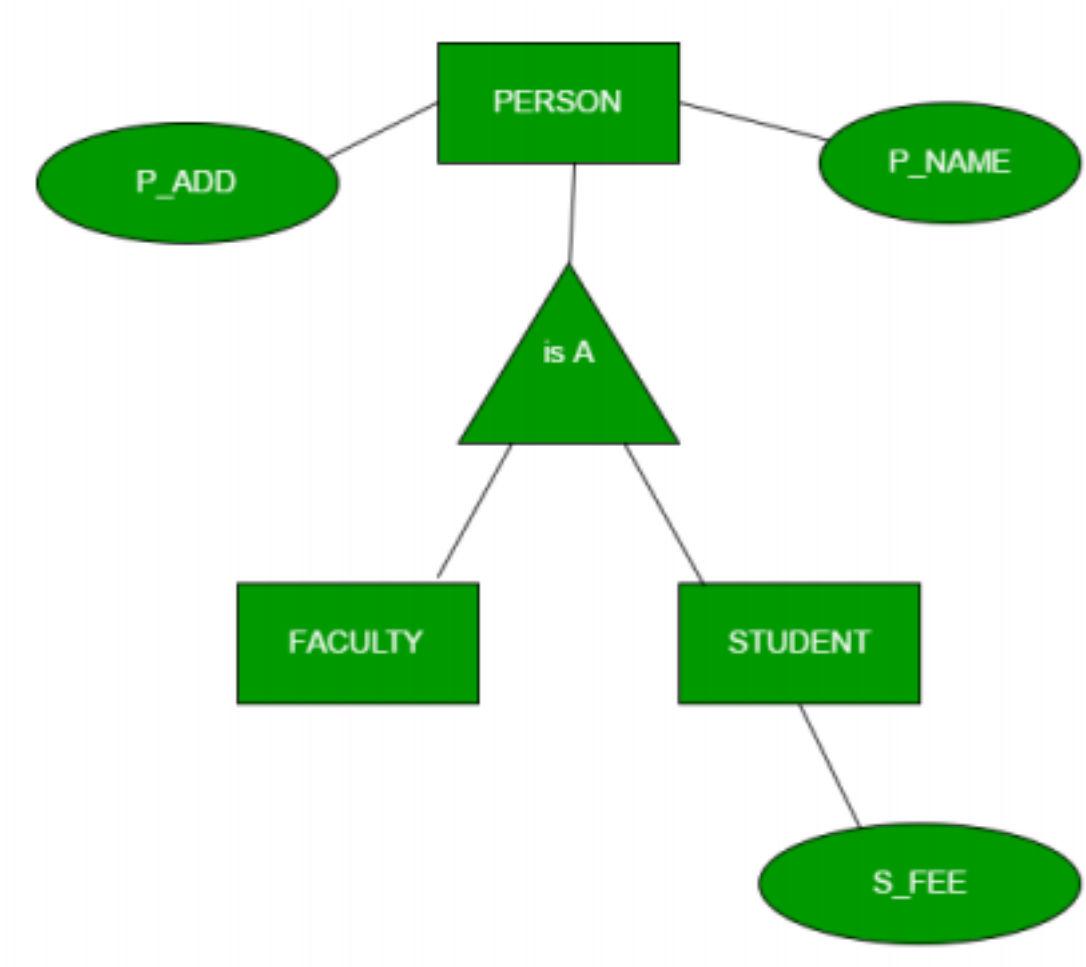
## Generalization, Specialization and Aggregation in ER Model

