinct attributes but also some shared attributes
- Such entities can be represented separately in the EER model as **subclasses** of an entity (**superclass**) whose attributes are common across all subclasses
- it is not necessary that every entity in a superclass is a member of some subclass



Subclasses & Superclasses

- Formally, a subclass is a **subset of all the entities in a superclass entity type** (recall: venn diagram)
- Relationship between these? Class-subclass (IS-A) relationship
- We say a SECRETARY is an EMPLOYEE, a TECHNICIAN is an EMPLOYEE, and so on.
- An entity in a subclass must also be present in the superclass (not vice-versa), but has a distinct *specific role* that differentiates it
- **Type Inheritance:** An entity that is a member of a subclass inherits all the attributes and relationships of the entity as a member of the superclass
- A subclass also has its own local attributes and relationships

Subclasses & Superclasses

- **Is a subclass the same as an entity type?**

Subclasses & Superclasses

- **Is a subclass the same as an entity type?**
- Yep! In general, a superclass or subclass represents a collection of entities of the same type and hence also describes an entity type.
- that is why superclasses and subclasses are all shown in rectangles in EER diagrams

Specialisation

- Specialisation is the process of defining a set of subclasses of an entity type
- This entity type is called the superclass of the specialization
- Specialisation is defined on the basis of some **distinguishing characteristic** of entities in the superclass
- What are the specialisations and their characteristics in the EER diagram shown?
- **Why include specialisation?**

Specialisation

- Specialisation is the process of defining a set of subclasses of an entity type
- This entity type is called the superclass of the specialization
- Specialisation is defined on the basis of some **distinguishing characteristic** of entities in the superclass
- What are the specialisations and their characteristics in the EER diagram shown?
- **Why include specialisation?**
- Certain attributes may apply to some but not all entities of the superclass entity type
- Similarly, certain relationship types may be participated in only by entities that are members of the subclass

Superclass/Subclass Relationships

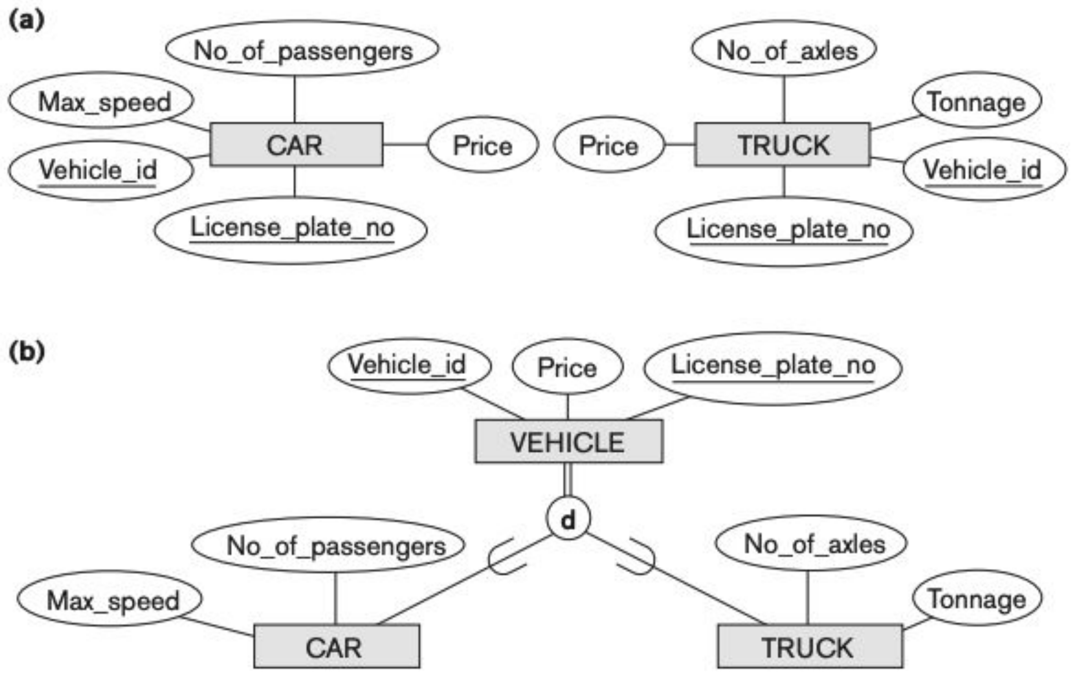
- Is it the same as a 1:1 relationship?

Superclass/Subclass Relationships

- Is it the same as a 1:1 relationship?
- Not Exactly!
- The main difference is that in a 1:1 relationship **two distinct entities** are related, whereas in a superclass/subclass relationship the entity in the subclass is the **same real-world entity** as the entity in the superclass but is playing a **specialized role**
- For example, an EMPLOYEE specialized in the role of SECRETARY, or an EMPLOYEE specialized in the role of TECHNICIAN.

