Some Terminologies related to ER models

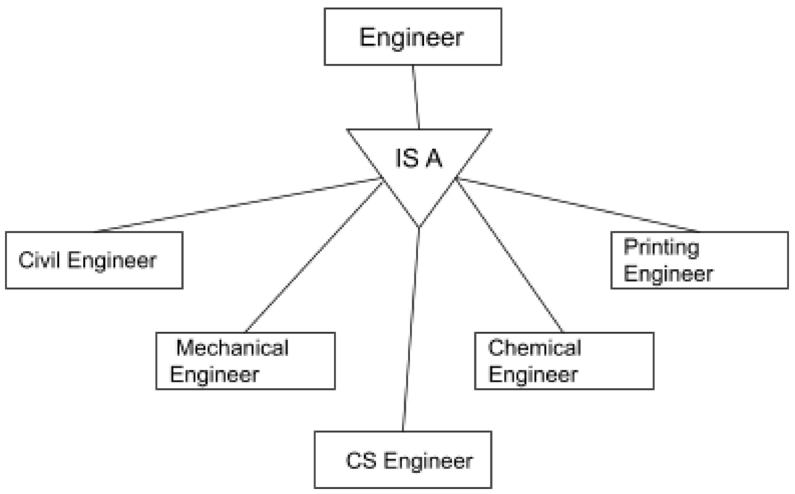
When complexity of data increases, it becomes harder to use the traditional ER model for data modeling.
So, there are some enhancements in the traditional ER model to make it able to handle complex applications better. We also may call it Enhanced ER-model.
These are -

- Specialization
- Generalization
- Aggregation

1. Specialization

In layman language, Specialization is a process of mastering a specific domain (refer to example given below).
But, with respect to ER Model, specialization is the procedure to splitting up the entities into further sub entities on the basis of their functionalities, specialities and features.
These sub-designation of entities are distinctive from other entities in the set.

To define it in a more technical way, Specialisation is a process which follows a Top-to-Down approach.
In this a higher-level entity is specialized in two or more low level entities. (we design sub-entities for a given entity)



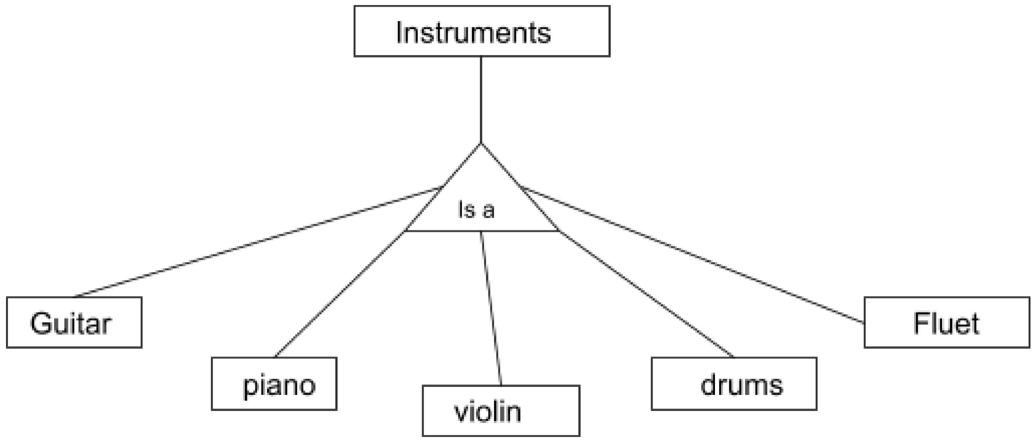
Initially there was an Engineer named single entity. But sometimes we don’t need all the engineers present in the world for necessary work. We need Someone specialized in a specific domain.For example, a CS engineer can not be asked to build a bridge or building. So Engineers can be further divided into Civil Engineer, Mechanical Engineer, CS engineer, Chemical Engineer, Printing Engineer and many more.

Why Specialization?
One of the features of Specialization in a data model is that certain attributes may only be applicable to a few entities of the Parent Set but not to all. In order to group the entities to which these attributes are applicable, a sub-group is defined. A large number of the attributes of the members of the sub-group may still be shared with other members of the Parent Entity set.

Note: In Specialisation, schema size increases.

2. Generalization

In simple terms, Generalization is the reverse process of specialization. In generalization, the sub entities are combined together resulting in the formation of a parent entity set on the basis of some common features in such a way that the new entity thus formed contains all the features of the sub entities.
Generalization is a process which follows a Bottom-to-Up approach.
So to define it in technical terms, two or more low-level entities are synthesised to make higher-level entity on the basis of common attributes.
This process involves establishing relationship among tables, by analyzing the attributes belonging to the entities and then making an informed decision to form a high level entity.
The attributes for this high level entity are those attributes which are believed to be most required for each child-entity.



In the diagram above entities like guitar, piano, violin, drums, flute are generalized into a single parent entity, named Instrument.
This generalization could happen because of the common features of all the entities.

Why Generalization?
It optimizes the database, by making it simpler.
Data repetition is also avoided, as now there won't be no need to mention the same attributes or data in different tables/relation.

