Enhanced Entity-Relationship (EER) Model

Part 3

Introduction

- It is very harder to apply the conventional ER paradigm for database modeling as data complexity rises today.
- The existing ER model needs to be enhanced or improved in order for it to better handle the complicated application in order to reduce the modeling complexity.
- The requirements and complexity of complicated databases are represented using enhanced entity-relationship diagrams, which are sophisticated database diagrams very similar to standard ER diagrams.

Enhanced ER Model

- Enhanced entity-relationship diagrams are advanced database diagrams very similar to regular ER diagrams which represent the requirements and complexities of complex databases.
- ER model was introduced for modeling most common business problems and has widespread use.
- The term enhanced entity relationship model is used to identify the result for extending the original ER model with the new modeling constructs.
- It is a diagrammatic technique for displaying the Sub Class and Super Class; Specialization and Generalization; Union or Category; Aggregation etc.

Super types and Sub types:

•Supertype - an entity type that relates to one or more subtypes.

•Subtype - a subgroup of entities with unique attributes.

•Sub class and Super class relationship leads the concept of Inheritance.

•Inheritance - the concept that subtype entities inherit the values of all supertype attributes.

Note: subtype instances are also classified as supertype instances.

1. Super Class

- Super class is an entity type that has a relationship with one or more subtypes.
- An entity cannot exist in database merely by being member of any super class.

For example: Shape super class is having sub groups as Square, Circle, Triangle.

2. Sub Class

- Sub class is a group of entities with unique attributes.
- Sub class inherits properties and attributes from its super class.

For example: Square, Circle, Triangle are the sub class of Shape super class.

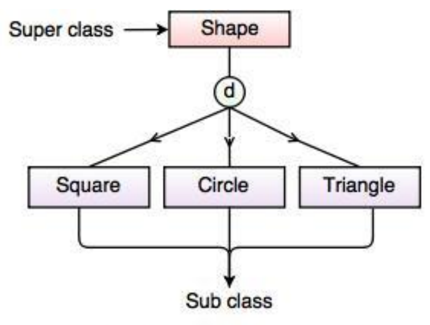
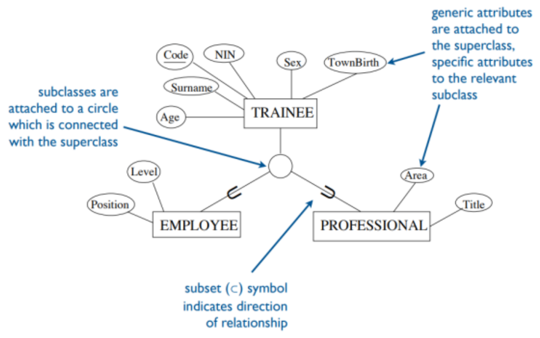


Fig. Super class/Sub class Relationship

EER notation


Code
NIN
Sex
TownBirth
Surname
Age
TRAINEE
generic attributes are attached to the superclass, specific attributes to the relevant subclass
subclasses are attached to a circle which is connected with the superclass
Level
Position
EMPLOYEE
Area
Title
PROFESSIONAL
subset (⊂) symbol indicates direction of relationship

Constraints –

There are two types of constraints on the "Sub-class" relationship.

1.Total or Partial –

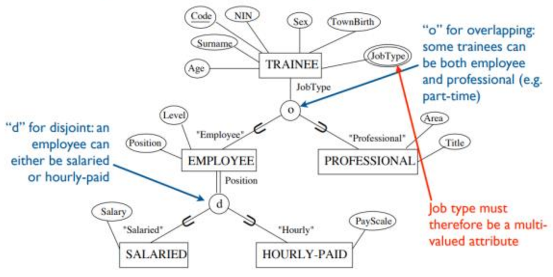
- A sub-classing relationship is total if every super-class entity is to be associated with some sub-class entity, otherwise partial.
- Sub-class "job type based employee category" is **partial** sub-classing – not necessary every employee is one of (secretary, engineer, and technician),
  i.e. union of these three types is a proper subset of all employees.
- Whereas other sub-classing "Salaried Employee AND Hourly Employee" is **total**; the union of entities from sub-classes is equal to the total employee set,
  i.e. every employee necessarily has to be one of them.