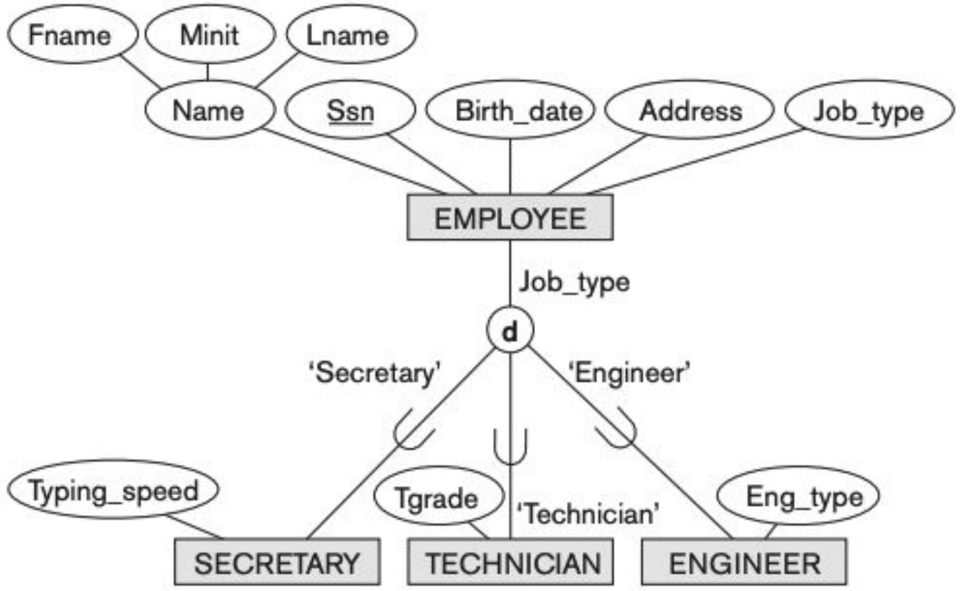
Generalisation

- Specialisation is the process of creating subclasses from one entity type based on their specialised attributes
- Generalisation can be viewed as the inverse of specialisation!



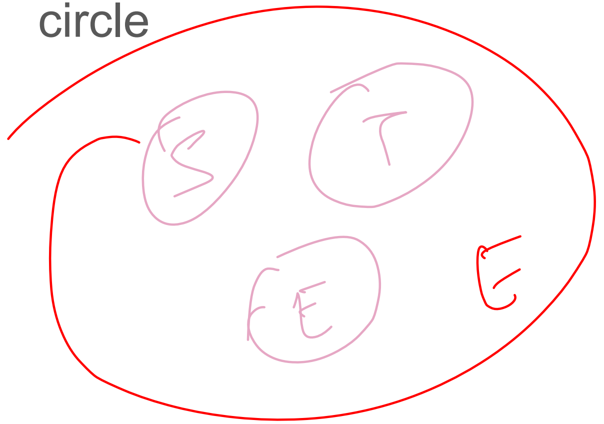
Constraints on Specialisation

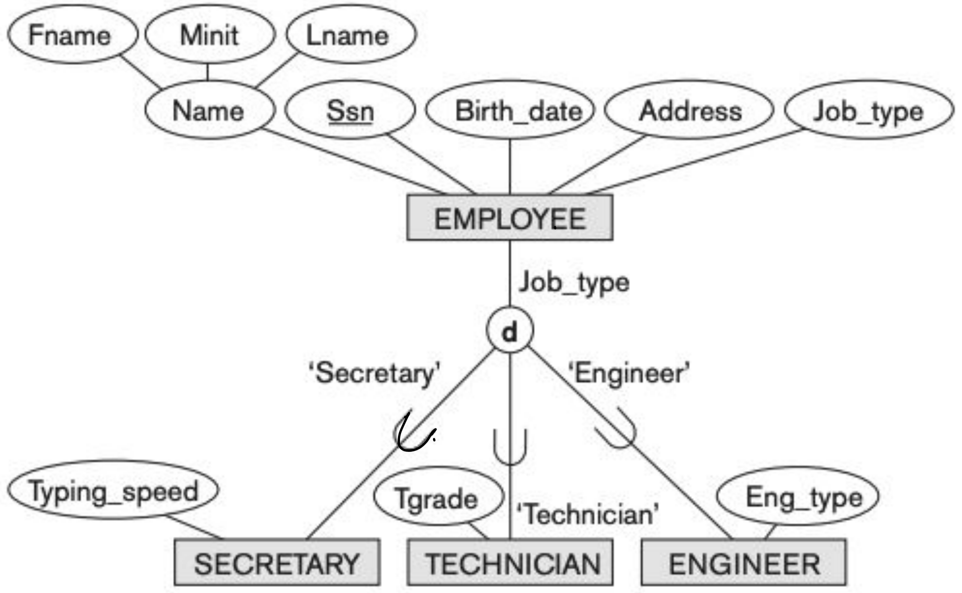
- **Predicate-defined subclasses:** we can specify the condition of membership in the SECRETARY subclass by the condition (Job_type = ‘Secretary’)
- This condition is called the **defining predicate (condition)** of the subclass