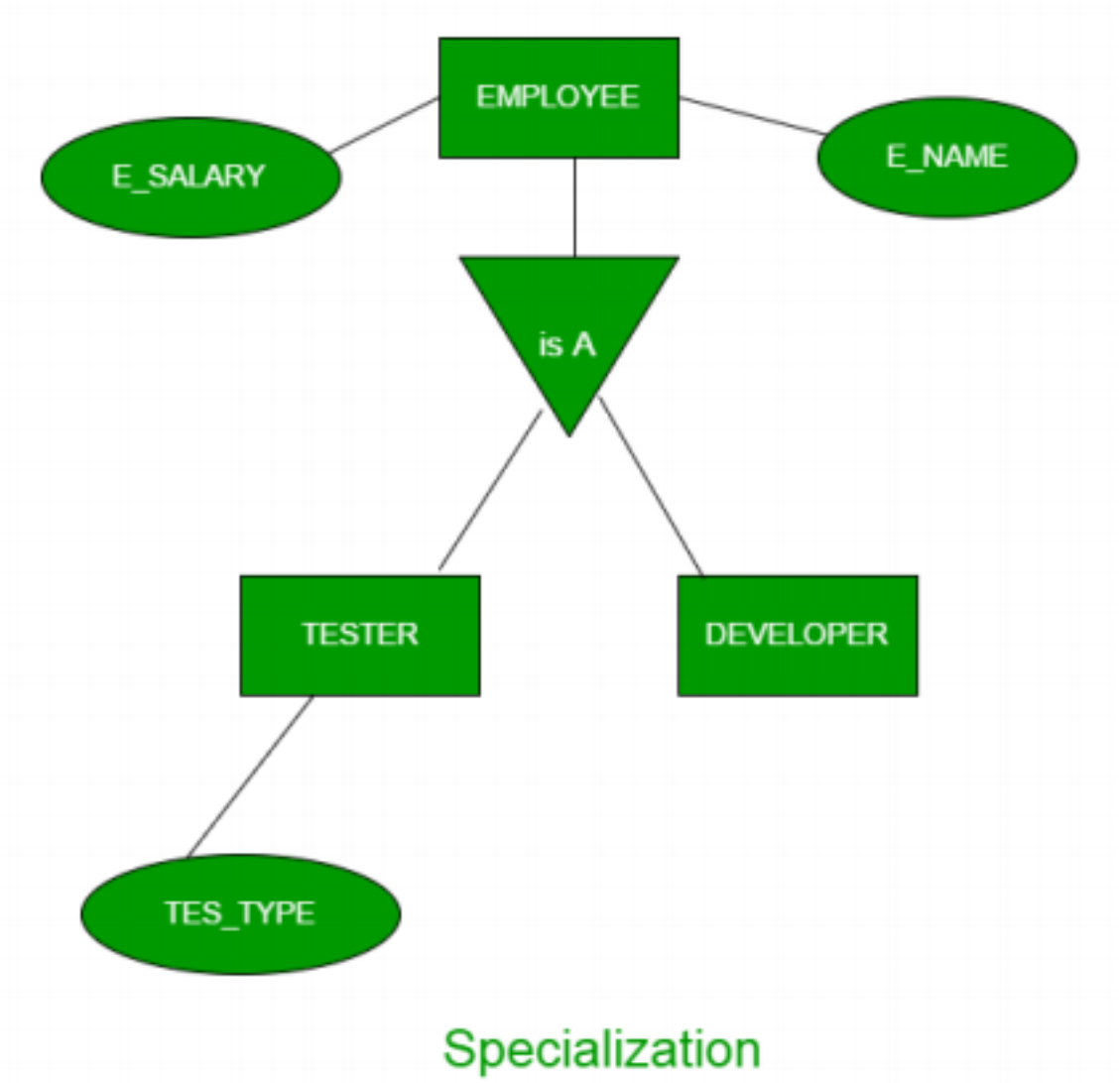
Generalization, Specialization and Aggregation in ER model are used for data abstraction in which abstraction mechanism is used to hide details of a set of objects.