Constraints –

2. Overlapped or Disjoint –

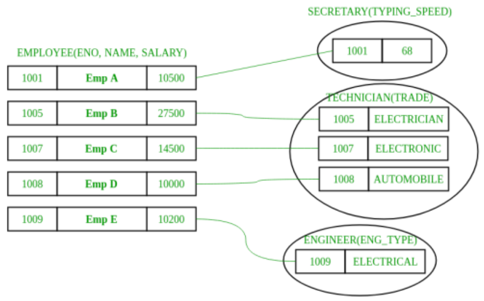
- If an entity from a super-set can be related (can occur) in multiple sub-class sets, then it is overlapped sub-classing, otherwise disjoint.
- The disjointness between sub class and super class is denoted with ⓓ symbol, where the **d** in the circle stands for *disjoint*. The **d** notation also applies to user-defined subclasses of a specialization that must be disjoint, as illustrated by the specialization {HOURLY_EMPLOYEE, SALARIED_EMPLOYEE} in Figure 4.1.
- Both the examples: job-type based and salaries/hourly employee sub-classing are disjoint.

disjoint/overlap notation

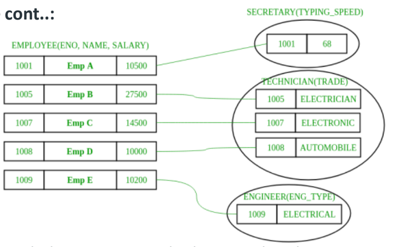

Code
NIN
Sex
TownBirth
Surname
Age
TRAINEE
JobType
JobType
o
"o" for overlapping: some trainees can be both employee and professional (e.g. part-time)
Level
"Employee"
"Professional"
Position
Area
Title
EMPLOYEE
PROFESSIONAL
"d" for disjoint: an employee can either be salaried or hourly-paid
Position
d
Salary
"Salaried"
"Hourly"
PayScale
SALARIED
HOURLY-PAID
Job type must therefore be a multi-valued attribute

Example:

This example instance of **"sub-class"** relationships. Here we have three sets of employees: Secretary, Technician, and Engineer. The employee is a super-class of the rest three sets of individual sub-class is a subset of Employee set.



Example cont..:



- An entity belonging to a sub-class is related to some super-class entity. For instance emp, no 1001 is a secretary, and his typing speed is 68. Emp no 1009 is an engineer (sub-class) and her Eng_Type is "Electrical", so forth.
- Sub-class entity "inherits" all attributes of super-class; for example, employee 1001 will have attributes eno, name, salary, and typing speed.

For example,
The entities that are members of the EMPLOYEE entity type may be distinguished further into SECRETARY, ENGINEER, MANAGER, TECHNICIAN, SALARIED_EMPLOYEE, HOURLY_EMPLOYEE, and so on. The set or collection of entities in each of the latter groupings is a subset of the entities that belong to the EMPLOYEE entity set, meaning that every entity that is a member of one of these subgroupings is also an employee. We call each of these subgroupings a **subclass** or **subtype** of the EMPLOYEE entity type, and the EMPLOYEE entity type is called the **superclass** or **supertype** for each of these subclasses,
We call the relationship between a superclass and any one of its subclasses a **superclass/subclass** or **supertype/subtype** or simply **class/subclass relationship.**

Figure 4.1 shows how we represent a specialization diagrammatically in an EER diagram.

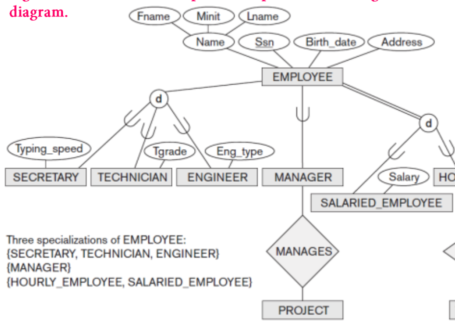


Figure 4.1
EER diagram notation to represent subclasses and specialization.

The subclasses that define a specialization are attached by lines to a circle that represents the specialization, which is connected in turn to the superclass. The <u>*subset symbol* on each line connecting a subclass to the circle indicates the direction of the superclass/subclass relationship</u>.

- The set of subclasses that forms a **specialization is defined on the basis of some distinguishing characteristic of the entities in the superclass**.
- For example, the set of <u>subclasses {SECRETARY, ENGINEER, TECHNICIAN}</u> is a specialization of the superclass EMPLOYEE that distinguishes among employee entities **based on the *job type*** of each employee.
- **We may have several specializations of the same entity type** <u>based on different distinguishing characteristics</u>.
- For example, another specialization of the EMPLOYEE entity type may yield the set of <u>subclasses {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE}; this specialization distinguishes among employees</u> **based on the *method of pay*.**

<u>**There are two main reasons for including class/subclass relationships and specializations.**</u>

<u>The first reson</u> is that certain attributes may apply to some but not all entities of the superclass entity type. A subclass is defined in order to group the entities to which these attributes apply. The members of the subclass may still share the majority of their attributes with the other members of the superclass.
For example,
in Figure 4.1 the SECRETARY subclass has the specific attribute Typing_speed, whereas the ENGINEER subclass has the specific attribute Eng_type, but SECRETARY and ENGINEER share their other inherited attributes from the EMPLOYEE entity type.

