04 EXTENDED ENTITY RELATIONSHIP MODEL

CS 340: INTRODUCTION TO DATABASE SYSTEMS

Adapted from: Dr. Omar Alomeir
2023

Course instructor:
Dr. Maram Alajlan.

LEARNING GOALS

CLO2: Develop conceptual modeling concepts and notations using Entity-Relationship ~~or UML diagrams~~ to analyze and design complex database applications. (Cognitive Skill)

Chapter objectives:

- Define 3 new concepts:
  - Sub-classes/Super-classes
  - Specialization/Generalization
  - Categories (UNION types)

DATABASES SO FAR

We learned that databases are useful but need to be carefully designed.

We learned how to design conceptual models using Entity Relationship diagrams

We will extend ER diagrams to support some new features.

TOPICS DISCUSSED IN CHAPTER 3

ER Model Concepts

- Entities and Attributes
- Entity Types
  - Strong Entity
  - Weak Entity Types
- Various Types of Attributes
  - Simple, Composite, Multi-valued, Derived, Key Attributes
- Relationships and Relationship Types
- Constraints on relationships
- Roles and Attributes in Relationship Types

EER

WHY ENHANCED ER MODEL?

The traditional ER Model concepts were difficult to use for database modelling of complex data. Hence some improvements or enhancements were made to the existing ER Model to make it able to handle the complex applications better.

EER is a high-level data model that incorporates the extensions to the original ER model.

CHAPTER OUTLINE

EER stands for Enhanced ER or Extended ER

EER Model Concepts

- Includes all modeling concepts of basic ER
- 3 additional concepts:
  - Sub-classes/Super-classes
  - Specialization/Generalization
  - Categories (UNION types)
- Constraints on Specialization/Generalization

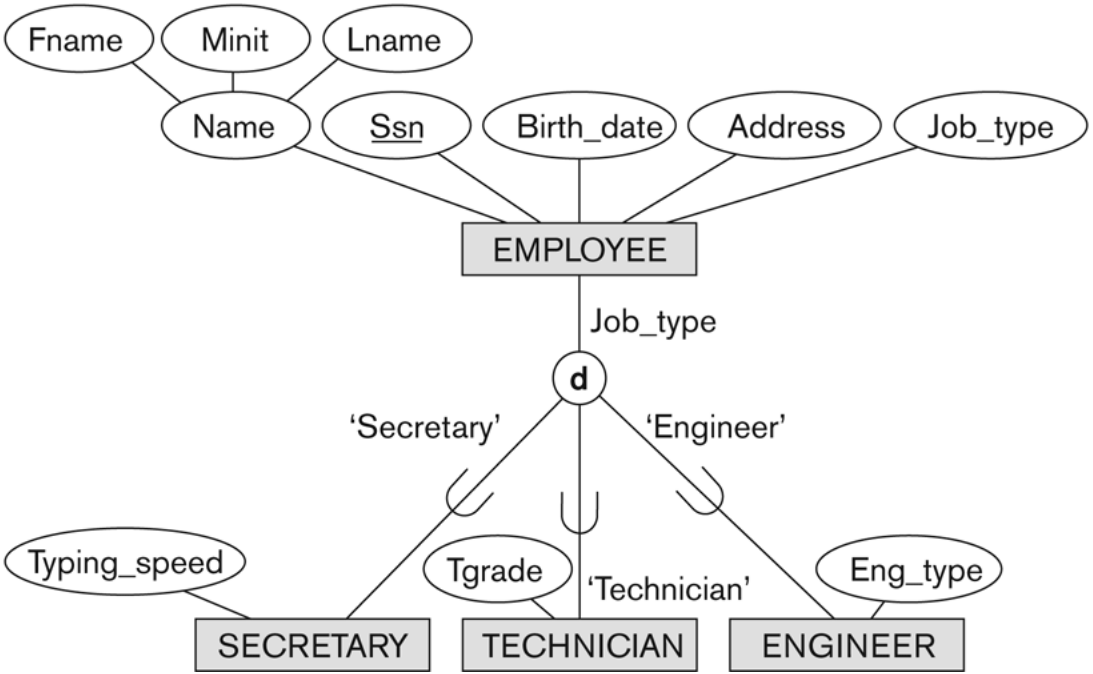
SUB CLASSES AND SUPER CLASSES

An entity type may have additional meaningful subgroupings of its entities meaningful to the organization

- Example: EMPLOYEE may be further grouped into:
  - SECRETARY, ENGINEER, TECHNICIAN, …
    - Based on the EMPLOYEE's Job
  - MANAGER
    - EMPLOYEEs who are managers (the role they play)
  - SALARIED_EMPLOYEE, HOURLY_EMPLOYEE
    - Based on the EMPLOYEE's method of pay

EER diagrams extend ER diagrams to represent these additional subgroupings, called *subclasses* or *subtypes*

SUB CLASSES AND SUPER CLASSES



SUB CLASSES AND SUPER CLASSES

Each of these subgroupings is a subset of EMPLOYEE entities

Each subset is called a **subclass** of EMPLOYEE

EMPLOYEE is the **superclass** for each of these subclasses

These are called superclass/subclass relationships:

- EMPLOYEE/SECRETARY
- EMPLOYEE/TECHNICIAN
- EMPLOYEE/MANAGER
- …

These are also called IS-A relationships

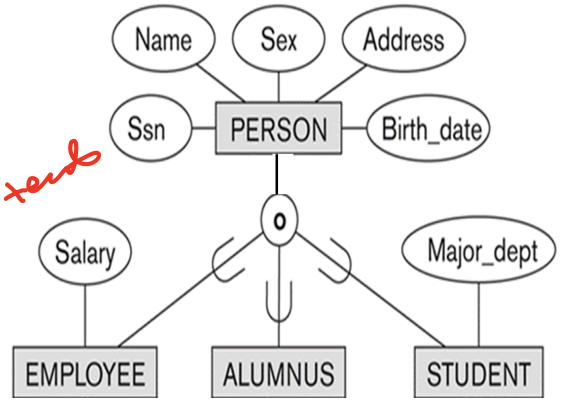
- SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE, ….