❖ **Generalization –** Generalization is the process of extracting common properties from a set of entities and create a generalized entity from it. It is a bottom-up approach in which two or more entities can be generalized to a higher level entity if they have some attributes in common. For Example, STUDENT and FACULTY can be generalized to a higher level entity called PERSON as shown in Figure 1. In this case, common attributes like P_NAME, P_ADD become part of higher entity (PERSON) and specialized attributes like S_FEE become part of specialized entity(STUDENT).




❖ **Specialization –** In specialization, an entity is divided into sub-entities

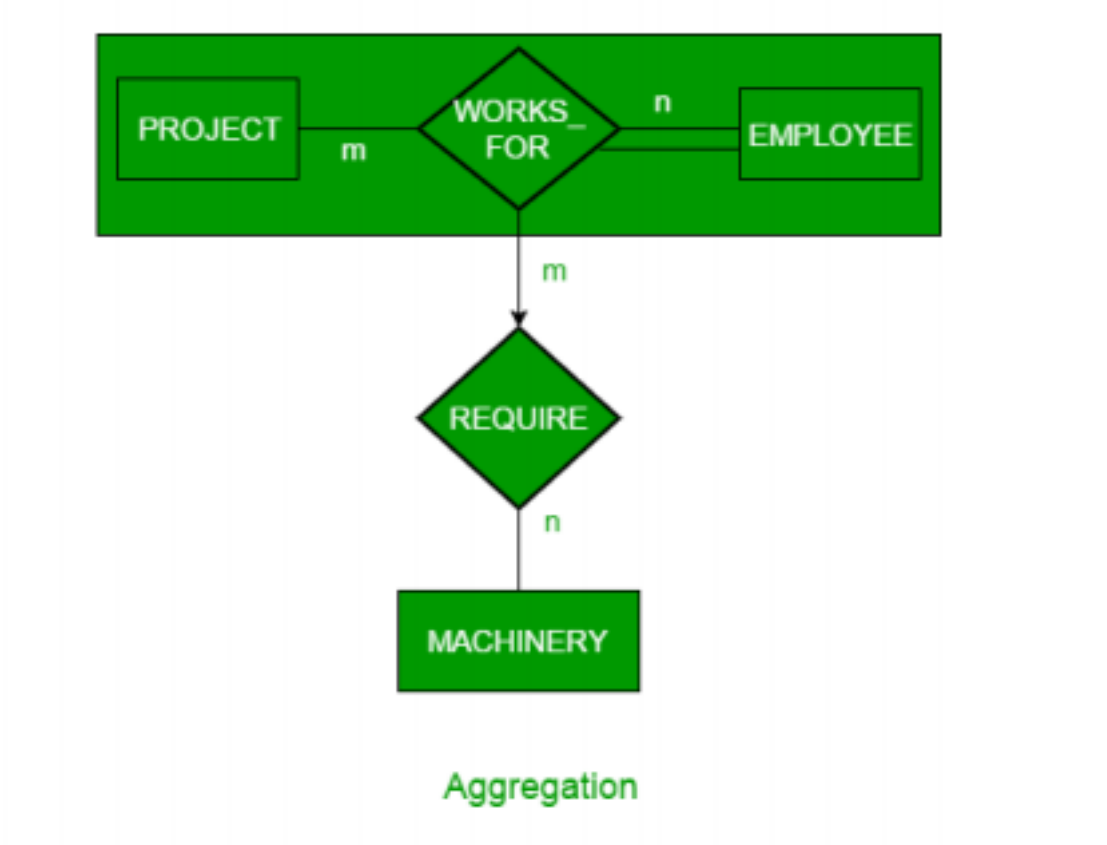
based on their characteristics. It is a top-down approach where higher level entity is specialized into two or more lower level entities. For Example, EMPLOYEE entity in an Employee management system can be specialized into DEVELOPER, TESTER etc. as shown in Figure 2. In this case, common attributes like E_NAME, E_SAL etc. become part of higher entity (EMPLOYEE) and specialized attributes like TES_TYPE become part of specialized entity (TESTER).




