

COMP67321
Enhanced ER model

Goran Nenadic

School of Computer Science
University of Manchester

1

Aim

- Review and understand the EER model
- Understand specialisation and generalisation and their role in conceptual modelling
 - analyse the related constraints
- Understand union types
- Understand aggregation

2

Enhanced ER model

- Includes all modeling concepts of the basic ER
- Additional concepts are used to model DBs more completely and more accurately
 - includes some object-oriented concepts (e.g. inheritance)
- Additional concepts:
 - subclasses/superclasses
 - specialisation/generalisation
 - attribute and relationship inheritance
 - UNION type

3

Subclasses and superclasses

- A superclass or subclass represents a collection (or set or grouping) of entities
 - it represents a particular *type of entity*
- Shown in rectangles in EER diagrams (as are entity types)
- We can call all entity types **classes**, whether they are entity types, superclasses, or subclasses

4

continued

Subclasses and superclasses

- An entity type may have additional meaningful sub-groupings of its entities
 - e.g. EMPLOYEE may be further grouped into
 - SECRETARY, ENGINEER, TECHNICIAN, ...
 - Based on the EMPLOYEE's Job
 - MANAGER
 - EMPLOYEEs who are managers
 - each of these subgroupings is a subset of EMPLOYEE entities
 - EMPLOYEE is the superclass for each of these subclasses

5

continued

Subclasses and superclasses

- These additional sub-groupings are called *subclasses* or *subtypes*
- Superclass/subclass relationships
 - EMPLOYEE/SECRETARY, EMPLOYEE/TECHNICIAN, EMPLOYEE/MANAGER, ...
- Also called **IS-A** relationships
 - SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE,

6

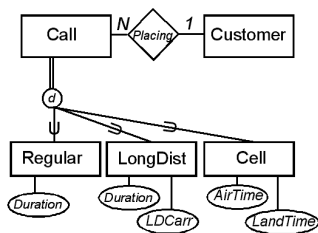
Subclasses and superclasses

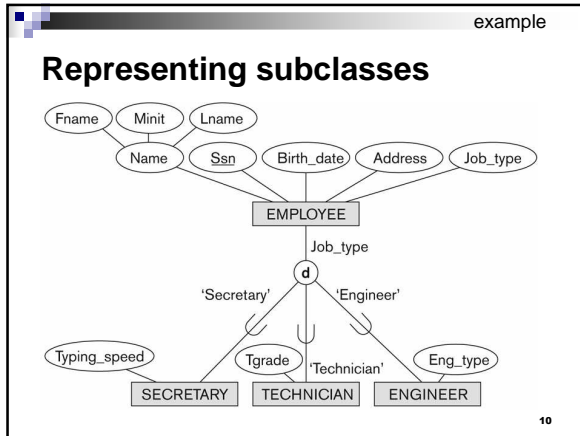
- Note: An entity that is member of a subclass represents the same real-world entity as some member of the superclass:
 - The subclass member is the same entity in a *distinct specific role*
 - An entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the superclass
 - A member of the superclass can be optionally included as a member of any number of its subclasses

Subclasses and superclasses

- Examples:
 - 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 every entity in a superclass be a member of some subclass

Representing subclasses



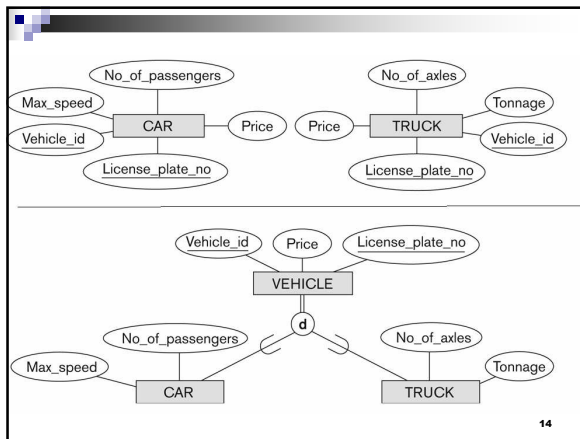


- ### Attribute & relationship inheritance
- An entity that is member of a subclass *inherits*
 - all attributes of the entity as a member of the superclass
 - all relationships of the entity as a member of the superclass
 - Example:
 - SECRETARY (as well as TECHNICIAN and ENGINEER) inherits the attributes (i.e. Name, SSN, ...) from EMPLOYEE
 - every SECRETARY entity will have values for the inherited attributes
- 11

