Chapter (4)
Enhanced Entity-Relationship and Object Modeling

Objectives

• Concepts of subclass and superclass and the related concepts of specialization and generalization.
• Concept of category, which is used to represent a collection of objects that is the union of objects of different entity types.
• Concepts is the important mechanism of attribute and relationship inheritance.
• Diagrammatic technique for displaying these concepts when they arise in an EER schema (enhanced-ER or EER diagrams).
• Object Data Modeling methodologies.

An entity type is used to represent both a type of entity, and the entity set or collection of entities of that type that exist in the database. In many cases an entity type has numerous subgroupings of its entities that are meaningful and need to be represented explicitly because of their significance to the database application.

Subclasses, Superclasses, and Inheritance

The first EER model we discuss is the subclass of an entity type.

So far we know that an entity type is used to:
• represent a type of entity, and the entity set, or
• collection of entities of that type that exist in the database.

Example: 
EMPLOYEE entity refers to:
The attributes and relationships of each employee entity, OR
Current set of EMPLOYEE entities in the COMPANY database.

An entity may have subgrouping of its entities that are meaningful. That may be due to their significance to the database application.

Example: 
EMPLOYEE in a group with SECRETARY, ENGINEER, MANAGER, TECHNICIAN, SALARIED_EMPLOYEE, HOURLY_EMPLOYEE, and so on.

Each of these is a subclass of EMPLOYEE.
Subclasses, Superclasses, and Inheritance

The relationship between a superclass and a subclass is called superclass/subclass or class/subclass relationship.

A subclass represents that ‘same real-world entity’ as some member of the superclass. Mr. John Data-holder can be both. He is an employee but in a 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.

Example: salaried_employee who is also an engineer. But, in general, it is not necessary that every entity in a superclass be a member of some subclass.

An entity in the subclass represents the same real-world entity from the superclass, it should posses values for its specific attributes as well as values of its attributes as a member of the superclass.

We say that an entity that is a member of a subclass inherits all the attributes of the entity as a member if superclass. The entity in the subclass also inherits all the relationships in which the superclass participates.

Specialization and Generalization

Specialization is the process of defining a set of subclasses of an entity type; this entity type is called the superclass of the specialization.

Members of the specialization are defined based on some distinguishing characteristics of the entities in the superclass.

We may have several specializations of the same entity type based on different distinguishing characteristics.

Think about method of pay or type of job as characteristics.

A subclass may have additional attributes, which are called specific attributes or local attributes of the subclass. Also, a subclass may participate in additional relationship type (specific relationship types).

There are two main reasons for including class/subclass relationships and specialization in a data model:

1) Certain attributes may apply to some but not all entities of the superclass,
2) Some relationship types may be participated only by entities that are members of the subclass.
Specialization and Generalization

The specialization process allows us to do the following:
• Define a set of subclasses of an entity type,
• Establish additional specific attributes with each subclass,
• Establish additional specific relationship types between each subclass and other entity types or other subclasses.

Generalization it is the reverse process of abstraction in which we suppress the differences among several entity types, identify their common features, and generalize them into a single superclass of which the original entity types are special subclasses. Generalization is the process of defining a generalized entity type from the given entity types.
Figure 4.2  Some instances of the specialization of EMPLOYEE into the
{SECRETARY, ENGINEER, TECHNICIAN} set of subclasses.

Figure 4.3  Examples of generalization. (a) Two entity types CAR and
TRUCK. (b) Generalizing car and TRUCK into VEHICLE.
**Constraints and Characteristics of Specialization and Generalization**

How do we determine exactly the entities that will become members of each subclass?

In some specialization we can do this by placing a condition on the value of some attribute of the superclass. Such subclasses are called **predictive-defined** (or **condition-defined**) subclasses.

For example the EMPLOYEE entity may have an attribute called JobType. Then for a SECRETARY subclass we predicate (JobType = 'Secretary'), which we call the **defining predicate** of the subclass. This condition is satisfied in both the EMPLOYEE and SECRETARY, how?

- in EMPLOYEE all secretaries must have JobType = 'Secretary'
- in SECRETARY all members must satisfy the predicate.

If all subclasses in a specialization have a membership condition on the same attribute of the superclass, the specialization itself is called an **attribute-defined specialization**, and the attribute is called the **defining attribute** of the specialization.

When we do not have a condition for determining membership in a subclass, the subclass is called **user-defined**.

Membership in such a subclass is determined by the database users when they apply the operation to add an entity to the subclass.

Each membership is specified individually for each entity by the user, NOT by any condition that may be evaluated automatically.

Two other constraints may apply to specialization:

- **disjointness constraint**, which specifies that the subclass of the specialization must be disjoint. This means that an entity can be a member of **at most one of the subclasses** of the specialization.

A specialization that is attribute defined implies the disjointness constraint if the attribute used to define the membership predicate is single valued.

- **completeness constraint**, which may be total or partial. A **total specialization** constraint specifies that every entity in the superclass must be a member of some subclass in the specialization.
Constraints and Characteristics of Specialization and Generalization

- **completeness constraint**, which may be total or partial. A **total specialization** constraint specifies that every entity in the superclass must be a member of some subclass in the specialization. For example the constraint between the EMPLOYEE and \{HOURLY_EMPLOYEE, SALARIED_EMPLOYEE\}. A total specialization is shown with a double line. A **partial specialization** allows an entity to belong to any of the subclasses.