Specialization

- **Aggregation –** An ER diagram is not capable of representing relationship between an entity and a relationship which may be required in some scenarios. In those cases, a relationship with its corresponding entities is aggregated into a higher level entity. Aggregation is an abstraction through which we can represent relationships as higher level entity sets.

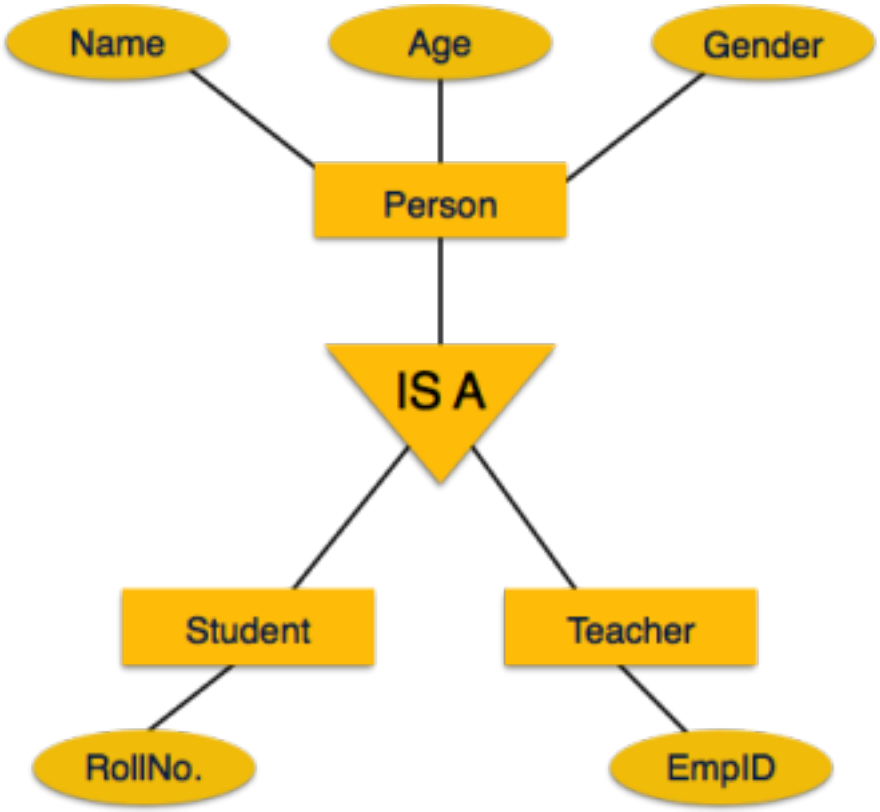
For Example, Employee working for a project may require some machinery. So, REQUIRE relationship is needed between relationship WORKS_FOR and entity MACHINERY. Using aggregation, WORKS_FOR relationship with its entities EMPLOYEE and PROJECT is aggregated into single entity and relationship REQUIRE is created between aggregated entity and MACHINERY.




❖ **Inheritance**

The details of entities are generally hidden from the user; this process known as **abstraction**.

