



COSC210 Database Management Systems

Lecture 6 - The Enhanced Entity-Relationship Model

Dr. Mitchell Welch / Dr. Edmund Sadgrove

Reading

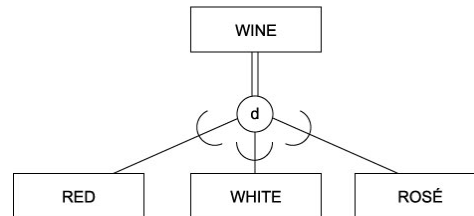
- Chapter 4 from *Fundamentals of Database Systems* by Elmazri and Navathe

Summary

- ER Model vs EER Model
- EER Model - Object Oriented Approach
- Specialisation
- Generalisation
- Constraints of Specialisation and Generalisation
- Hierarchies and Lattices
- Modeling Union Types
- UNIVERSITY Example
- Design Choices

ER Model vs EER Model

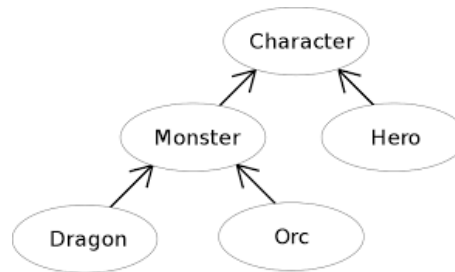
- The Enhanced Entity Relationship (EER) Model:
 - **Includes all** the modelling concepts of the **ER** Model.
 - Introduces the concept of ***object-oriented design***.
 - IN EER Models ***Object*** is used interchangeably with ***Entity***.



ER Model vs EER Model

- EER Models include the **concepts** of:

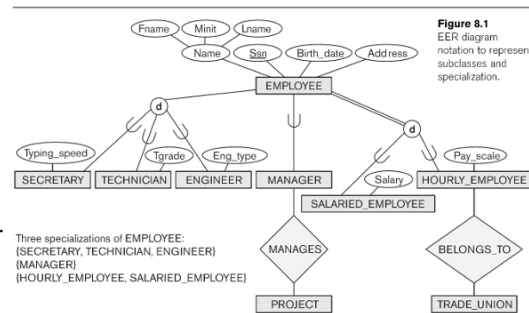
- Superclasses
- Subclasses
- Generalisation
- Specialisation
- Category/Union-type



- These concepts are analogous to the features presented within a typical object-oriented programming language.

EER Model - Object Oriented Approach

- An EER object can be:
 - **Superclass or *Supertype*:**
 - Parent entity type.
 - Exists independently.
 - E.g. employee.
 - **Subclass or *Subtype*:**
 - A **sub-grouping** within the **entity type**.
 - Requires a superclass to exist.
 - E.g. supervisor, engineer, administration.
- Often called an *is-a* relationship.



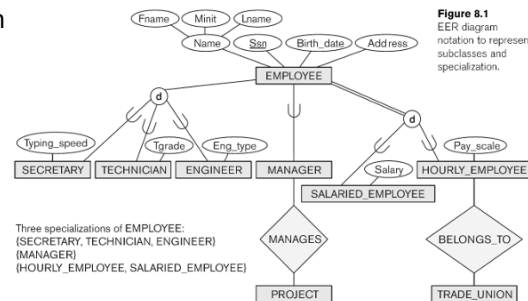
³A class/subclass relationship is often called an **IS-A** (or **IS-AN**) relationship because of the way we refer to the concept. We say a SECRETARY is an EMPLOYEE, a TECHNICIAN is an EMPLOYEE, and so on.

EER Model - Object Oriented Approach

- The EER model also uses **type inheritance**.
- **Recall** that an **entity type** is defined by the **value set**.
 - The powerset of all possible values (attribute domain).
- In the EER model:
 - A **subclass inherits** the domain of their superclasses attributes.
 - The subclass will also have attributes **unique** to the subclass type.

Specialisation

- **Specialisation:**
 - Defines a set of subclasses based on common characteristics.
 - Represented by a **grouping circle** with adjoining lines.
 - The *subset* symbol represents direction (arch).
 - Example:
 - EMPLOYEE has two specialisations:
 - Job type.
 - Method of Pay.
 - Unique attributes are called **specific attributes**.



