The Enhanced Entity Relationship (EER) Model:

The EER is achieved by incorporation of a **semantic data modeling** concepts into the conceptual ER Model. These semantic concepts are:

1. **Object oriented concepts**
   - Superclass & subclass relationship
   - Attribute & relationships inheritance
2. **The concept of specialization =>** Looking for the real world from different point of views
3. **The concept of categories** => Generation of a class which represents the union of entities of other classes.

Features of the superclass / subclass relationship concept on EER:

1. An entity in a subclass is related via the key attribute to its superclass entity.
2. An entity cannot exist in a DB by being a member of a subclass unless it is a member in superclass.
3. An entity may be a member in multiple subclasses, but it is not necessary that every entity in a superclass is a member in a subclass.
4. An entity that is a member of a subclass inherits all the attributes of its superclass and inherits its relationships as well.
5. A member entity of the subclass represents the same real-world entity in the related superclass but in a distinct specific role.

Constraint and Characteristics of *Specialization*:

Definition Constraints:

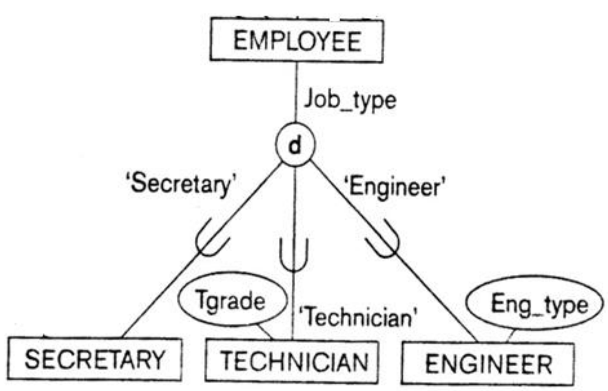
a. **Predicate defined specialization**: The process of defining a condition to determine exactly the entities that will become members of each subclass *by placing a condition on the value of some attribute* of the superclass, which is called the **defining attribute** of the related subclass.

b. **User defined specialization**: When we do not have any condition to determine membership in a subclass hence membership is specified individually for each entity by the user and not by any condition that can be evaluated automatically.

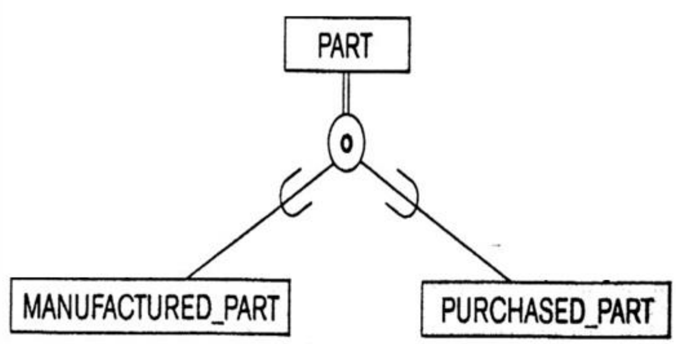
Disjoints Constraints:

a. **Disjoint specialization**: An entity can be a member of ***at most one subclass*** of a specialization.

b. **Overlapped specialization**: An entity can be a member ***in any number of subclasses*** of specialization



Disjoint Specialization



Overlapped specialization:

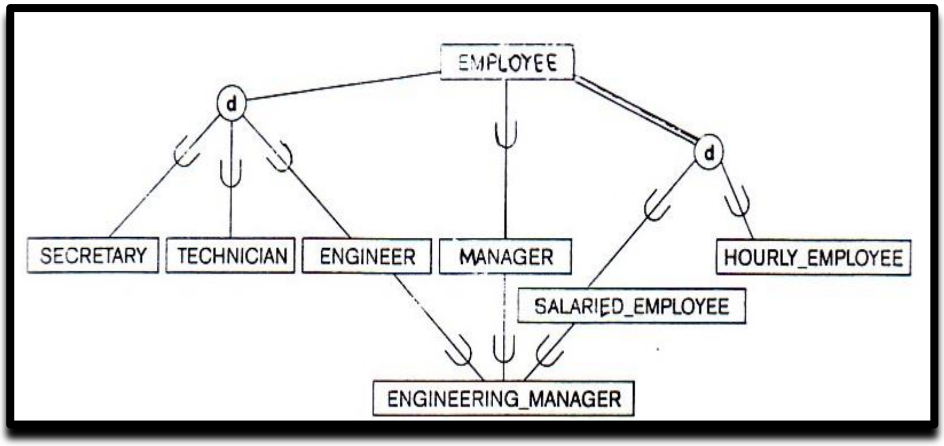
Participation Constraints:

a. **Total participation specialization**: Specifies that every entity ***in a superclass*** must be a member of ***at least one subclass*** in the specialization. Shown with a double line.

b. **Partial participation specialization**: Allows an entity ***in a superclass*** **not** to belong to any of its subclasses in the specialization. Shown with a single line.