<u>The second reason</u> for using subclasses is that <u>some relationship types may be participated in only by entities</u> that are members of the subclass.

For example, if only HOURLY_EMPLOYEES can belong to a trade union, we can represent that fact by creating the subclass HOURLY_EMPLOYEE of EMPLOYEE and relating the subclass to an entity type TRADE_UNION via the BELONGS_TO relationship type, as illustrated in Figure 4.1.

Specialization and Generalization

1. Generalization

- Generalization is the process of generalizing the entities which contain the properties of all the generalized entities.
- It is a <u>bottom up approach</u>, in which <u>two lower level entities combine to form a higher level entity.</u>
- Generalization is <u>the reverse process of Specialization</u>.
- It defines a general entity type from a set of specialized entity type.
- It minimizes the difference between the entities by identifying the common features.

Specialization and Generalization

1.Generalization

For example:

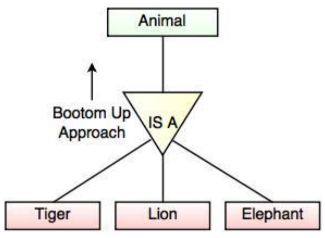


Fig. Generalization

In the above example, Tiger, Lion, Elephant can all be generalized as Animals.

Specialization and Generalization

2. Specialization

- Specialization is a process that defines a group entities which is divided into sub groups based on their characteristic.
- It is a top down approach, in which one higher entity can be broken down into two lower level entity.
- It maximizes the difference between the members of an entity by identifying the unique characteristic or attributes of each member.
- It defines one or more sub class for the super class and also forms the superclass/subclass relationship.

Specialization and Generalization

2. Specialization

For example:

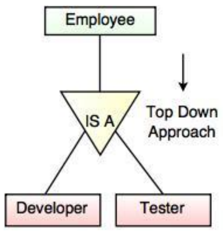
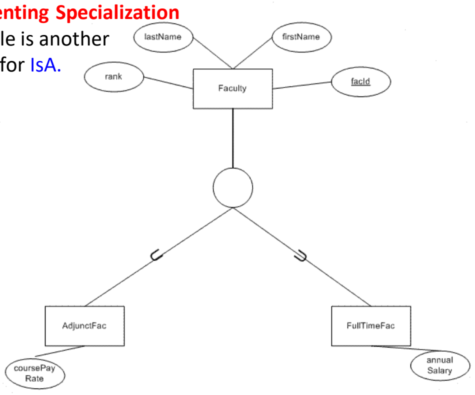


Fig. Specialization

In the above example, Employee can be specialized as Developer or Tester, based on what role they play in an Organization.

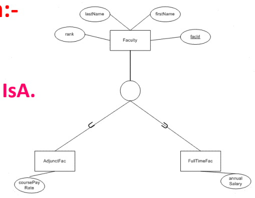
Representing Specialization

The circle is another symbol for IsA.



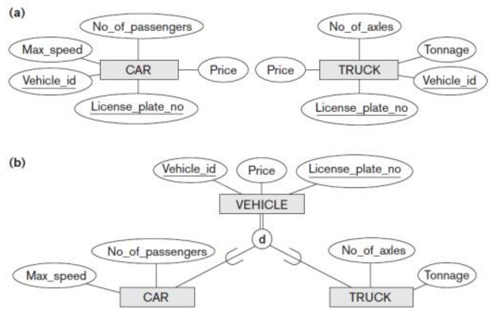
Representing Specialization:-

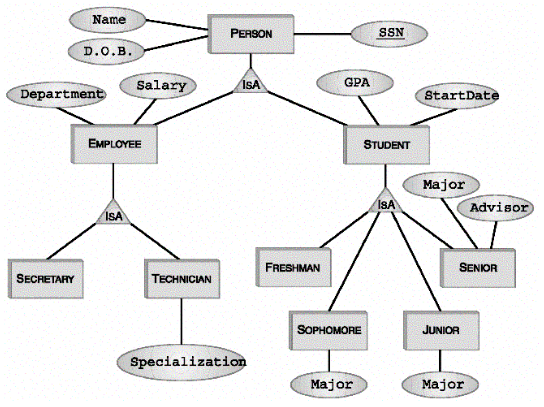
The circle is another symbol for **IsA.**



- E-ER diagram – shows specialization circle (IsA relationship), and inheritance symbol (subset symbol).
- Specialization can also involve just one subclass – no need for circle, but show inheritance symbol.
- The sub-entities are most likely invoking the disjointedness constraint.

Figure 4.3
Generalization. (a) Two entity types, CAR and TRUCK.
(b) Generalizing CAR and TRUCK into the superclass VEHICLE.