³A class/subclass relationship is often called an **IS-A** (or **IS-AN**) relationship because of the way we refer to the concept. We say a SECRETARY is an EMPLOYEE, a TECHNICIAN is an EMPLOYEE, and so on.

Specialisation

- Subclass specification:
 - May share the majority of their **attributes**.
 - May only have a small number of **specific attributes**.
 - May participate in different relationships to the super/subclasses.
- Specialisation represents related but distinct object types.

Generalisation

- **Generalisation:**
 - The reverse process of **specialisation**.
 - Common features are consolidated into a **superclass**.
 - Uncommon attributes are included as **specific attributes**.
 - Example:
 - CAR and TRUCK can be generalised into VEHICLE.
 - With common attributes placed in the superclass.

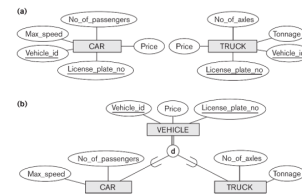
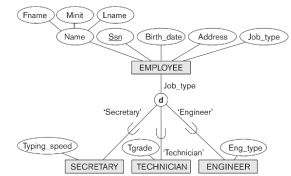


Figure 8.3
Generalization. (a) Two entity types, CAR and TRUCK. (b)
Generalizing CAR and TRUCK into the superclass VEHICLE.

Constraints on Specialisation and Generalisation

- We can have **condition-defined** subclasses:
 - Membership is dependent on a superclass attribute value.
 - Example:
 - Adding attribute *job_type* to our EMPLOYEE supertype.
 - In **specialisation**:
 - All subclasses can have conditional membership:
 - This is called **attribute-defined**.
 - Implies disjointed (next slide).
 - With no conditional membership:
 - This is called **user-defined**.

Figure 8.4
EER diagram notation
for an attribute-defined
specialization on
job_type.



¹⁰Such an attribute is called a discriminator in UML terminology.

Constraints on Specialisation and Generalisation

- The EER model has inheritance constraints:
- The **disjoint constraint (d)**:
 - Specifies an entity can be a member of one subclass (only).
- The **overlapping constraint (o)**:
 - Specifies an entity can be a member of multiple subclasses.
- The **total specialisation constraint**:
 - Must be a member (double-line).
- The **partial specialisation constraint**:
 - May be a member (single-line).

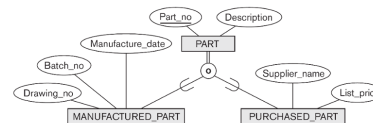


Figure 8.5
EER diagram notation
for an overlapping
(nondisjoint)
specialization.

[†]The notation of using single or double lines is similar to that for partial or total participation of an entity type in a relationship type, as described in Chapter 7.

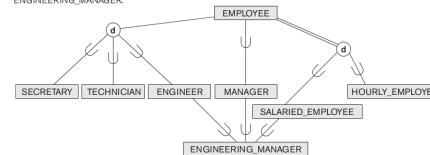
Constraints on Specialisation and Generalisation

- There are four possible **combinations**:

- Disjoint, total
- Disjoint, partial
- Overlapping, total
- Overlapping partial

- The combinations used will be determined by the application.
- Typically superclasses will have **total participation** as the superclass is generalised from the subclasses.

Figure 8.6
A specialization lattice with shared subclass
ENGINEERING_MANAGER.



Constraints on Specialisation and Generalisation

- Rules apply when manipulating entities in a generalisation/specialisation relationship (think aggregation):
 - **Deleting:**
 - Deleting a **superclass** entity deletes from all **subclasses**.
 - **Inserting :**
 - Inserting into a **superclass** means that it is inserted into *predicate-defined* subclasses (matching attributes).
- **Inserting** into a **superclass** with **total specialisation** implies that it is inserted into at least one **subclass**.

Hierarchies and Lattices

- Class relationships can be in the form of *hierarchies* and *lattice* structures.

- **Hierarchy:**

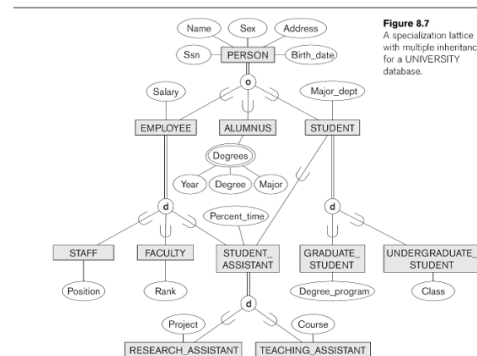
- Implies just one relationship connection.

