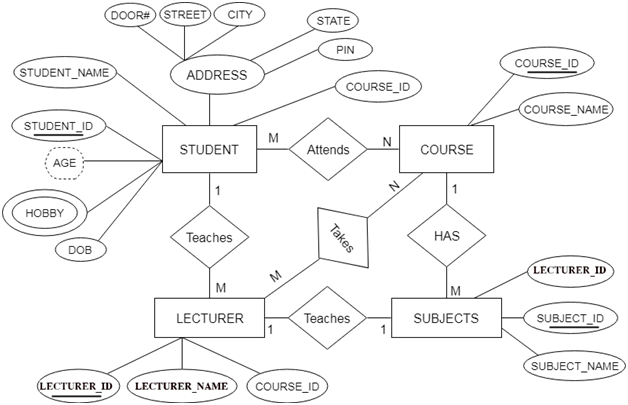
**Reduction of ER diagram to Table**

The database can be represented using the notations, and these notations can be reduced to a collection of tables.

In the database, every entity set or relationship set can be represented in tabular form.

The ER diagram is given below:



There are some points for converting the ER diagram to the table:

* **Entity type becomes a table.**

All the entities represented in the rectangular box in the ER diagram become independent tables in the database. In the given ER diagram, LECTURE, STUDENT, SUBJECT and COURSE forms individual tables.

* **All single-valued attribute becomes a column for the table.**

All the attributes, whose value at any instance of time is unique, are considered as columns of that table. In the STUDENT entity, STUDENT\_NAME and STUDENT\_ID form the column of STUDENT table. Similarly, COURSE\_NAME and COURSE\_ID form the column of COURSE table and so on.

* **A key attribute of the entity type represented by the primary key.**

In the given ER diagram, COURSE\_ID, STUDENT\_ID, SUBJECT\_ID, and LECTURE\_ID are the key attribute of the entity.

* **The multivalued attribute is represented by a separate table.**

In the student table, a hobby is a multivalued attribute. So, it is not possible to represent multiple values in a single column of STUDENT table. Hence, we create a table STUD\_HOBBY with column name STUDENT\_ID and HOBBY. Using both the column, we create a composite key.

* **Composite attribute represented by components (Any composite attributes are merged into same table as different columns.).**

In the given ER diagram, student address is a composite attribute. It contains CITY, PIN,

DOOR#, STREET, and STATE. In the STUDENT table, these attributes can merge as an

individual column.

* **Derived attributes are not considered in the table.**

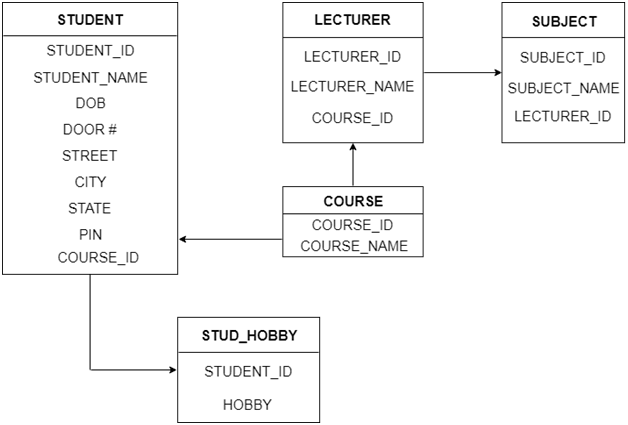
In the STUDENT table, Age is the derived attribute. It can be calculated at any point of time

by calculating the difference between current date and Date of Birth.

* **Declare the foreign key column, if applicable.**

In the diagram, attribute COURSE\_ID in the STUDENT entity is from COURSE entity. Hence add COURSE\_ID in the STUDENT table and assign it foreign key constraint. COURSE\_ID and SUBJECT\_ID in LECTURER table forms the foreign key column. Hence by declaring the foreign key constraints, mapping between the tables is established.

