Mapping EER diagrams to Relational Schema

1. We represent an entity as a relation. Essentially we represent a relation as a table with name as the name of the entity and columns of the table as represented by the attributes of the entity.
2. When the attributes are composite we directly use the individual or atomic attribute. For example if there is a Name attribute composed of First_name and Last_name then we don't show Name attribute in the table, we directly use First_name and Last_name in the relation.
3. When we have an attribute which can take multiple values, then we create another relation which takes the primary key of the entity as its own primary key. We add a column for the attributes. The primary key also acts as a foreign key to refer to the parent entity.
4. For an entity we define a primary key which is the minimum of collection attributes which always have a unique value in the table (Uniqueness Constraint). For defining relations between tables we mention a foreign key which refers to a relation and is governed by the referential integrity constraints.
5. The referential constraints are as follows:
    a. The domain of attributes in the foreign key must be the same as the domain of the primary key of the referred relation.
    b. The value of foreign key has the values that are in the primary key of the referred relation for some tuple. It can also have a NULL values.
6. For a weak entity type, we make the relation using all its attributes, but we also include the primary key of the owner entity. The primary key of the weak entity is a combination of primary key of the identifying entity and the weak entity key. The primary key of the owner entity also works as a foreign key which refers to the owner entity.
7. For subclasses we can use a variety of ways to represent them in the relational schema based on their disjoint constraints. For our design we have used the following methods.
    a. We represent the super entity using the conventional method and for every subclass entity we use their attributes combined with the primary key of the superclass entity. This primary acts as a primary key of the subclasses and also as a foreign key to refer to the superclass entity.
    b. We can represent the subclasses only without the need to represent a separate relation for superclass. The way we do it is that we create relations for each subclass where we include the attributes of the superclass and the attributes of the corresponding subclass. The primary key of each subclass is the primary key of the superclass. This method is used when we have total participation from the superclass and disjointed constraint is also met.
    c. When we have disjoint subclasses we can go a step further and have a single relation which comprises super class and all the subclasses. We combine all the attributes of the super class and all the subclasses and add a type attribute. The values of the type attribute can be used to specify which subclass is being referred. The primary key of the super class is the primary key of the relation.

8. Representing relations: For a 1:1 and 1:N relationship we give the primary key of one relation to the other as a foreign key. Let’s say we have two entities A and B which have a 1:1 relation among them we give the primary key of B to A as a foreign key also any attributes of the relation are given to the relation A. We usually give the keys and attributes to the relation which has total participation.
9. For M:N relations, we make a new table which has the primary keys from all the participating relations. The primary keys from all the relations combined act as the primary key of the new relation. These keys also acts as foreign key to refer to their respective relations.
10. For a relation with degree > 2 we follow the same steps followed to represent the M:N relation.
11. When we have a union of two entities, we give the union entity a surrogate key along with it's attributes in it's table. We give this surrogate key to all the participating entities, this works as a foreign key for them. This process is required only when the entities being unioned have no common attribute, otherwise we can directly refer that attribute.