- **Lattice:**

- Allows more than one relationship connection.

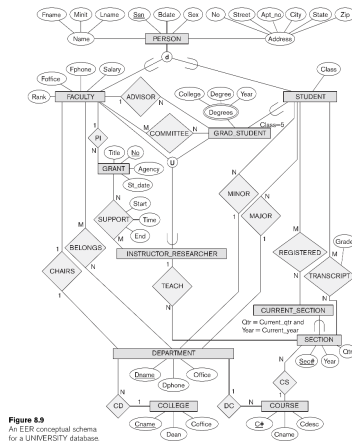
- In both cases a subclass **inherits** the attributes from all predecessors superclasses .

- A **leaf node** is a class that has no subclasses.



Hierarchies and Lattices

- In a **lattice** structure:
 - Possible for a subclass to **inherit** the same attributes twice.
 - Example:
 - STUDENT_ASSISTANT entity will inherit the PERSON attributes twice.
 - In this situation the attributes are only inherited once.



Hierarchies and Lattices

- Lattices and Hierarchies can be generated through: a top-down or bottom-up refinement process.
 - **Top-down** approach:
 - Generalise the superclasses and specialise down to the subclasses.
 - **Bottom-up** approach:
 - Specialise as subclasses and generalise up into common attribute superclasses.
- These processes can create identical arrangements of superclasses and subclasses.

Modeling Union Types

- So far all relationship examples have had a single **superclass**.

- It may be necessary to use multiple **superclasses**:

- This means a superclass will represent a subset of the subclass entity.
- This is called a **union** relationship (**U**).
- Inherits a set of attributes depending on the subset.
- Example:
 - Person, bank or company can be a vehicle owner.

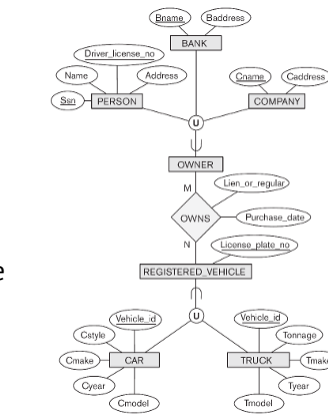
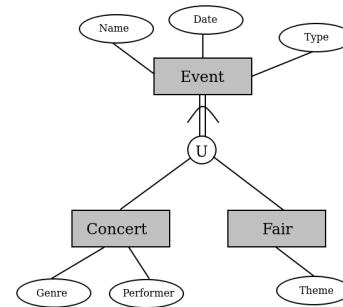


Figure 8.8
Two categories (union types): OWNER and REGISTERED_VEHICLE.

Modeling Union Types

- **Union** participation can be *total* or *partial*.
 - **Total participation:**
 - A subclass holds the **union** of all entities in its superclass
 - **Partial Participation:**
 - A subclass can hold a **subset** of the superclasses.
 - **Total** participation is denoted using the **double line** between the subclass and the grouping circle.



Design Choices

- **Conceptual database design** is an iterative process of *refinement*.
 - Some guidelines include:
 - Only represent **subclasses** when necessary.
 - Subclasses can be **merged** into a superclass.
 - If no specific relationship or few local attributes.
 - Avoid using **union types**:
 - Can be complex and difficult to implement.
 - Will become separate relations.
 - The choice of **constraints** (e.g. disjoint, overlapping and total, partial) is driven by the mini-world. The **default** will usually be **overlapping-partial**.

Customer Banking Example

- Example banking EERD:
 - Entities:
 - Bank branch.
 - Customer.
 - Customer account.
 - Student.
 - Investor.
 - Standard.
 - Customer loan.
 - Car.
 - House.

Summary

- ER Model vs EER Model
- EER Model - Object Oriented Approach
- Specialisation
- Generalisation
- Constraints of Specialisation and Generalisation
- Hierarchies and Lattices
- Modeling Union Types
- UNIVERSITY Example
- Design Choices

Questions?

Next Lecture

- Generating an SQL schema from a ER/EER model..

Reading

- Chapters 4 from *Fundamentals of Database Systems*
- Chapter 7 from *Fundamentals of Database Systems* for Fridays prac session.