Specialization *Hierarchies* and Lattices:

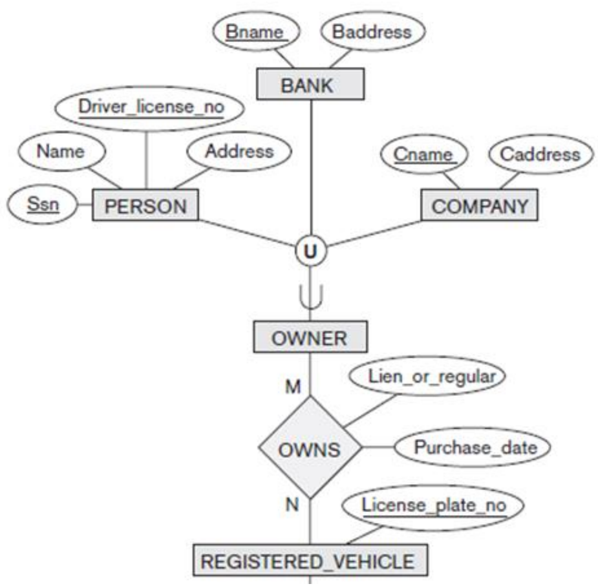
1. **Specialization Hierarchy (Tree Inheritance)**: The constraint that every subclass participates as a subclass in only subclass/class relationship.
2. **Specialization Lattice (Multiple Inheritance)**: The constraint that a subclass can be a subclass in more than one class/subclass relation.



- In specialization with lattice or hierarchy inheritance, a subclass inherits the attributes not only of its direct superclass but also of all its predecessor superclasses all the way to the root of the hierarchy or lattice.
- **Leaf Node Class:** it is a class that has no subclasses of its own.
- **Shared Subclass:** it is a subclass with more than one superclass and its entities represent a subset of the intersection of the entities of its superclasses. ***This means that an entity of the shared subclass must exist as an entity in all its superclasses.*** For the example above, the shared subclass ENGINEERING_MANAGEER means that an engineering manager must be an engineer, manager, and salaried_employee.

The concept of Category:

Category is a union type represented by a subclass that contains *a collection of real-world entities* (objects) which are a subset of the union of entity types. ***A category member must exist in at least one of its super classes.***



EER-to-Relational Mapping:

Here we are going to add further step to the ER-to-Relational mapping algorithm **(6 Steps)** to handle the mapping of specialization. This step will have 4-main options and conditions under which we can determine the suitable option. We use Attrs(R) to denote the attributes of relation R and PK(R) to denote the primary key of R.

Step 7: Options for mapping Specialization:

Convert each specialization with m subclasses {S1, S2, …, Sm} and superclass C, where the attributes of C are {k,a1,…,an} and K is the primary key, into table schemas using one of the following option.

A. **Option 7A Multiple relations_Superclass and Subclasses:-**

Create a table L for C with attributes (L)= $\{k,a_1,\ldots,a_n\}$ and PK(L)=k. Create a relation L_i for each subclass S_i, $1 \le i \le m$, with the attributes(L_i)= {k} U {attributes of S_i} and PK (L_i)=k. This option works for any specialization (total or partial, disjoint or overlapping).

B. **Option 7B Multiple relations-Subclass relation Only:-**

Create a table L_i for each subclass S_i, $1 \le i \le m$ with the Attributes (L_i) = {attributes of S_i} U $\{k, a_1, \ldots, a_n\}$ and PK(L_i) = k. This option only works for a specialization whose subclasses are total **(Why?).** If the specialization is overlapping; an entity may be duplicated in several relations. (**If the specialization is disjoint & total it will be optimal mapping).**

C. **Option 7C Single relation with one type Attribute:**

Create a single table L with attributes (L) = $\{k, a_1, \ldots, a_n\}$ U {attributes of S_1} U…U {attributes of S_m} U {t} and PK(L)=k. The attribute t is called a type (or discriminating) attribute that indicates the subclass to which each tuple belongs, if any. This Option works only for a specialization whose subclasses are **disjoint** and has the potential for generating many Null values if many specific attributes exist in a subclass.

D. **Option 7D: Single relation with multiple type attributes:**

Create a single table schema L with Attributes(L) =$\{k,a_1,\ldots,a_n\}$ U { attributes of S_1} U…U {attributes of S_m} U $\{t_1,t_2,\ldots,t_m\}$ and PK(L) = k.

Each t_i , $1 \le i \le m$, is a Boolean type attribute indicating whether a tuple belongs to subclass S_i .