SUB CLASSES AND SUPER CLASSES

An entity that is member of a sub-class represents the same real-world entity as some member of the super-class:

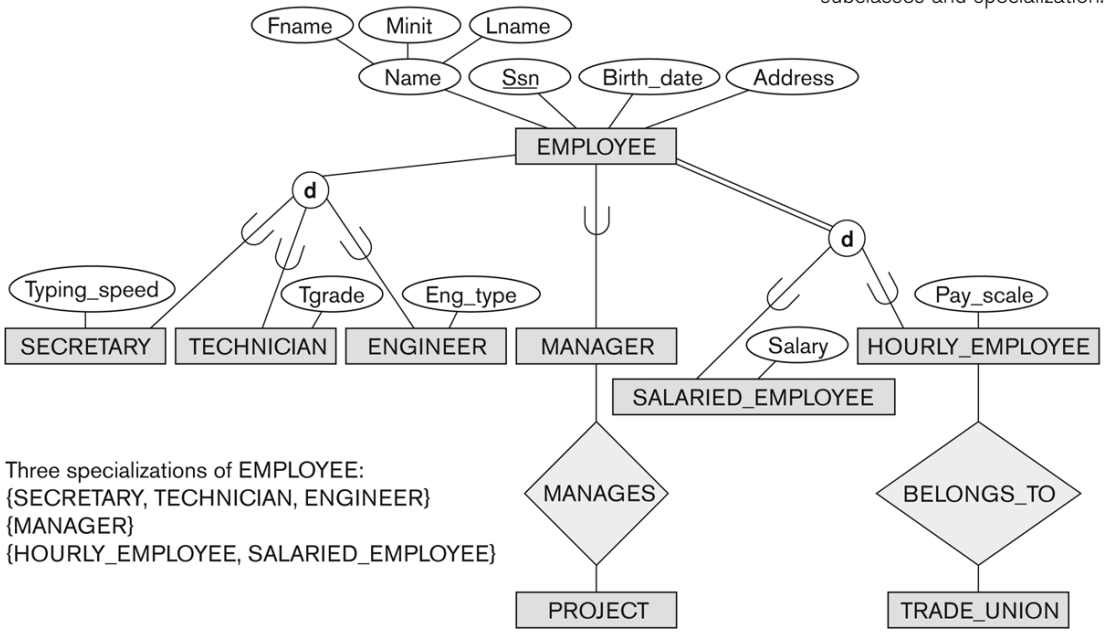
- The subclass member is the same entity in a *distinct specific role*
- An entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the superclass.

It is not necessary that every entity in a superclass be a member of some subclass



SUB CLASSES AND SUPER CLASSES

Figure 4.1
EER diagram notation to represent subclasses and specialization.



SUB CLASSES AND SUPER CLASSES

A member of the superclass can be optionally included as a member of any number of its subclasses

Examples:

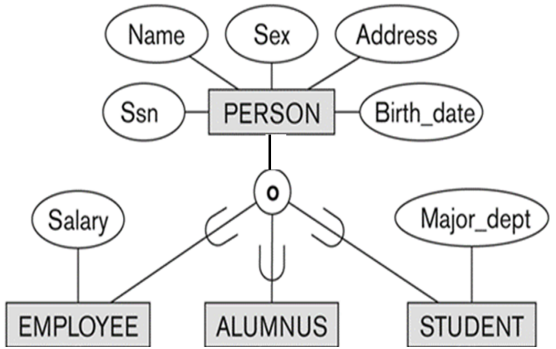
- A salaried employee who is also an engineer belongs to the two subclasses:
  - ENGINEER, and
  - SALARIED_EMPLOYEE
- A salaried employee who is also an engineering manager belongs to the three subclasses:
  - MANAGER,
  - ENGINEER, and
  - SALARIED_EMPLOYEE

BASIC NOTATION

The superclass is connected with a line to a circle, which in turn is connected by a line to each subclass that has been defined.

The U- shaped symbol on each line connecting a subclass to the circle indicates that the subclass is a subset of the superclass.

It also indicates the direction of the subclass/superclass relationship.



ATTRIBUTE INHERITANCE IN SUPERCLASS / SUBCLASS RELATIONSHIPS

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:

- In the previous slide, SECRETARY (as well as TECHNICIAN and ENGINEER) inherit the attributes Name, SSN, …, from EMPLOYEE
- Every SECRETARY entity will have values for the inherited attributes

WHEN IS IT NECESSARY TO INCLUDING SUPERCLASS/SUBCLASS RELATIONSHIP?

There are two reasons for including superclass/subclass relationship in data model:

- Certain attributes may apply to some but not to all entities of the superclass. Thus, a subclass is defined to group the entities to which these attributes apply.
- Some relationship types can only allow certain sub-types to participate.

SPECIALIZATION

عملية تحديد التخصصات

Specialization is the process of defining a set of subclasses of a superclass

The set of subclasses is based upon some distinguishing characteristics of the entities in the superclass

- Example: {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of EMPLOYEE based upon *job type.*
- Example: MANAGER *is a specialization of EMPLOYEE based on the role the employee plays*
  - May have several specializations of the same superclass

GENERALIZATION

Generalization is the process of generalizing the entities which contain the common attributes of all the generalized entities.

It is a bottom approach, in which two lower-level entities combine to form a higher-level entity. It's more like Superclass from a group of common sub classes Subclass system

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.

GENERALIZATION

Generalization is the reverse of the specialization process

Several classes with common features are generalized into a superclass;

- original classes become its subclasses

Example: CAR, TRUCK generalized into VEHICLE;

- both CAR, TRUCK become subclasses of the superclass VEHICLE.
- We can view {CAR, TRUCK} as a specialization of VEHICLE
- Alternatively, we can view VEHICLE as a generalization of CAR and TRUCK

GENERALIZATION

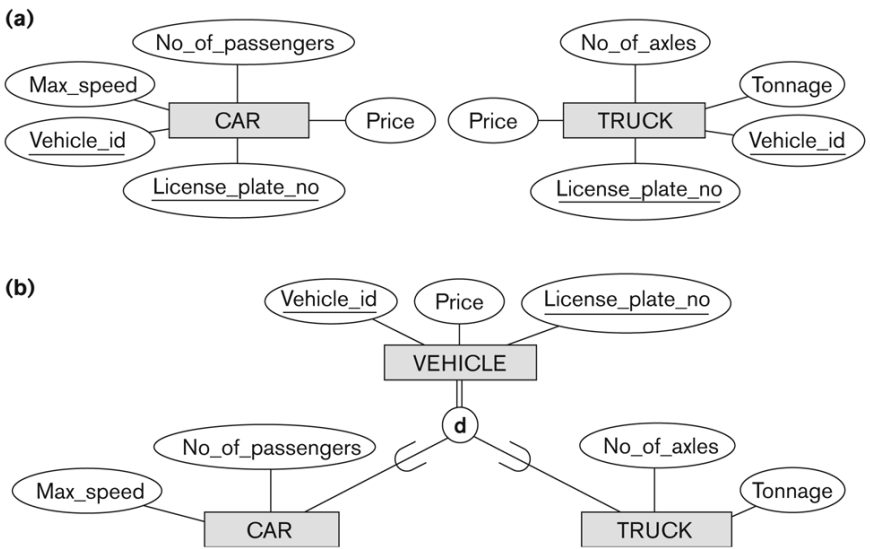


Figure 4.3
Generalization. (a) Two entity types, CAR and TRUCK.
(b) Generalizing CAR and TRUCK into the superclass VEHICLE.