In summary, we have the following four possible constraints on specialization:

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

Note: a superclass that was identified through the generalization process usually is total. Why?
Because the superclass is derived from the subclasses.
Constraints and Characteristics of Specialization and Generalization

Rules that apply to insertion and/or deletion of specialization (or generalization):

• 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 implies that the entity is mandatorily inserted in all predicate-defined or attribute-defined subclasses for which the entity satisfies the defining predicate, and
• 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.

A subclass with more than one superclass is called a shared subclass. Example: If every engineer must be a SALRIED_EMPLOYEE and MANAGER, then ENGINEERING-MANAGER should be a shared subclass of all three subclasses.

In such cases the shared subclass may inherit several attributes and relationships from multiple classes, this is called multiple inheritance.

Figure 4.5  Notation for specialization with overlapping (nondisjoint) subclasses.
Figure 4.6 A specialization lattice with the shared subclass ENGINEERING_MANAGER.
Utilizing Specialization and Generalization in Conceptual Data Modeling

In specialization process, we typically start with an entity type and then define subclasses of the entity type by successive specialization. This successive specialization corresponds to a **top-down conceptual refinement process**.

It is possible to arrive at the same thing using generalization process. In such a case, the process is **bottom-up conceptual synthesis**.

Modeling of UNION Types using categories

Sometimes, there is a need for modeling a single superclass/subclass relationship with more than one superclass. In such a case the superclasses represent different entity types.

In such a case the subclass will represent a collection of objects that is the UNION of distinct entity types. We call such a subclass a **union type** or a **category**.

Example: PERSON, BANK, and COMPANY

A car can be owned by any of them, but you have one vehicle registration. A category **OWNER** is a subclass of the UNION of the three entity sets of PERSON, BANK, and COMPANY.

A category has two or more superclasses that may represent distinct entity types, whereas other superclass/subclass relationships always have a single superclass.

Can you find the difference between

OWNER – PERSON, COMPANY, BANK and

ENGINEER_MANAGER – MANAGER, SALARIED_EMPLOYEE, ENGINEER ?
Utilizing Specialization and Generalization in Conceptual Data Modeling

Attribute inheritance works more selectively in the case of categories. A category such as OWNER entity inherits the attributes of COMPANY, a PERSON, or a BANK, depending on the superclass to which the entity belongs.

On the other hand, a shared subclass such as ENGINEER-MANAGER inherits all the attributes of its superclasses SALARIED_EMPLOYEE, ENGINEER, and MANAGER.

A category can be total or partial.
An Example – UNIVERSITY Database (Fig 4.10)

Requirements:
For each person, that database maintain information on the person’s Name [name], social security number [Ssn], address [Address], sex [Sex], and birth date [BDate].

Two subclasses of the PERSON entity are FACULTY and STUDENT.

FACULTY should have attributes for rank [Rank], office [Office], office phone [FPhone], and salary [Salary]. All faculty BELONGS to one or more departments.

STUDENT should have attributes for class [Class]. Students are related to their MAJOR or MINOR, to the course sections [REGISTERED], and to the course completed [TRANSCRIPT]. Each TRANSCRIPT instance includes the grade student received [Grade] in the course section.

GRAD_STUDENT is a subclass of STUDENT. In addition, they have a multi-valued attribute [Degree]. They also are related to a faculty [ADVISOR] and to a thesis [COMMITTEE], if one is in place.
An academic department has the attribute name [DName], telephone … I let you fill the rest …

The course has attribute course number [C#], …

There is at least one category presented, which is …
It is the UNION of …

The entity GRANT keep track of …
Universal Modeling Language (UML)

A class is displayed as a box with three partitions:
- Top Part: Name of the entity
- Middle Part: Attributes
- Bottom Part: Operations allowed

Relation types are called **associations**. Relationship instances are called **links**.

A dash line to a box identifies the **link attribute**.

The **min, max** is used to specify relationship constraints, which is called **multiplicities** in UML.

There are usually two types of relationships in UML: **association** and **aggregation**. Aggregation is meant to represent a relationship between a whole object and its component parts. See the locations of a department. For now assume there is no structural differences between association and aggregation.

Weak entities are identified by **qualified association** in UML.

A blank triangle indicates a **disjoint** specialization/generalization, filled one otherwise.
It was: Schema Diagram of COMPANY

EMPLOYEE
FNAME | UNIT | LNAME | SSN | BDATE | ADDRESS | SEX | SALARY | SUPERSSN | DNO

DEPARTMENT
CNAME | DNUMBER | MGRSSN | MGRSTARTDATE

DEPT_LOCATIONS
DNUMBER | DLOCATION

PROJECT
PNAME | PNUMBER | PLOCATION | DNUM

WORKS ON
ESSN | PMO | HOURS

DEPENDENT
ESSN | DEPENDENT_NAME | SEX | BDATE | RELATIONSHIP

PERSON
Name
Stit
Birthday
Sex
Address
Age

EMPLOYEE
Salary
Hire_date

ALUMNUS
sex_alumnus

DEGREE
Year
Degree
Major

STUDENT
MajorDept
change_major

GRADUATE_STUDENT
Degree_plan
change_degree_plan

UNDERGRADUATE_STUDENT
Class
change_classification

STAFF
Position
Rank

FACULTY
Manager

STUDENT_ASSISTANT

GRADUATE_STUDENT

RESEARCH_ASSISTANT
Project
change_project

TEACHING_ASSISTANT
Course
change_to_course