- ### Specialisation & generalisation
- Subclasses and superclasses can help in data modelling by identifying roles, related entities etc.
 - Two approaches to data modelling
 - specialisation: top-down approach
 - generalisation: bottom-up approach
 - Basically, these are inversions of one another
 - Represented in the same way in diagrams
- 12

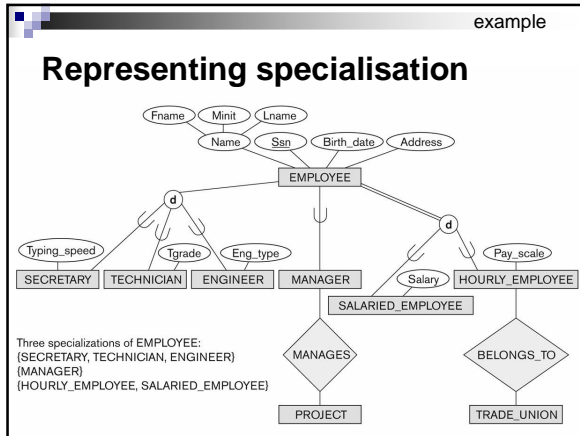
Specialisation & generalisation

- Specialisation: designate subgroupings (subclasses) within an entity set that are distinctive from other entities in the set.
 - these subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set
- Generalisation: combine a number of entity sets that share the same features into a higher-level entity set (superclass)



Specialisation & generalisation

- Note: the same entity type may have different specialisations:
 - e.g. EMPLOYEE based on the type of job (SECRETARY, ENGINEER, etc)
 - e.g. EMPLOYEE based on the pay method (HOURLY, SALARIED)



Membership constraints

- Two basic constraints on whether or not entities may belong to more than one lower-level entity set within a single generalisation.
 - disjointness constraint
 - completeness constraint

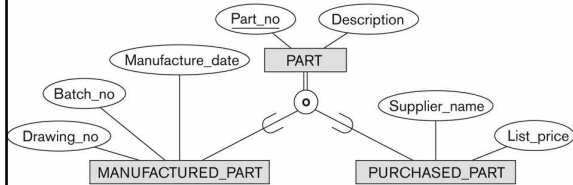
17

Disjointness constraint

- Specifies that the subclasses of the specialisation must be *disjoint*.
 - an entity can be a member of at most one of the subclasses of the given specialization
 - specified by d in EER diagrams
- If not disjoint, specialisation is *overlapping*
 - that is the same entity may be a member of more than one subclass of the specialization
 - specified by o in EER diagrams

18

Example: overlapping subclasses



19

Class membership

- If we can determine exactly those entities that will become members of each subclass by a condition, the subclasses are called predicate-defined (or condition-defined) subclasses
 - condition is a constraint that determines subclass members
 - display a predicate-defined subclass by writing the predicate condition next to the line attaching the subclass to its superclass