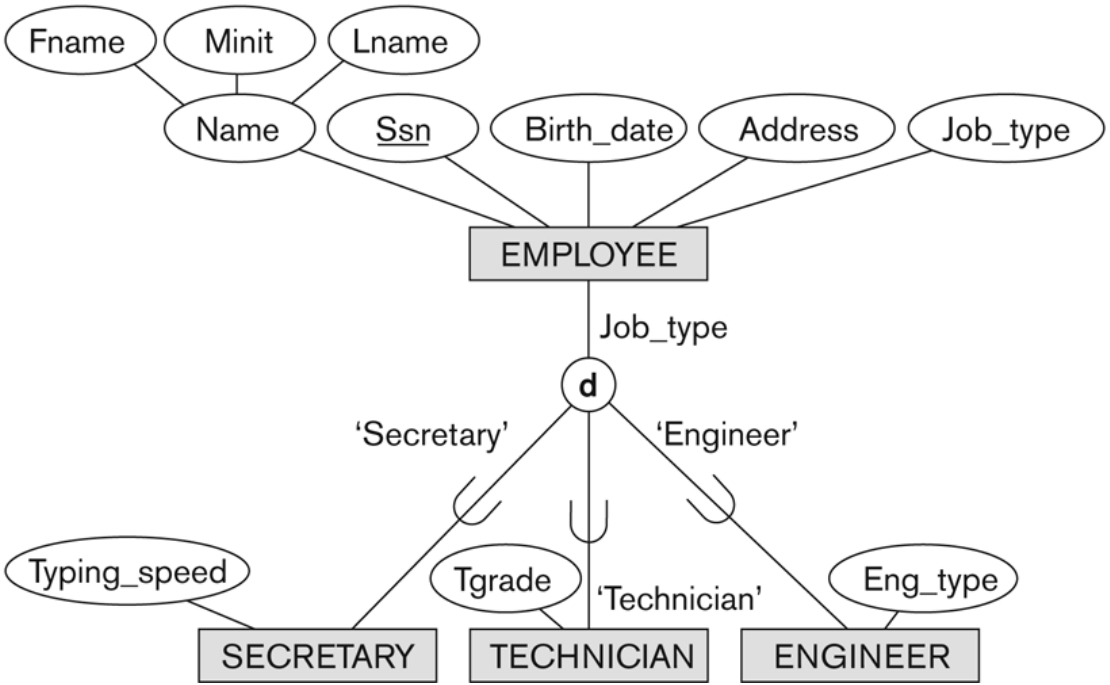
ATTRIBUTE-DEFINED SPECIALIZATION IN EER DIAGRAMS

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 or Discriminator.

Example: Job Type is the defining attribute of the specialization {SECRETARY, TECHNICIAN, ENGINEER} of EMPLOYEE

DISPLAYING AN ATTRIBUTE-DEFINED SPECIALIZATION IN EER DIAGRAMS



CONSTRAINTS ON SPECIALIZATION AND GENERALIZATION

Two basic constraints can apply to a specialization/generalization:

- Disjoint-ness Constraint
- Completeness Constraint

CONSTRAINTS ON SPECIALIZATION AND GENERALIZATION

Disjoint-ness Constraint:

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

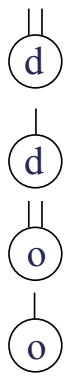
CONSTRAINTS ON SPECIALIZATION AND GENERALIZATION (3)

Completeness (Exhaustiveness) Constraint:

- *Total* specifies that every entity in the superclass must be a member of some subclass in the specialization/generalization
- The same as total participation constraint in ER
- Shown in EER diagrams by a ***double line***
- *Partial* allows an entity not to belong to any of the subclasses
- Shown in EER diagrams by a single line

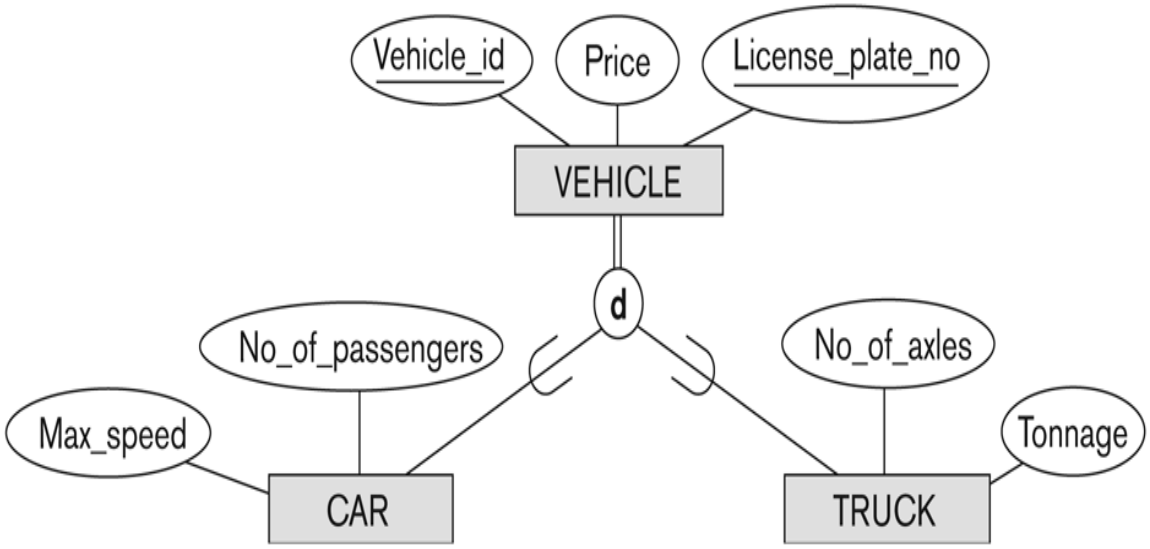
CONSTRAINTS ON SPECIALIZATION & GENERALIZATION

- The disjointness and completeness constraints are independent.
- Four possible constraints in specialization:
  - Disjoint, total.
  - Disjoint, partial.
  - Overlapping, total.
  - Overlapping, partial.