Using these rules, you can convert the ER diagram to tables and columns and assign the mapping between the tables. Table structure for the given ER diagram is as below:

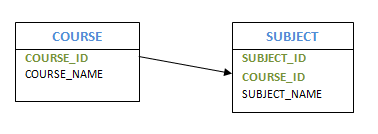


Let us see some of the special cases.

* **Converting Weak Entity**

Weak entity is also represented as table. All the attributes of the weak entity form the column of the table. But the key attribute represented in the diagram cannot form the primary key of this table. We have to add a foreign key column, which would be the primary key column of its strong entity. This foreign key column along with its key attribute column forms the primary key of the table.

In our example above, SUBJECTS is the weak entity. Hence, we create a table for it. Its attributes SUBJECT\_ID and SUBJECT\_NAME forms the column of this table. Although SUBJECT\_ID is represented as key attribute in the diagram, it cannot be considered as primary key. In order to add primary key to the column, we have to find the foreign key first. COURSE is the strong entity related to SUBJECT. Hence the primary key COURSE\_ID of COURSE is added to SUBJECT table as foreign key. Now we can create a composite primary key out of COURSE\_ID and SUBJECT\_ID.

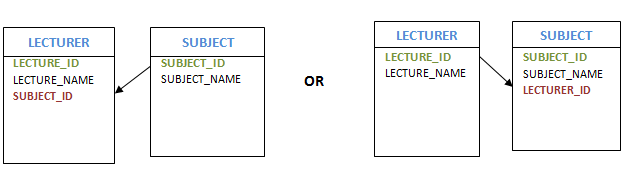


* **Representing 1:1 relationship**

Imagine SUBJECT is not a weak entity, and we have LECTURER teaches SUBJECT relation. It is a 1:1 relation. i.e.; one lecturer teaches only one subject. We can represent this case in two ways

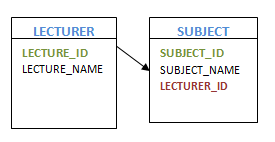
* Create table for both LECTURER and SUBJECT. Add the primary key of LECTURER in SUBJECT table as foreign key. It implies the lecturer’s name for that particular subject.
* Create table for both LECTURER and SUBJECT. Add the primary key of SUBJECT in LECTURER table as foreign key. It implies the subject taught by the lecturer.

In both the case, meaning is same. Foreign key column can be added in either of the table, depending on the developer’s choice.



* **Representing 1: N relationship**

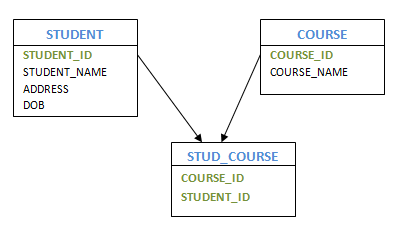
Consider SUBJECT and LECTURER relation, where each Lecturer teaches multiple subjects. This is a 1: N relation. In this case, primary key of LECTURER table is added to the SUBJECT table. i.e.; the primary key at 1 cardinality entity is added as foreign key to N cardinality entity



* **Representing M: N relationship**

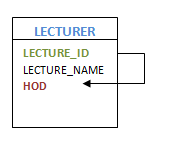
Consider the example, multiple students enrolled for multiple courses, which is M: N relation. In this case, we create STUDENT and COURSE tables for the entities. Create one more table for the relation ‘Enrolment’ and name it as STUD\_COURSE. Add the primary keys of COURSE and STUDENT into it, which forms the composite primary key of the new table.

That is, in this case both the participating entities are converted into tables, and a new table is created for the relation between them. Primary keys of entity tables are added into new table to form the composite primary key. We can add any additional columns, if present as attribute of the relation in ER diagram.