Constraints and Characteristics of Specialization and Generalization

For example,
If the EMPLOYEE entity type has an attribute Job_type, as shown in Figure 4.4, we can specify the condition of membership in the SECRETARY subclass by the condition (Job_type = 'Secretary'), which we call the **defining predicate** of the subclass.
This condition is a *constraint* specifying that exactly those entities of the EMPLOYEE entity type whose attribute value for Job_type is 'Secretary' belong to the subclass. We display a predicate-defined subclass by writing the predicate condition next to the line that connects the subclass to the specialization circle.

Constraints and Characteristics of Specialization and Generalization

- If all subclasses in a specialization have membership condition on same attribute of the superclass, specialization is called an attribute-defined specialization
  - Attribute is called the defining attribute of the specialization
  - Example: JobType is the defining attribute of the specialization {SECRETARY, TECHNICIAN, ENGINEER} of EMPLOYEE
  - Display an attribute-defined specialization by placing the defining attribute name next to the arc from the circle to the superclass, as shown in Figure 4.4.
- If no condition determines membership, the subclass is called user-defined
  - Membership in a subclass is determined by the database users by applying an operation to add an entity to the subclass
  - Membership in the subclass is specified individually for each entity in the superclass by the user

Displaying an attribute-defined specialization in EER diagrams

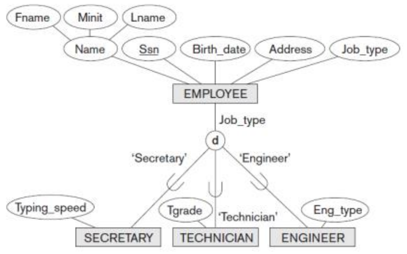
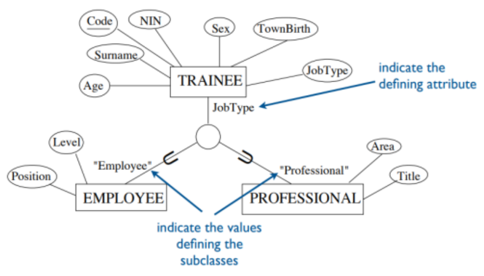
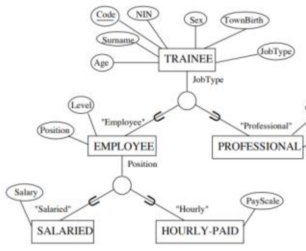


Figure 4.4
EER diagram notation for an attribute-defined specialization on Job_type.

Defining attribute notation



- we can have further specialisations of the same entity type using different distinguishing characteristics e.g. EMPLOYEE can be specialised on the basis of Position into SALARIED and HOURLY-PAID



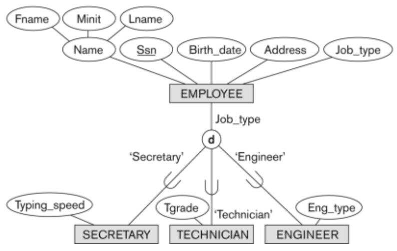
- Note that SALARIED and HOURLY-PAID are both EMPLOYEE and TRAINEE

Constraints and Characteristics of Specialization and Generalization

Where the **d in the circle** stands for ***disjoint***. The **d** notation also applies to user-defined subclasses of a specialization that must be
disjoint, as illustrated by the specialization {HOURLY_EMPLOYEE, SALARIED_EMPLOYEE} in Figure 4.1.
If the subclasses are not constrained to be disjoint, their sets of entities may be ***overlapping***; that is, the same (real-world) entity may be a member of more than one subclass of the specialization. This case, which is the default, is displayed by placing an **o in the circle**, as shown in Figure 4.5.

Example of disjoint partial Specialization

Figure 4.4
EER diagram notation for an attribute-defined specialization on Job_type.



Slide 4- 34

Example of overlapping total Specialization

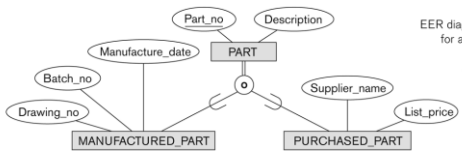


Figure 4.5
EER diagram notation for an overlapping (nondisjoint) specialization.

Slide 4- 35

Constraints and Characteristics of Specialization and Generalization

The second constraint on specialization is called the <u>**completeness (or totalness) constraint**</u>, which may be total or partial.

A **total specialization** constraint specifies that *every* entity in the superclass must be a member of at least one subclass in the specialization.