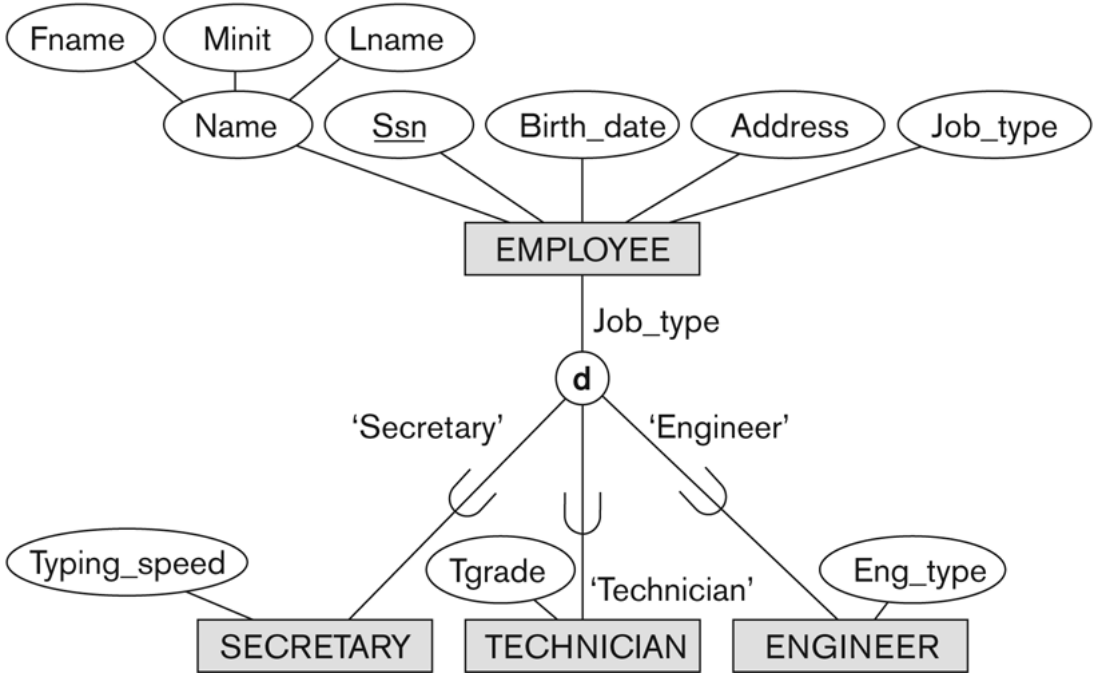


- *Generalization is usually total. why?*

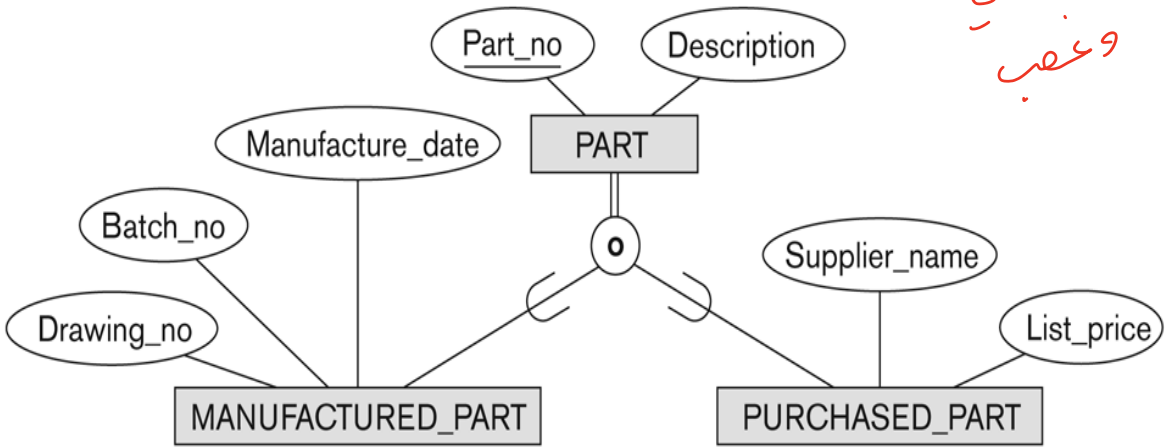
EXAMPLE OF DISJOINT & TOTAL PARTICIPATION



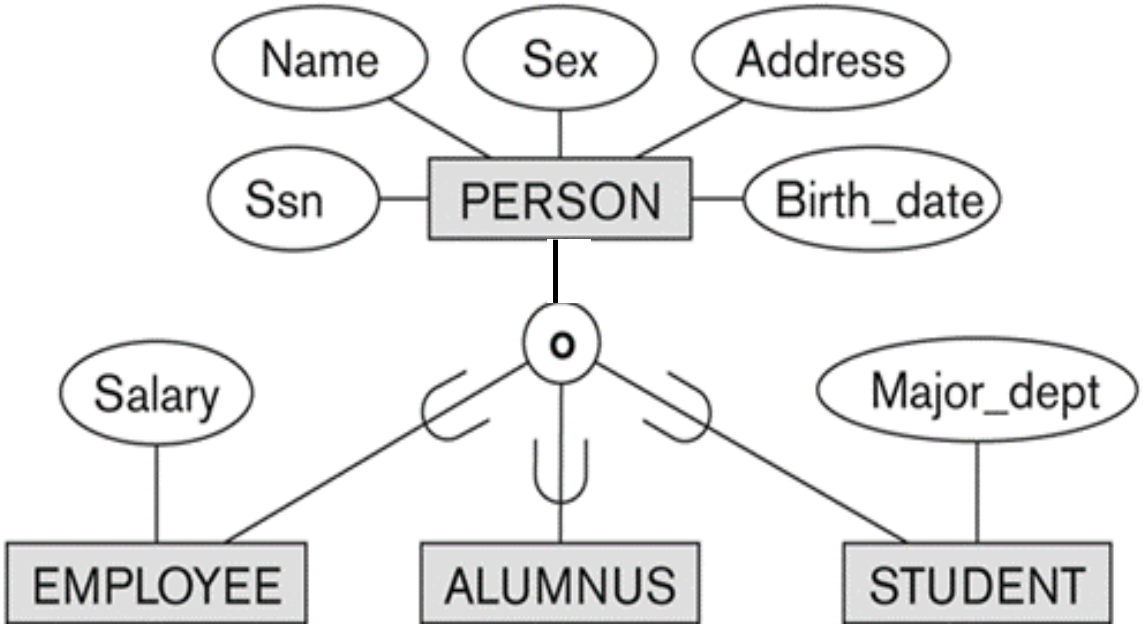
EXAMPLE OF DISJOINT & PARTIAL PARTICIPATION