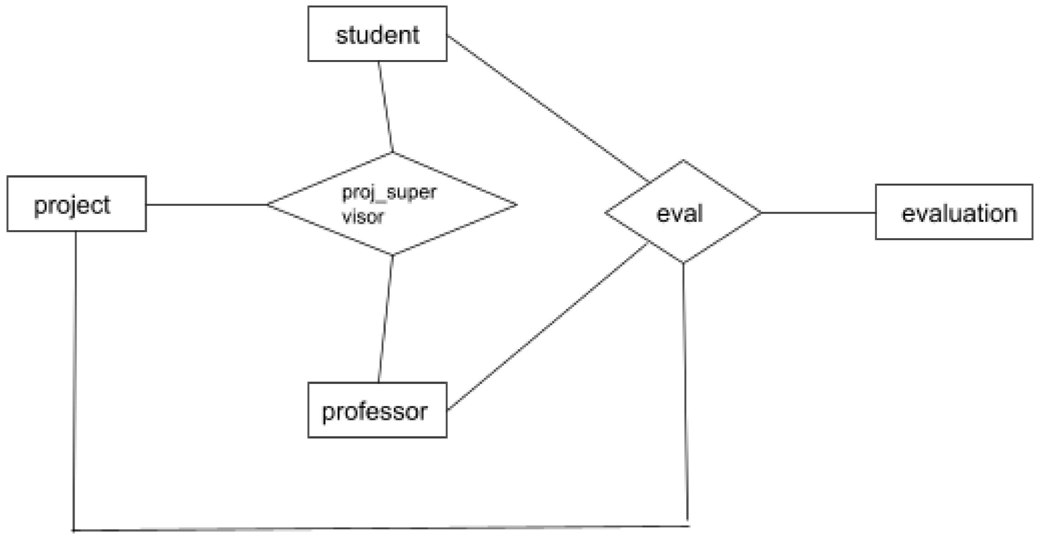
Note: In Generalization, schema size shrinks.

* Both Generalisation and Specialization follow attribute inheritance (i.e. The attributes of the higher-level entity sets are said to be inherited by the lower-level entity sets, similarly vice-versa), they also follow participation inheritance (i.e. if a parent entity participates in a relationship set then its sub-entities will also participate in that relationship set.)

3. Aggregation

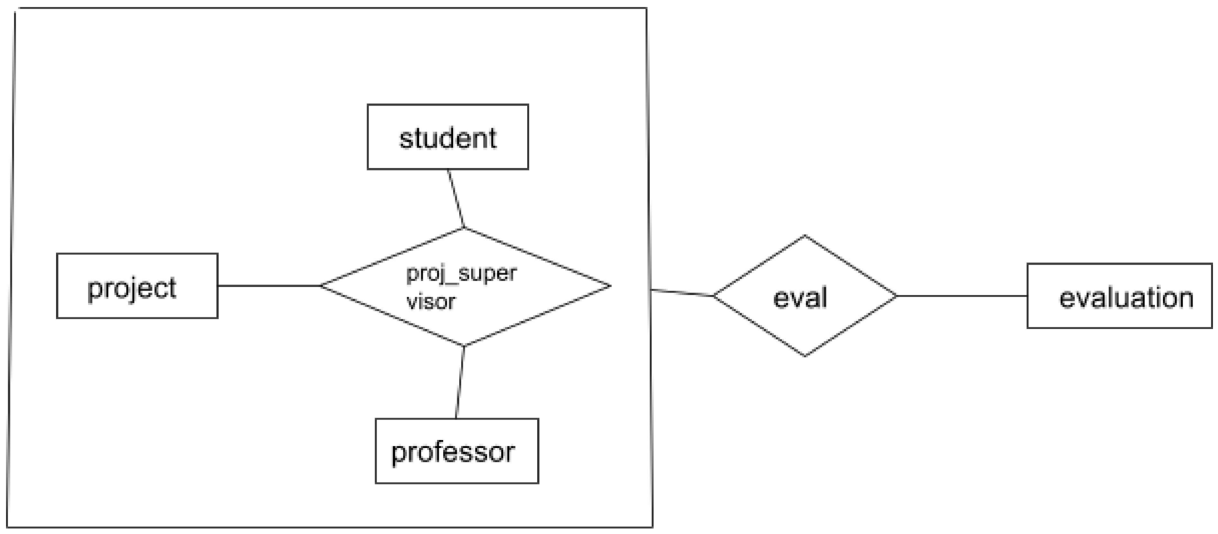
Aggregation is used when we need to express a relationship among relationships. It is like abstraction through which relationships are treated as higher-level entities.
In this multiple entities are considered as a single entity and again this single entity has a relationship with another entity. It treats relationships as an abstract entity.

Let’s understand with an example:
In our college times, many of us take up a project under a professor during our summer time or whenever an individual wants, so let’s Consider a ternary relationship proj_supervisor, suppose now the instructor/professor needs to evaluate the projects students have been doing under them.
So we record those evaluations.
Here’s an ER-Model for this:-



Relationship sets eval and proj_supervisor represent overlapping information. Every eval relationship corresponds to a proj_supervisor a relationship that is evaluation can only happen if a student is allocated a project under some professor.
However, some proj_supervisor relationships may not correspond to any eval relationships.
I.e. there might not be any student to evaluate, though that student has been allocated a project under some professor. So we can’t discard the proj_supervisor relationship.
Now, We need to have a relationship between relationships.
For solving this problem, we use aggregation.
As, A student is supervised by a particular professor on a particular project. A student, professor, project combination may have an associated evaluation.
Basically, we treat their combination as one entity.

Enhanced ER-model after aggregation:-



Why Aggregation?

Due to ER Model's limitation to represent relationships among relationships. Although if we try to represent ternary relationships using ER-diagram it might result in a lot of redundancies.

Hence, aggregation is used which is redundancy free.