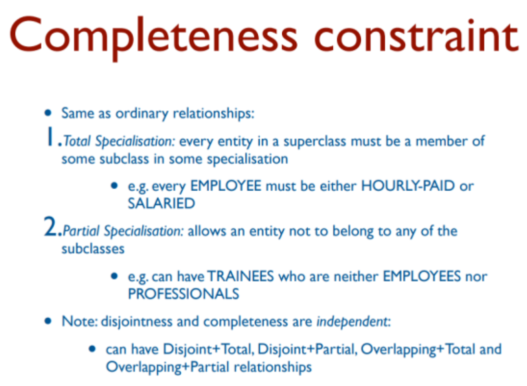
For example, if every EMPLOYEE must be either an HOURLY_EMPLOYEE or a SALARIED_EMPLOYEE, then the specialization {HOURLY_EMPLOYEE, SALARIED_EMPLOYEE} in Figure 4.1 is a <u>**total specialization**</u> of EMPLOYEE.

This is shown in EER diagrams by using a **double line to connect the superclass to the circle.** A single line is used to display a **partial specialization**, which allows an entity not to belong to any of the subclasses.

Constraints and Characteristics of Specialization and Generalization

A single line is used to display a **partial specialization**, which allows an entity not to belong to any of the subclasses.

For example, if some EMPLOYEE entities do not belong to any of the subclasses {SECRETARY, ENGINEER, TECHNICIAN} in Figures 4.1 and 4.4, then that specialization is partial.



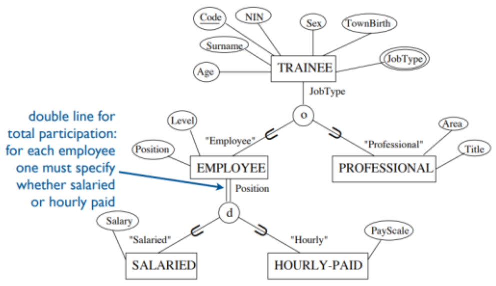
Completeness constraint

- Same as ordinary relationships:

1. *Total Specialisation:* every entity in a superclass must be a member of some subclass in some specialisation
   - e.g. every EMPLOYEE must be either HOURLY-PAID or SALARIED
2. *Partial Specialisation:* allows an entity not to belong to any of the subclasses
   - e.g. can have TRAINEES who are neither EMPLOYEES nor PROFESSIONALS

- Note: disjointness and completeness are *independent:*
  - can have Disjoint+Total, Disjoint+Partial, Overlapping+Total and Overlapping+Partial relationships

total/partial participation



Constraints and Characteristics of Specialization and Generalization

Certain <u>insertion and deletion rules apply to specialization</u> (and generalization) as a consequence of the constraints specified earlier. Some of these rules are as follows:

- <u>Deleting an entity from a superclass implies that it is automatically deleted from all the subclasses</u> to which it belongs.
- <u>Inserting an entity in a superclass implies</u> that <u>the entity is mandatorily inserted in all *predicate-defined* (or *attribute-defined*) subclasses</u> for which the entity satisfies the defining predicate.
- Inserting an entity in a superclass of a *total specialization* implies that the entity is mandatorily inserted in at least one of the subclasses of the specialization.

Insertion & Deletion Rules

1. deleting an entity from a super class implies that it is automatically deleted from all of the subclasses it belongs to
2. inserting an entity in a superclass implies that the entity is inserted in all predicate-defined subclasses for which the entity satisfies the defining predicate
3. inserting an entity in a superclass of total specialisation implies that the entity is inserted in at least one of the subclasses of the specialisation
4. inserting an entity in a super class of disjoint, total specialisation implies that the entity is inserted in one and only one of the subclasses of the specialisation

Specialization/Generalization Hierarchies, Lattices & Shared Subclasses

- A **subclass may itself have further subclasses** specified on it
  - **forms 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***
- In a ***lattice***, a subclass can be subclass of more than one superclass (called ***multiple inheritance***)

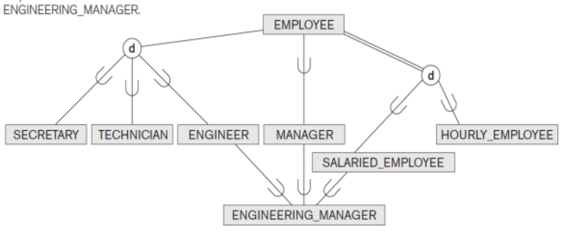
Specialization and Generalization Hierarchies and Lattices

- A subclass itself may have further subclasses specified on it, forming a <u>hierarchy or a lattice</u> of specializations.
- For example, in Figure 4.6 ENGINEER is a subclass of EMPLOYEE and is also a superclass of ENGINEERING_MANAGER; this represents the real-world constraint that every engineering manager is required to be an engineer.
- A **specialization hierarchy** has the constraint that every subclass participates *as a subclass* in *only one* class/subclass relationship; that is, each subclass has only one parent, which results in a **tree structure** or **strict hierarchy**.
- In contrast, for a <u>**specialization lattice**, a subclass can be a subclass in *more than one* class/subclass</u> relationship. Hence, Figure 4.6 is a lattice.