EXAMPLE OF OVERLAPPING & TOTAL PARTICIPATION



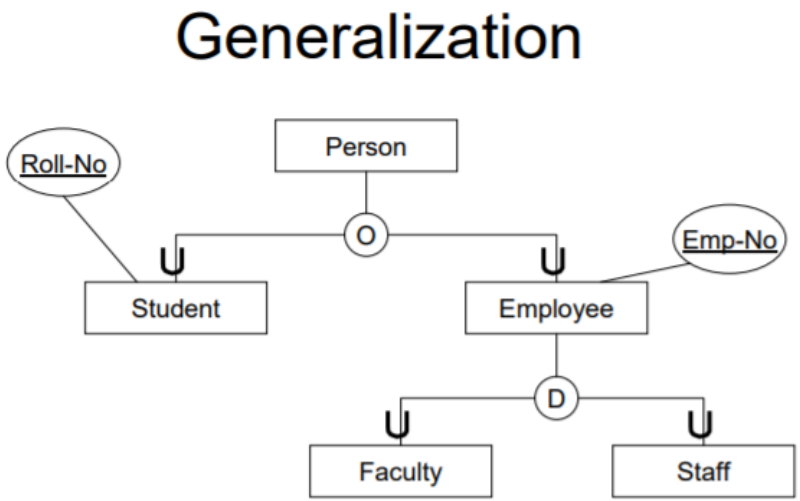
EXAMPLE OF OVERLAPPING & PARTIAL PARTICIPATION



CONSTRAINTS ON SPECIALIZATION & GENERALIZATION

- Certain insertion and deletion rules apply to specialization and generalization as a result of the disjoint and completeness constraints:
- Deleting an entity from a superclass implies that it is automatically deleted from all the subclasses to which it belongs.
- Inserting an entity in a superclass of a **total specialization** implies that the entity is mandatory inserted in at least one of the subclasses of the specialization.
- Inserting an entity in a superclass of a **partial specialization** implies that the entity is not necessarily inserted in any of the subclasses of the specialization.

ANOTHER EXAMPLE OF DISJOINT & OVERLAPPING



HIERARCHIES, LATTICES, & SHARED SUB-CLASSES

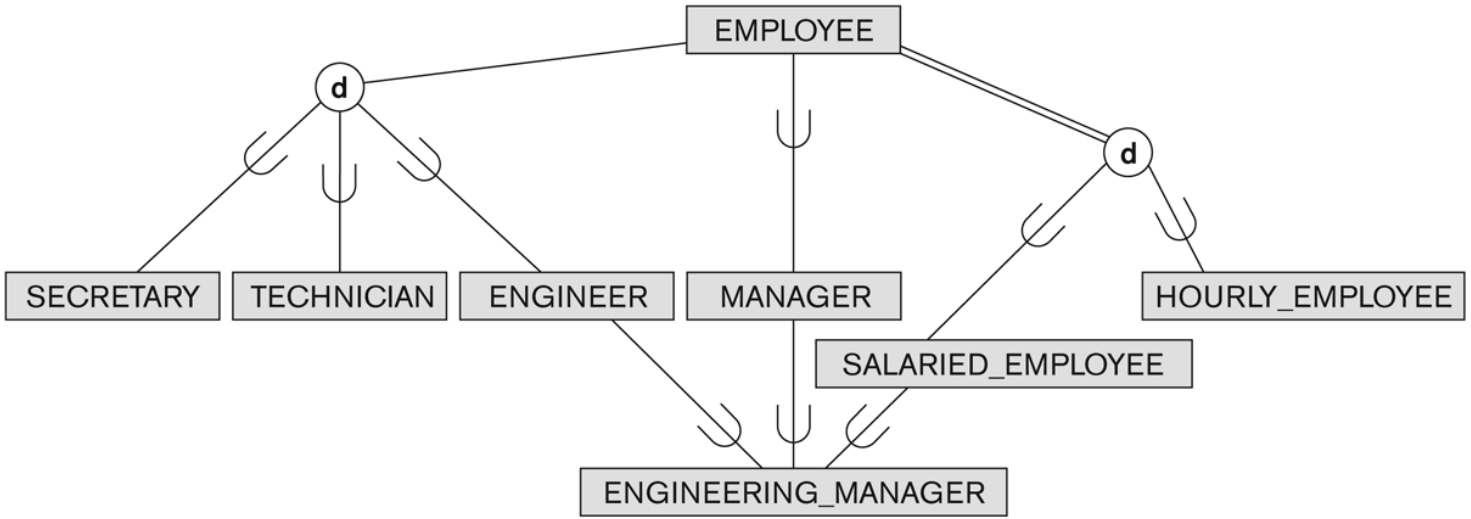


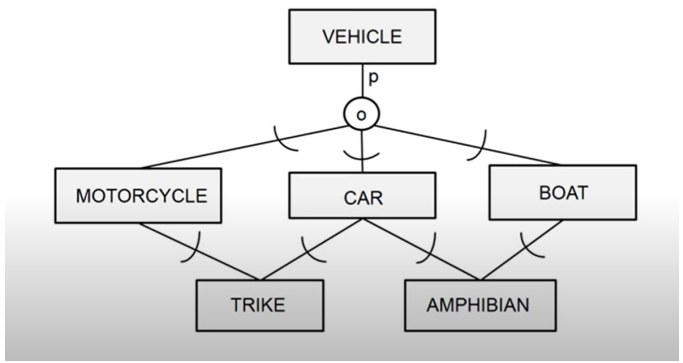
Figure 4.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.

HIERARCHIES, LATTICES & SHARED SUB-CLASSES

lattices (*multiple inheritance*) and ***hierarchies*** (*single inheritance*).

In a lattice or hierarchy, a subclass inherits attributes or relationships not only of its direct superclass, but also of all its predecessor superclasses

A subclass with more than one superclass is called a shared subclass (multiple inheritance). It inherits from all of