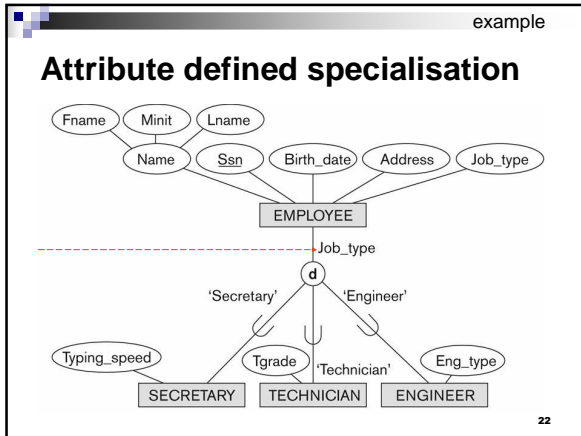
20

continued

Class membership

- If all subclasses in a specialization have membership condition on same attribute of the superclass, specialisation is called an attribute-defined specialisation
 - attribute is called the **defining** attribute
 - e.g. JobType is the defining attribute of the specialization {SECRETARY, TECHNICIAN, ENGINEER} of EMPLOYEE
- If no condition determines membership, the subclass is called user-defined
 - membership is determined by the database users by applying an operation to add an entity to the subclass
 - membership is specified individually for each entity

21



Completeness constraint

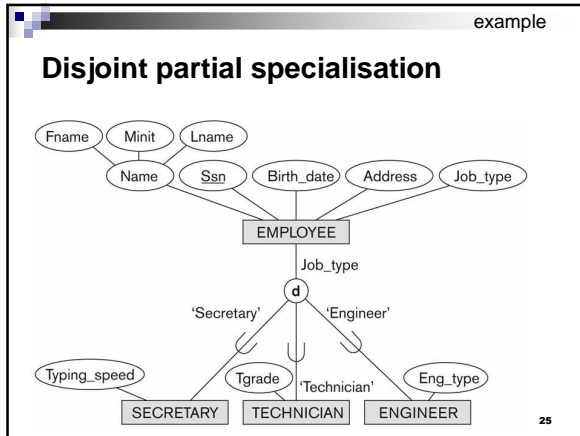
- *Total* (or mandatory) specifies that every entity in the superclass must be a member of some subclass in the given specialization/generalization
 - shown in EER diagrams by a **double line**
- *Partial* (or optional) allows an entity not to belong to any of the subclasses
 - shown in EER diagrams by a single line

23

Disjointness/completeness constraints

- Four types of specialisation/ generalisation:
 - disjoint + total
 - disjoint + partial
 - overlapping + total
 - overlapping + partial
- Note: generalisation is usually total because the superclass is derived from the subclasses.

24



Hierarchies and lattices

- A subclass may itself have further subclasses specified on it, forming a hierarchy or a lattice
- **Hierarchy** has a constraint that every subclass has only one superclass (called **single inheritance**);
 - this is basically a *tree structure*

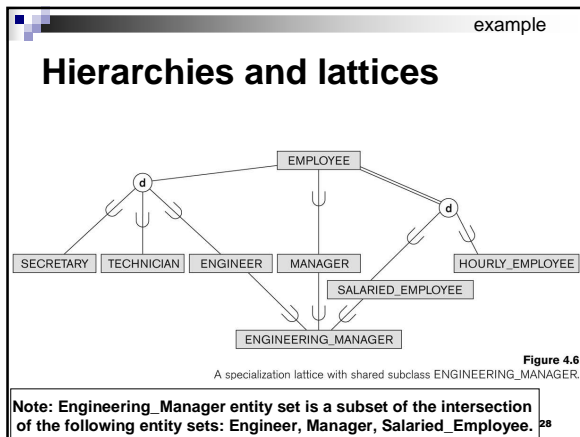
26

continued

Hierarchies and lattices

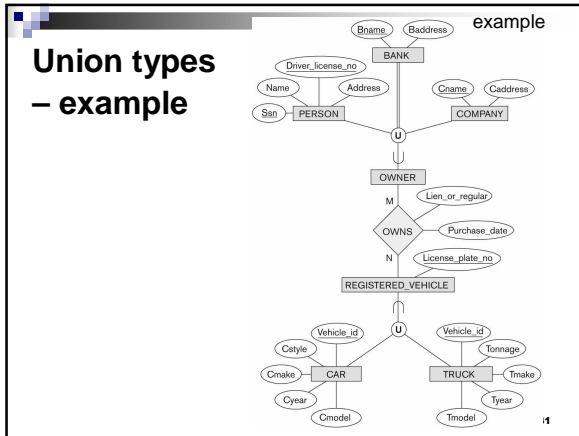
- In a **lattice**, a subclass can be subclass of more than one superclass (called **multiple inheritance**)
 - subclass with more than one superclass is called a shared subclass
- In a lattice or hierarchy, a subclass inherits attributes and relationships not only of its direct superclass, but also of all its predecessor superclasses