Specialization and Generalization Hierarchies and Lattices

<u>Shared Subclass "Engineering_Manager"</u>

Figure 4.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.



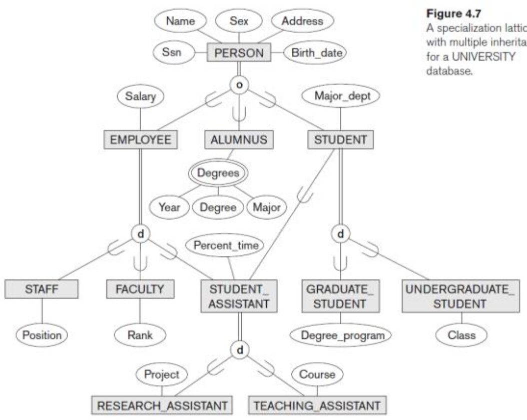


Figure 4.7
A specialization lattice with multiple inheritance for a UNIVERSITY database.

Category or Union

- Category represents a single super class or sub class relationship with more than one super class.
- It can be a total or partial participation.

For example

Car booking, Car owner can be a person, a bank (holds a possession on a Car) or a company.

Category (sub class) → Owner is a subset of the union of the three super classes → Company, Bank, and Person. A Category member must exist in at least one of its super classes.

Category or Union

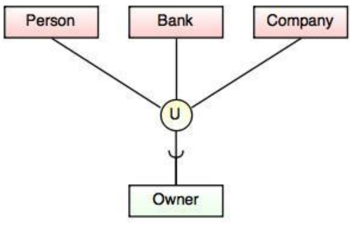


Fig. Categories (Union Type)

Category or Union

- Subclass related to a collection of super-classes.
- Each instance of subclass belongs to one, not all, of the super-classes.
- Super-classes form a union or category.
- Ex. A Sponsor may be a team, a department, or a club.
- Each Sponsor entity instance is a member of one of these super-classes, so Sponsor is a subclass of the union of Team, Dept, Club
- EER diagram - connect each superclass to union circle, connect circle to subclass, with subset symbol on line between circle and subclass.

Category or Union

Example

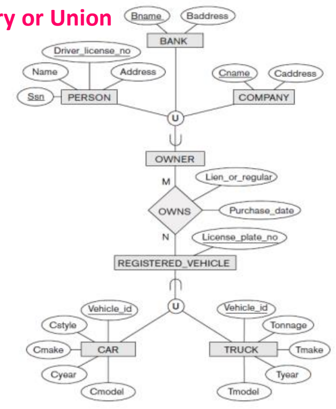
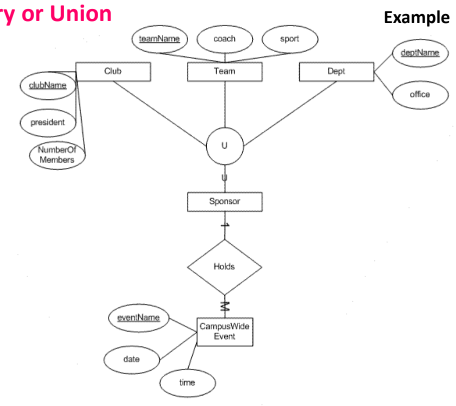


Figure 4.8
Two categories (union types): OWNER and REGISTERED_VEHICLE.

Category or Union

Example



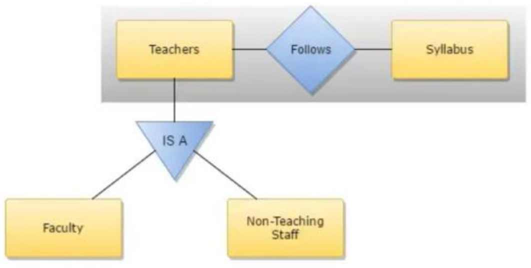
Aggregation

- In DBMS (Database Management System), aggregation is the **process of joining two or more entities** to create a more meaningful new entity.
- The aggregate method is used when the entities do not make sense on their own. In order to produce aggregation between two entities that cannot be used for their own attributes, a relationship is constructed and the end product is created into a new entity.
- Any form of relationship can be used, such as SUM, AVG, AND, OR, and so on. A wide range of tools are available on the market for table table aggregation.

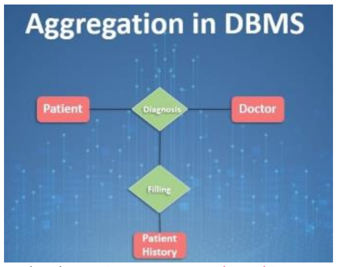
What is Aggregation?