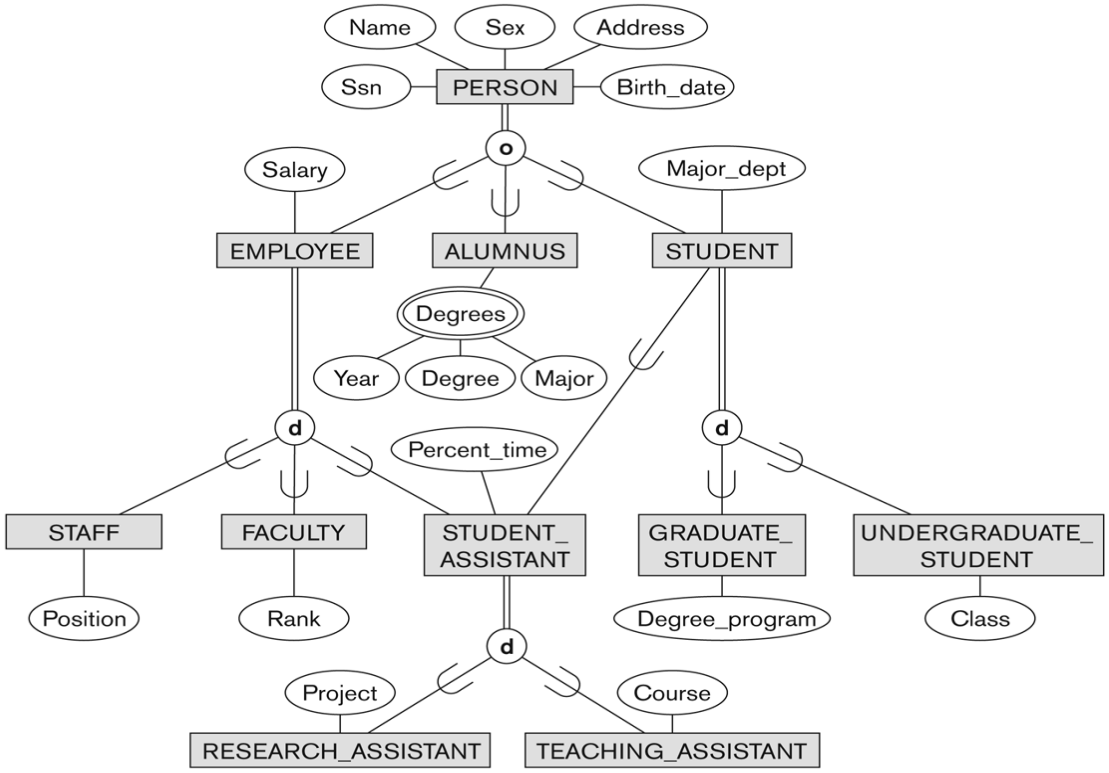


LATTICE EXAMPLE (UNIVERSITY)

The requirements for this UNIVERSITY database are the following:

1. 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.

2. 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.

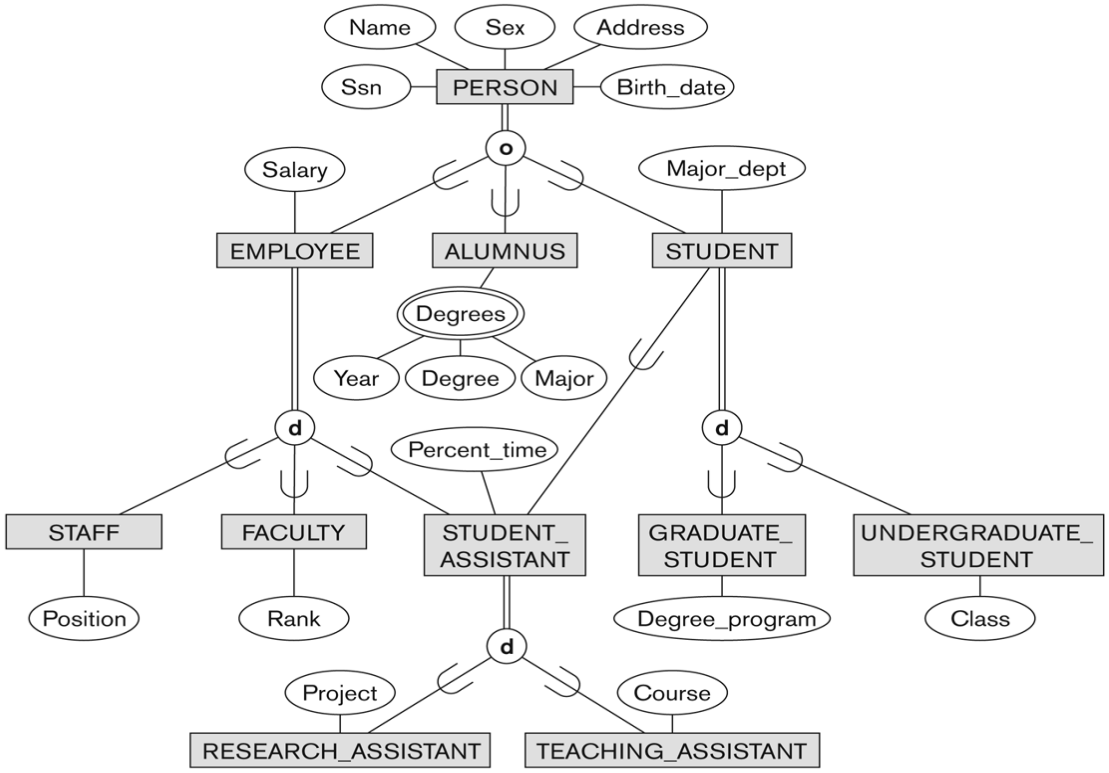


LATTICE EXAMPLE (UNIVERSITY)

The requirements for this UNIVERSITY database are the following:

3. 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.

4. 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 undergraduates.



CATEGORIES (UNION TYPES)

In some cases, we need to model a *single superclass/subclass relationship* with *more than one* superclass

A category is a subclass that has several possible super-classes.

Each superclass represents a different entity type.

The category represents a collection of entities that is subset of the UNION of the entities (super classes) of distinct entity types.

Such a subclass is called a category or UNION TYPE.

Attribute inheritance works more selectively in the case of categories

CATEGORIES (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.

- A *category* (UNION type) called OWNER is created to represent a subset of the *union* of the three super-classes COMPANY, BANK, and PERSON
- A category member must exist in ***one*** of its super-classes

Difference from *shared subclass*, which is a:

- subset of the *intersection* of its super-classes
- shared subclass member must exist in ***all*** of its super-classes at the same time.

CATEGORIES (OWNER, AND REGISTERED_VEHICLE)