* **Self-Referencing 1: N relation**

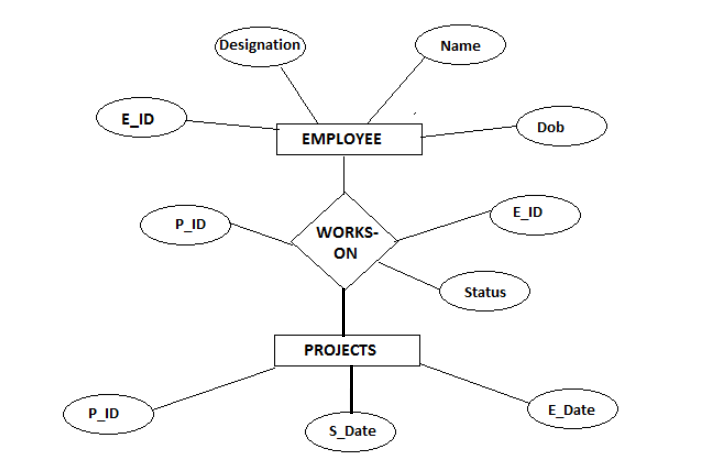
Consider the example of HOD and Lecturers. Here one of the Lecturers is a HOD of the department. i.e.; one HOD has multiple lecturers working with him. In this case, we create LECTURER table for the Lecturer entity. Create the columns and primary keys as usual. In order to represent HOD, we add one more column to LECTURER table which is same column as primary key, but acts as a foreign key. i.e.; LECTURER\_ID is the primary key of LECTURER table. We add one more column HOD, which will have LECTURER\_ID of the HOD. Hence LECTURER table will show HOD’s Lecturer ID for each Lecturer. In this case, primary key column acts as a foreign key in the same table.



**Thumb Rules to Remember**

While determining the minimum number of tables required for binary relationships with given cardinality ratios, following thumb rules must be kept in mind-

* For binary relationship with cardinality ration m: n, separate and individual tables will be drawn for each entity set and relationship.
* For binary relationship with cardinality ratio either m: 1 or 1: n , always remember “many side will consume the relationship” i.e. a combined table will be drawn for many side entity set and relationship set.
* For binary relationship with cardinality ratio 1: 1 , two tables will be required. You can combine the relationship set with any one of the entities sets.
* **Example**



Now for the above example we can create three relations-

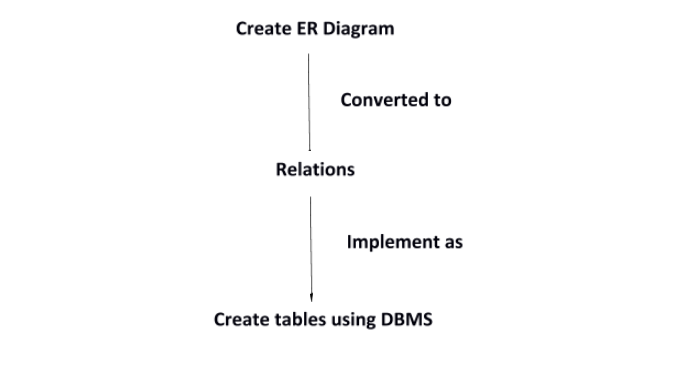
* Employee
* Works\_On
* Projects

Transform attributes to fields-

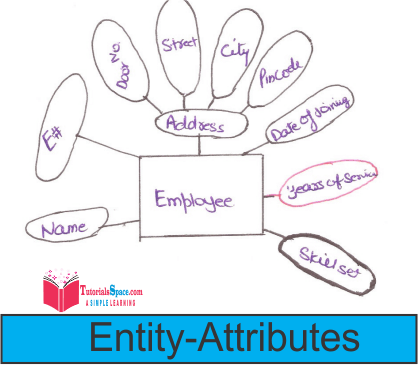
* Employee will have E\_ID, Name, Designation and Dob.
* Works\_On will have E\_ID, Status and P\_ID.
* Projects will have P\_ID, S\_Date and E\_Date.

Now we can create tables in DBMS.

Overall transformation summary is −



* **Example**

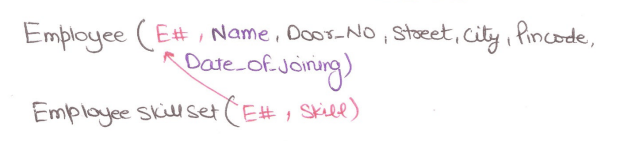


* Here Address is Composite Attribute.
* Years-Of-Service is Derived Attribute
* Skillset is a Multivalued Attribute: **We have to make separate table but remember** there should be relationship between an employee table and skill set table.