- A relationship represents a connection between two entity types that are conceptually at the same level.
- Sometimes you may want to model a **'has-a,' 'is-a' or 'is-part-of' relationship,** in which one entity represents a larger entity (the 'whole') that will consist of smaller entities (the 'parts').
- This special kind of relationship is termed as an aggregation.
- Aggregation does not change the meaning of navigation and routing across the relationship between the whole and its parts.
- **An example of aggregation** is the 'Teacher' entity following the 'syllabus' entity act as a single entity in the relationship. In simple words, aggregation is a process where the relation between two entities is treated as a single entity.

What is Aggregation?



Specialization and Aggregation



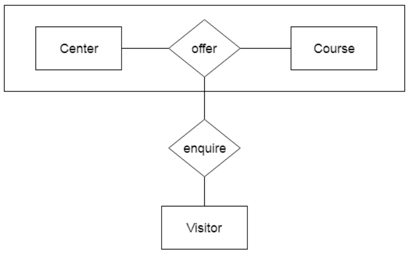
In this example, the patient cannot work on his own. He has to form a relationship with the doctor to get a diagnosis. The doctor also cannot perform a diagnosis without the patient. In the future, the doctor will need data about the patient's history, that will require him to collect it from a filing system.
The last entity (patient's history) ensures that the entire system is functional. Getting the patient's history cannot be done without a diagnosis from the doctor and a filing system.

Aggregation

- In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.
- **For example:** Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor.
In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.

Aggregation

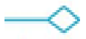
- **For example:** Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.



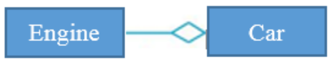
Aggregation

An example of aggregation is **the Car and Engine entities**.
A car is made up of an engine.
The car is the whole and the engine is the part.
Aggregation does not represent strong ownership. This means, a part can exist on its own without the whole. There is no stronger ownership between a car and the engine. An engine of a car can be moved to another car.

A line with a diamond at the end is used to represent aggregation.



For example, the Car-Engine relationship would be represented as shown below:



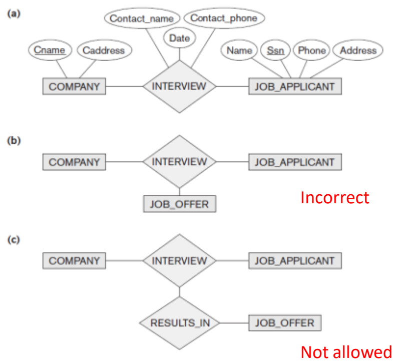


Figure 4.11
Aggregation. (a) The relationship type INTERVIEW. (b) Including JOB_OFFER in a ternary relationship type (incorrect). (c) Having the RESULTS_IN relationship participate in other relationships (not allowed in ER). (d) Using aggregation and a composite (molecular) object (generally not allowed in ER but allowed by some modeling tools). (e) Correct representation in ER.

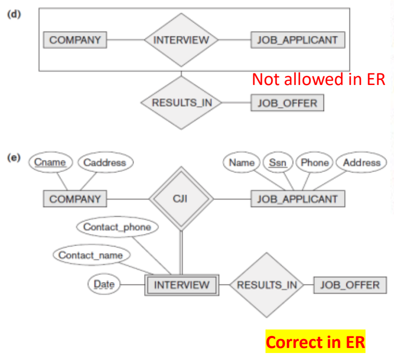


Figure 4.11
Aggregation. (a) The relationship type INTERVIEW. (b) Including JOB_OFFER in a ternary relationship type (incorrect). (c) Having the RESULTS_IN relationship participate in other relationships (not allowed in ER). (d) Using aggregation and a composite (molecular) object (generally not allowed in ER but allowed by some modeling tools). (e) Correct representation in ER.

Composition

Composition is a form of aggregation that represents an association between entities, where there is a strong ownership.

For example, a tree and a branch have a composition relationship. A branch is 'part' of a 'whole' tree - we cannot cut the branch and add it to another tree.

Enhanced Entity-Relationship (EER) Model

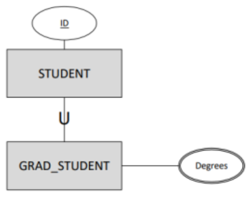
LMS Slide

- Additional entity types
  - Superclass: including one or more distinct subgroups in the data model
  - Subclass: a distinct subgroup of an entity type in the data model
- Attribute Inheritance
  - Specialization hierarchy (specialization: maximizing the differences between members of an entity by identifying their distinguishing characteristics)
  - Generalization hierarchy (generalization: minimizing the differences between entities by identifying their common characteristics)
  - Is-A hierarchy
- Constraints on specialization/generalization
  - Participation (mandatory, optional)
  - Disjoint: disjoint (or), non-disjoint (and)
- Other
  - Aggregation (has a or is part of)
  - Composition (strong ownership of aggregation)