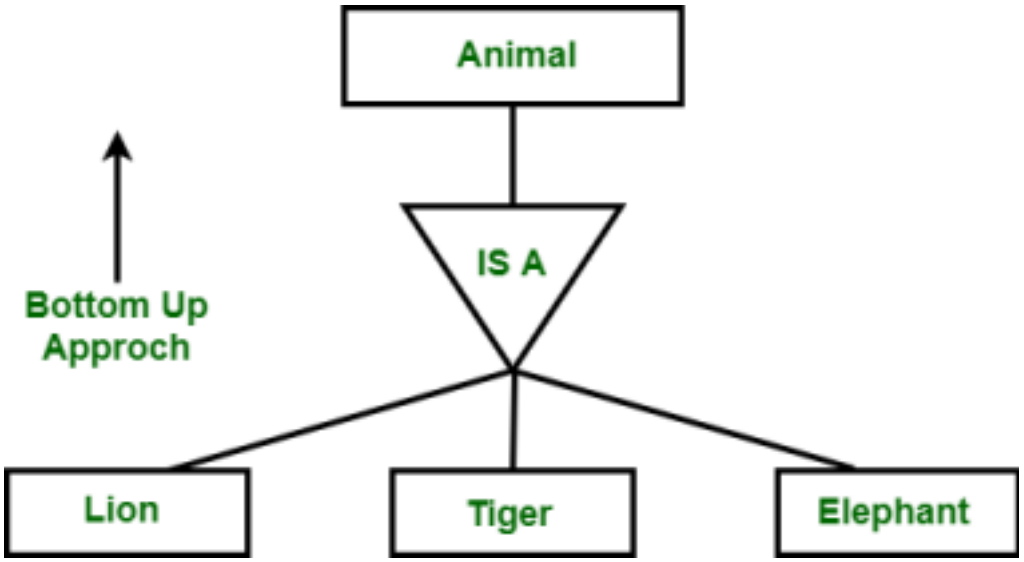
Inheritance is an important feature of Generalization and Specialization. It allows lower-level entities to inherit the attributes of higher-level entities.




For example, the attributes of a Person class such as name, age, and gender can be inherited by lower-level entities such as Student or Teacher.

❖ **Constraints on Generalization**

To model an enterprise more accurately, there are some constraints that are applicable onto the database on the particular generalization.




*Generalization*

There are three types of constraints on generalization which are as follows:

1. First one determines which entity can be a member of the low-level entity set.
2. Second relates to whether or not entities belong to more than one lower level entity set.
3. Third specifies whether an entity in the higher level entity set must belong to at least one of the lower level entity set within generalization.

**1. First one determines which entity can be a member of the low-level entity set:**

Such kind of membership may be one of the following:

- **Condition-defined** – In this lower-level entity sets, evaluation of membership is on basis whether an entity satisfies an explicit condition or not. For example, Let us assume, higher-level entity set student which has attribute student type. All the entities of student are evaluated by definition of attribute of student. Entities are accepted by the satisfaction of condition i.e. student type = "graduate" then only they are allowed to belong to lower-level entity set i.e. graduate student. By the satisfaction of condition student type = "undergraduate" then they are included in undergraduate student. In fact, all the lower-level entities are evaluated on the basis of the same attribute, thus it is also referred as attribute-defined.
- **User-defined** – In this lower-level entity sets are not get constrained by a condition named membership; users of database assigns entities to a given entity set. For example, Consider a situation where after 3 months of employment, the employees of the university are assigned to one of four work teams. For this purpose, we represent teams them as four lower-level entity sets of higher-level employee entity set. On the basis of an explicit defining condition, a given employee is not assigned to specific team entity. User in charge of this decision makes the team assignment on an individual basis. By adding entity to an entity set, assignment is implemented.

**2. Second relates to whether or not entities belong to more than one lower level entity set:**

Following is one of the lower-level entity sets:

- **Disjoint** – The requirement of this constraint is that an entity should not

belong to no more than one lower-level entity set. For example, the entity of student entity satisfy only one condition for student type attribute i.e. Either an entity can be a graduate or an undergraduate student, but cannot be both at the same time.

- **Overlapping –** In this category of generalizations, within a single generalization, the same entity may belong to more than one lower-level entity set. For example, in the employee work-team assume that certain employees participate in more than one work team. Thus, it offers a given employee that he may appear in more than one of the team entity sets that are lower-level entity sets of employee. Thus, generalization is overlapping.

**3. Third specifies whether pr not an entity in the higher level entity set must belong to at least one of the lower level entity set within generalization :**

This constraint may be one of the following:

- **Total generalization or specialization –**

  According to this constraint, each higher-level entity must belong to a lower-level entity set.

- **Partial generalization or specialization –**
  According to this constraint, some higher-level entities may not belong to any lower-level entity set.