So, the Relational Schema and Employee Skill set table is:>

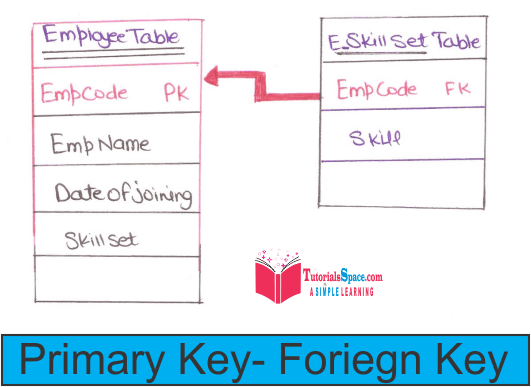
**Employee**(**E#,** **Name, Door\_No, Street, city, pincode, date\_of\_joining)**

**Employeeskillset**(**E#, Skill**)



As we saw E# has to be primary key for employee table to uniquely identify an entity and employee skill set has skill which came from skill set (multi valued).

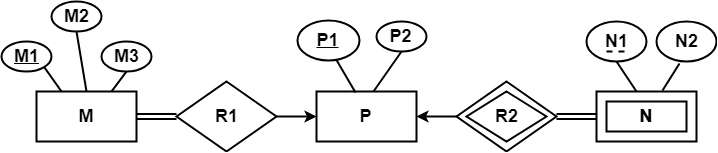
So, to create a relationship between these two tables we use E# i.e., Employee Code in **EmployeeSkillSet** table as a **"foreign key"** which implements **Referential Integrity.**



* **Example**

**Problem-01:**

Find the minimum number of tables required for the following ER diagram in relational model-



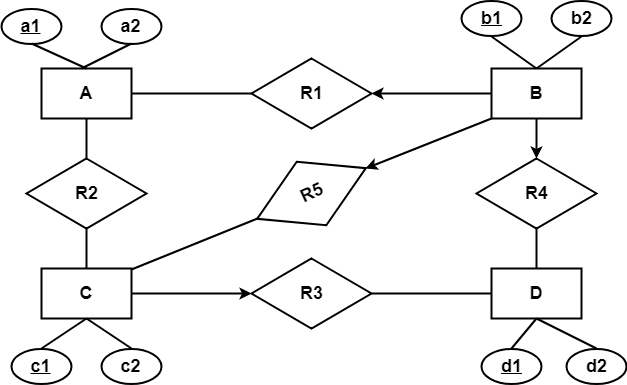
**Solution-**

Applying the rules, minimum 3 tables will be required-

* MR1 (M1 , M2 , M3 , P1)
* P (P1 , P2)
* NR2 (P1 , N1 , N2)

**Problem-02:**

Find the minimum number of tables required to represent the given ER diagram in relational model-



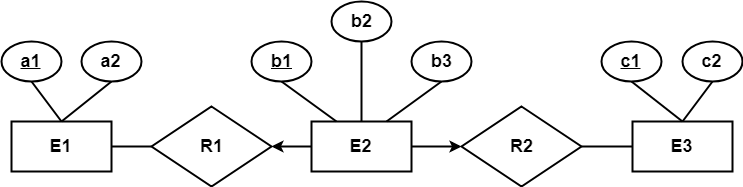
**Solution-**

Applying the rules, minimum 5 tables will be required-

* BR1R4R5 (b1 , b2 , a1 , c1 , d1)
* A (a1 , a2)
* R2 (a1 , c1)
* CR3 (c1 , c2 , d1)
* D (d1 , d2)

**Problem-03:**

Find the minimum number of tables required to represent the given ER diagram in relational model-



**Solution-**

Applying the rules, minimum 3 tables will be required-

* E1 (a1 , a2)
* E2R1R2 (b1 , b2 , a1 , c1 , b3)
* E3 (c1 , c2)