LMS Slide

Specialization/Generalization

Only a subset of entities within a type have certain attributes or participate in certain relationships

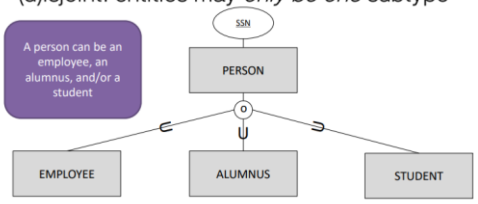


LMS Slid

Multiple Subtypes: Disjointedness

(o)verlap: may be more than one

(d)isjoint: entities may *only be one* subtype

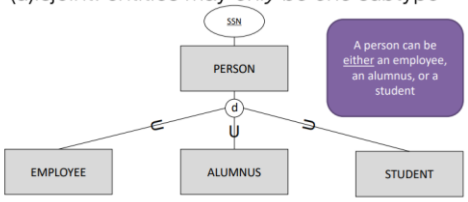


LMS Slid

Multiple Subtypes: Disjointedness

(o)verlap: may be more than one

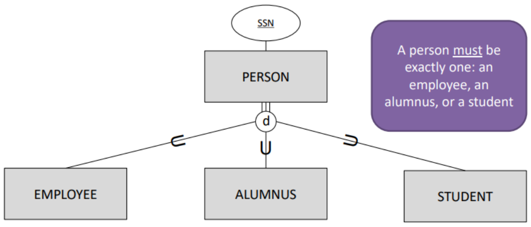
(d)isjoint: entities may *only be one* subtype


SSN
PERSON
A person can be either an employee, an alumnus, or a student
d
EMPLOYEE
ALUMNUS
STUDENT

LMS Slid

Multiple Subtypes: Completeness

Similar to relationships; can be total (<u>must</u> belong to subtypes) or partial (<u>can</u> belong)


SSN
PERSON
A person must be exactly one: an employee, an alumnus, or a student
d
EMPLOYEE
ALUMNUS
STUDENT

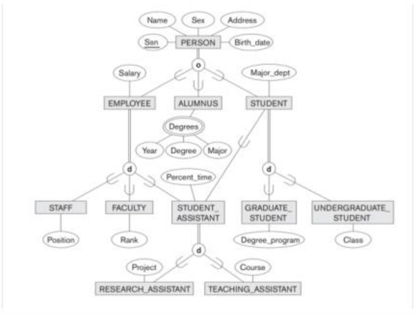
LMS Slid

Exercise

- The database keeps track of three types of persons: employees, alumni, and students. A person can belong to one, two, or all three of these types. Each person has a name, SSN, sex, address, and birth date.
- Every employee has a salary, and there are three types of employees: faculty, staff, and student assistants. Each employee belongs to exactly one of these types. For each alumnus, a record of the degree or degrees that he or she earned at the university is kept, including the name of the degree, the year granted, and the major department. Each student has a major department.
- Each faculty has a rank, whereas each staff member has a staff position. Student assistants are classified further as either research assistants or teaching assistants, and the percent of time that they work is recorded in the database. Research assistants have their research project stored, whereas teaching assistants have the current course they work on.
- Students are further classified as either graduate or undergraduate, with the specific attributes degree program (M.S., Ph.D., M.B.A., and so on) for graduate students and class (freshman, sophomore, and so on) for under- graduates.

LMS Slid

Answer



LMS Slid

Design Steps

- Identify
  - Entity types, relationship types
  - Cardinality and participation constraints
  - Attributes
  - Keys
  - Specialize/generalize
  - EER diagram
- EER model example

LMS Slide

Exercise:

- Create an enhanced ER diagram for a rental management using following entities:
  - Rental agency
  - Staff
    - Part time
    - Full time
  - Owner
  - Renter
  - Property
    - Business
    - Home

LMS Slide

A Sample Database Application

- COMPANY
  - Employees, departments, and projects
  - Company is organized into departments
  - Department controls a number of projects
  - Employee: store each employee’s name, Social Security number, address, salary, sex (gender), and birth date
  - Keep track of the dependents of each employee

LMS Slide

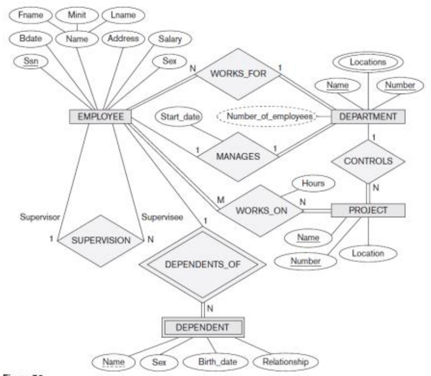


Figure 7.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 7.14.