27

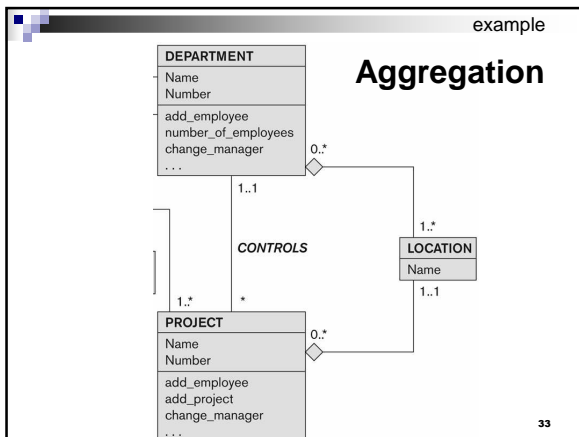


- ## Union types
- All of the *superclass/subclass relationships* we have seen so far have a single superclass
 - even shared classes, e.g. Engineering_Manager
 - In some cases, we need to model a *single superclass/subclass relationship with more than one superclass*
 - superclasses can represent different entity types
 - Such a subclass is called a union type
- 29

- example
- ## 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.
 - type called OWNER is created to represent a subset of the *union* of the three superclasses COMPANY, BANK, and PERSON
 - a union member must exist in **at least one** of its superclasses
 - Difference from *shared subclass*, which is a:
 - subset of the *intersection* of its superclasses
 - shared subclass member must exist in **all** of its superclasses
- 30



- Aggregation**
- Represents a 'has-a' or 'is-part-of' relationship between entity types
 - whole vs. part
 - Examples
 - building has flats (or a flat is part of building)
 - a department has a location;
 - branch 'has' staff
 - UML representation
 - on open diamond next to the 'whole' entity.



example

Aggregation

```

classDiagram
    class ALUMNUS {
        new_alumnus
        ...
    }
    class DEGREE {
        Year
        Degree
        Major
        ...
    }
    ALUMNUS o-- "*" DEGREE
  
```

34

Summary

- EER model concepts:
 - subclass and superclass
- Specialisation and generalisation
 - cardinality and participation
 - class membership and constraints
- Union types
- Aggregations

35

Reading for this lecture

- Main text: Chapter 4 in [Elmasri & Navathe]
- Also, Chapter 12 in [Connolly & Begg]
- See also Chapter 12 in [Elmasri & Navathe]

36

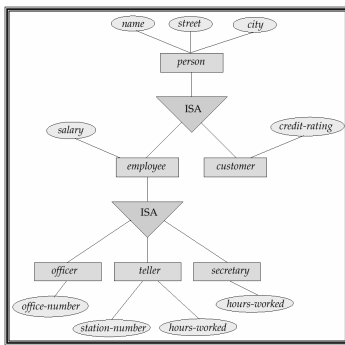
Notations, notations, ...

- Some notations also use arrows to pointing to the generalised superclass or specialised subclasses
 - some authors believe that it is often subjective as to which process is more appropriate for a particular situation
 - [Elmasri and Navathe] does not draw any arrows
 - [Connolly and Begg] frequently does use arrows
 - UML diagrams have arrows

37

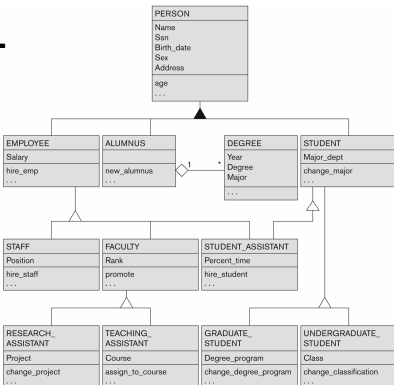
example

Representing specialisation



38

UML



39