Constraints on Specialisation

- **Disjointness constraint:** an entity can be a member of at most one of the subclasses of the specialization
- This is the “d” symbol in the circle

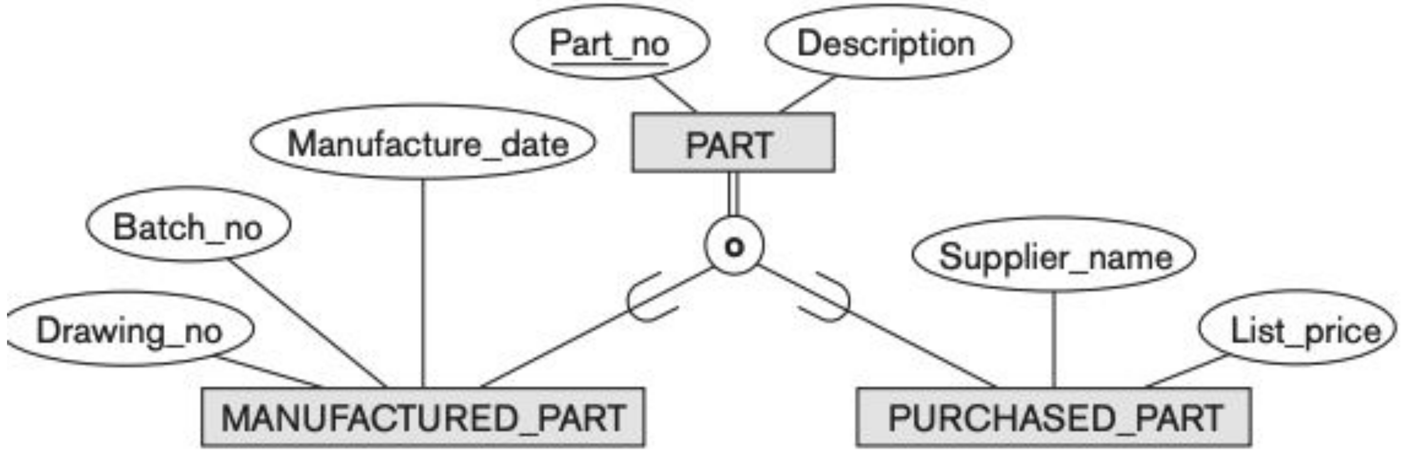




$A \subset B$

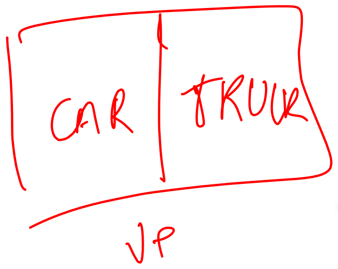
Constraints on Specialisation

- If the subclasses are not constrained to be disjoint, their set of entities might **overlap**

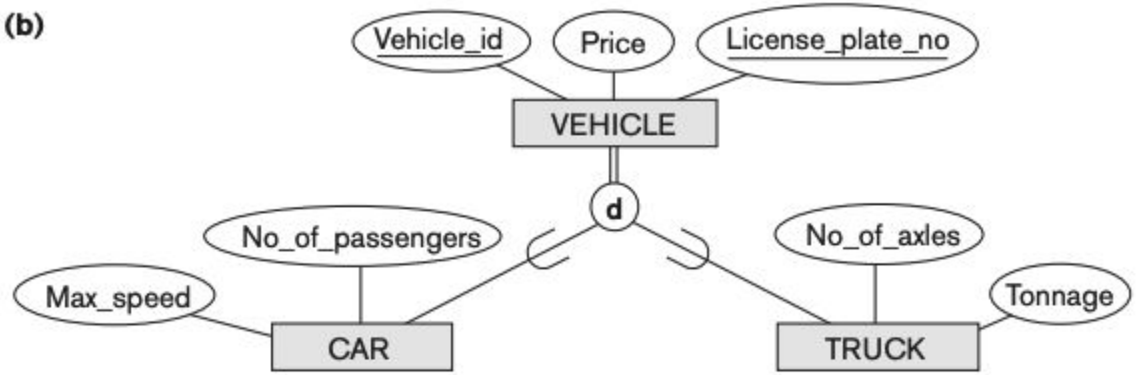


Constraints on Specialisation

- **Completeness constraint**
- A **total specialization** constraint specifies that every entity in the superclass must be a member of at least one subclass in the specialization.
- A **partial specialization** allows an entity in the superclass to not belong to any of the subclasses.



(b)


Vehicle_id
Price
License_plate_no
VEHICLE
d
No_of_passengers
Max_speed
CAR
No_of_axles
Tonnage
TRUCK