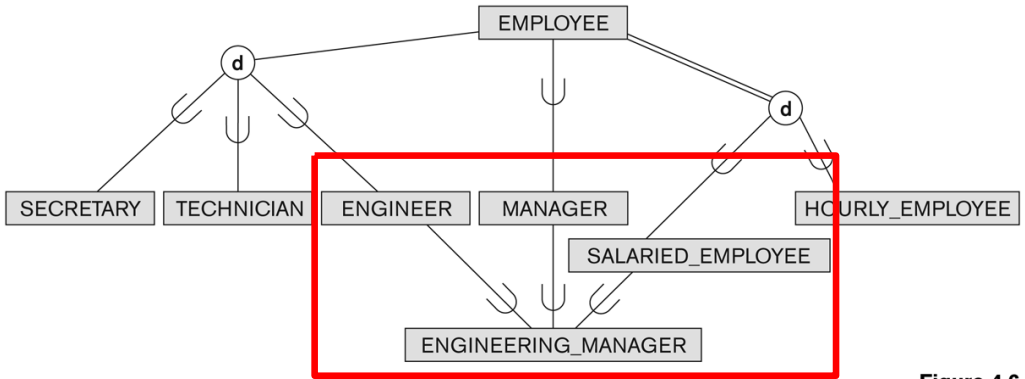
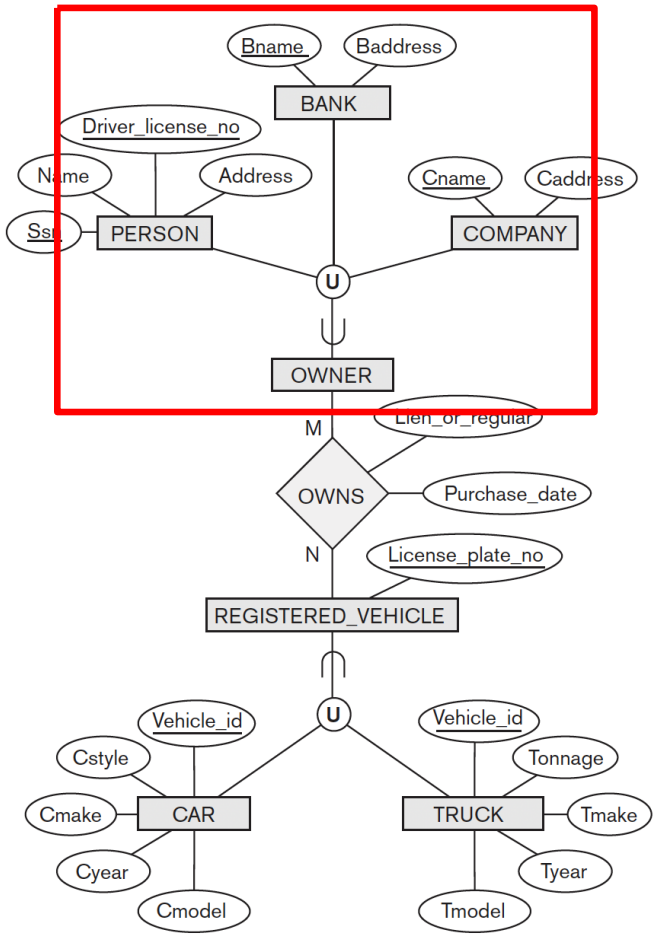


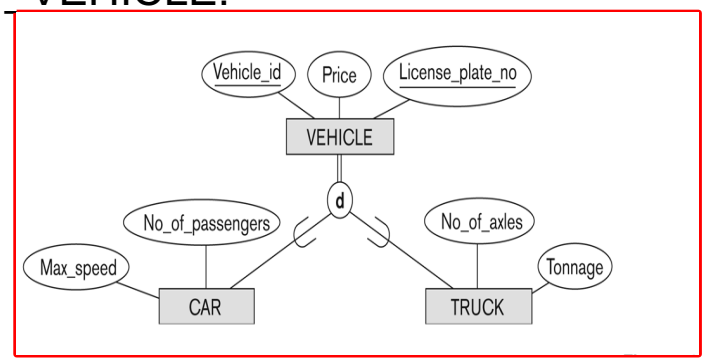
Figure 4.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.

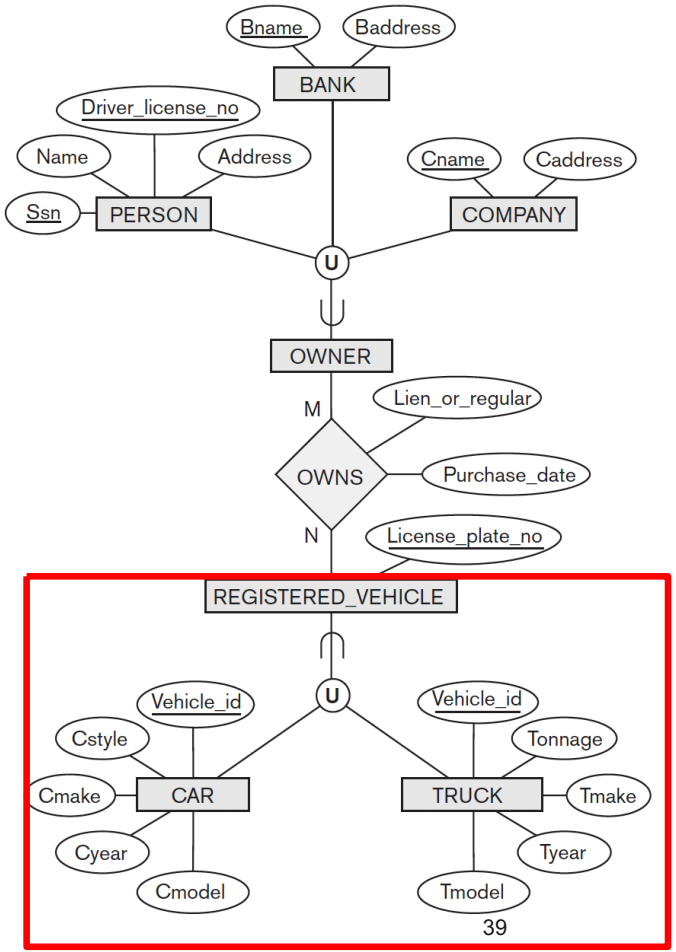


CATEGORIES (OWNER, AND REGISTERED_VEHICLE)

REGISTERED_VEHICLE is different from the generalized superclass VEHICLE, every car and every truck is a VEHICLE; the REGISTERED_VEHICLE category includes some cars and some trucks but not necessarily all of them.

A specialization or generalization, if partial, would not prevent VEHICLE from containing other types of entities, such as motorcycles. However, a category such as REGISTERED_VEHICLE implies that only cars and trucks can be members of REGISTERED_VEHICLE.





CATEGORIES INHERITANCE

Inheritance in the case of categorization corresponds to an entity inheriting only the attributes and relationships of that superclass, it is a member of (selective inheritance)

A categorization can be **total or partial**. A total category holds the **union of all entities** in its superclasses, whereas a partial category can hold a **subset of the union of all entities** .

A total category is represented diagrammatically by a **double line** connecting the category and the circle, whereas a partial category is indicated by a **single line.**

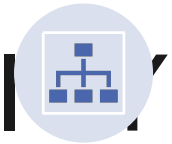
CATEGORIES (UNION TYPES)

The super-classes of a category may have different key attributes, as demonstrated by the OWNER category, or they may have the same key attribute, as demonstrated by the REGISTERED_VEHICLE category.

Notice that if a category is total (not partial), it may be represented alternatively as a total specialization (or a total generalization). In this case, the choice of which representation to use is subjective.

If the two classes represent the same type of entities and share numerous attributes, including the same key attributes, specialization/generalization is preferred; otherwise, categorization (union type) is more appropriate.

ACTIVITY



Consider a University Database System used at PSU, which stores information about the:

- Student Club identified by ClubName, President, and NumofMembers.
- Sports team identified by the TeamName, Coach, and the SportsName.
- Department identified by the DeptName and OfficeNum.
- A Sponsor of an event is identified by SponsorID. A sponsor may be a team, a department, or a club. Each sponsor entity instance is a member of one of these superclasses, so Sponsor is a subclass of Team, Dept, Club. Not every Club, Team, or Dept must be a sponsor.
- A sponsor holds one or many CampusWide Events. However, an event must be organized by one sponsor. CampusWide Events are identified by EventName, Date and time. CampusWideEvents can be of two types: Concerts and Fair. A Concert has a performer, and a Fair can also be described by Theme and Charity.

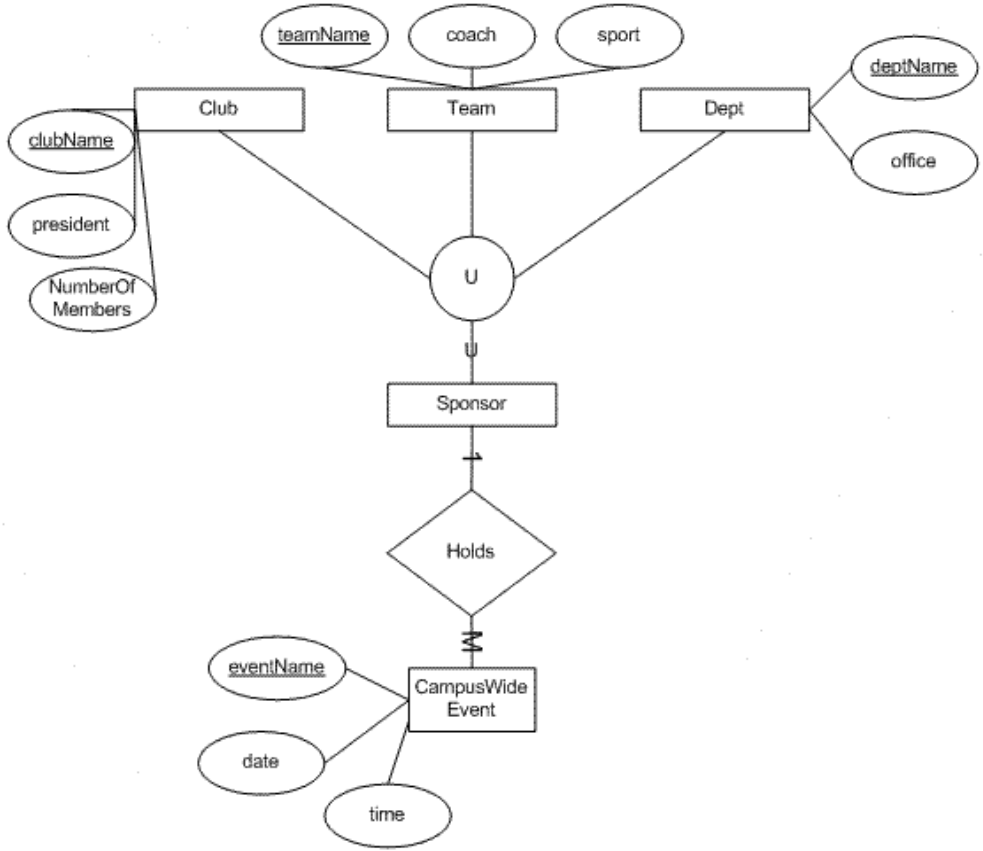
Draw an ER Diagram for the above scenario, showing cardinality and participation for all relationships.

SOLUTION

Sponsor is a sub-class of the union of Club, Team, and Dept. Why?

Why not use a subclass/superclass generalization instead?

CampusWide Event is further split into disjoint Fair, and Concert events with their respective attributes.



AGGREGATION (BRIEFLY)

Aggregation is an abstraction concept for building composite objects from their component objects.

b) requires each interview relationship instance to have a job offer.

c) Is not allowed in ER.

d) Allowed in some models.

e) Works for our model!

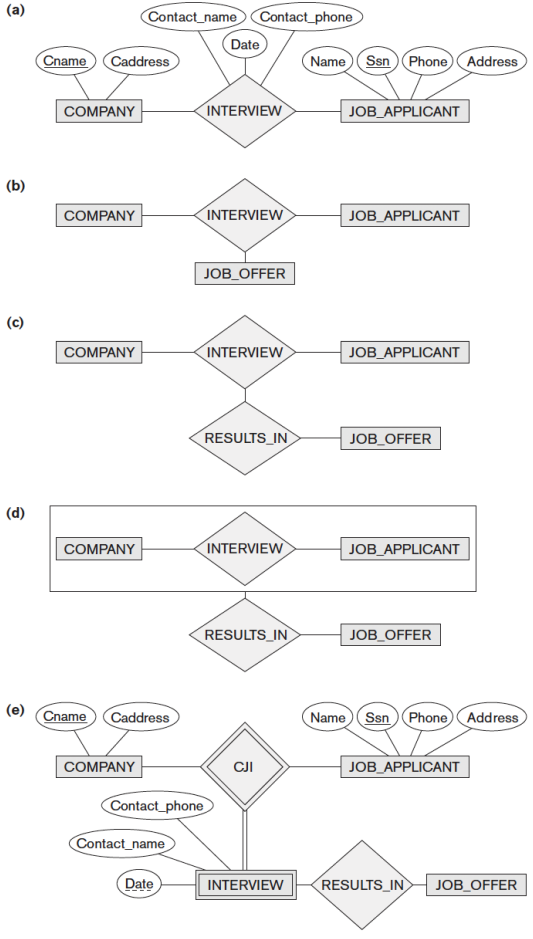


Figure 4.11
Aggregation. (a) The relationship type INTERVIEW. (b) Including JOB_OFFER in a ternary relationship type (incorrect). (c) Having the RESULTS_IN relationship participate in other relationships (not allowed in ER). (d) Using aggregation and a composite (molecular) object (generally not allowed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

SUMMARY OF EER CONCEPTS

- Super-classes and sub-classes (Generalization and specialization)
- Categories or Union types
- Aggregation

All of these concepts improve the expressive power of ER diagrams and enable us to design very complex database applications.

Next, we will see how to create logical designs and dive into the relational model.

PROJECT FIRST STEPS

1. Form a team of 3 or 4
2. Find a relational database management system that you want to use and start preparing by:
    1. Installing the DBMS somewhere.
    2. Making sure the installation is working.
    3. Letting me know which system you will be using.
3. Make a team decision on the technology you will use.
4. Start thinking about the data and domain you will explore. I can provide some ideas if you need.