Introduction

ER Data Model is based on the real world objects and their relationship. In other words, each and everything, either living or non-living things in this world forms the object in database world. We identify all the required objects for our database requirement and give the shape of database objects. Before putting them into database, it is very much essential to understand the requirement properly and design them efficiently. It is like a foundation of the building. For this purpose, we use ER diagrams where we plan the database pictorially. ER diagram basically breaks requirement into entities, attributes and relationship. Let us see them in detail.

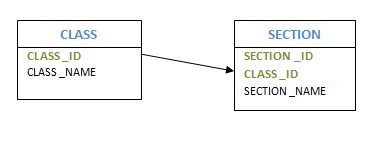
Entity

In a database, we would be grouping only related data together and storing them under one group name called Entity / Table. This helps in identifying which data is stored where and under what name. It reduces the time to search for a particular data in a whole database. Say, we are creating a School database. Then what all things come into our mind? It is school, students, teachers, subjects, class etc. If you observe what we have listed are items/names which are part of school, but each of them having their own group of related information? i.e.; Student has ID, name, address, DOB, class in which he is studying etc. for a Student. Similarly, TEACHERS will have their ID, name, address subjects that they are teaching, class which they are teaching etc. Hence each entity forms a group of related information. In real world each and every object forms an entity.

In short, all the living and non-living things in the world form entity. One can consider all nouns as entities, if that makes it simpler.

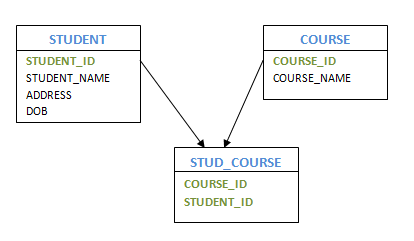
* **Strong Entity**: Entities having its own attribute as primary keys are called strong entity. For example, STUDENT has STUDENT\_ID as primary key. Hence it is a strong entity.
* **Weak Entity**: Entities which cannot form their own attribute as primary key are known weak entities. These entities will derive their primary keys from the combination of its attribute and primary key from its mapping entity.

Consider CLASS and SECTION entity. The SECTION has SECTION \_ID and NAME as its attribute. But SECTION \_ID alone cannot be a primary key, since it fails to tell for which course it is related to. We will not be uniquely identifying the course section by this attribute alone. But if this attribute along with CLASS\_ID gives the meaning for each section and we can uniquely identify the sections.

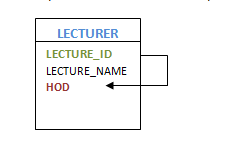


* **Composite Entity**: Entities participating in the many to many relationships are called composite entity. In this case, apart from two entities that are part of relation, we will one more hidden entity in the relation. We will be creating a new entity with the relation, and create a primary key by using the primary keys of other two entities.

Consider the example, multiple students enrolled for multiple courses. In this case, we create STUDENT and COURSE. Then we 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.



* **Recursive Entity**: If a relation exists between the same entities, then such entities are called as recursive entity. For example, mapping between manager and employee is recursive entity. Here manager is mapped to the same entity Employee. HOD of the department is another example of having recursive entity.



Attribute

In the above example of STUDENT entity, Student has ID, name, address, DOB, class in which he is studying are all called as attribute of STUDENT entity. It's also known as columns of the table. In other words, an attribute is a list of all related information of an entity, which has valid value.

An attribute can have single value or multiple value or range of values. In addition, each attribute can contain certain type of data like only numeric value, or only alphabets, or combination of both, or date or negative or positive values etc. Depending on the values that an attribute can take, it is divided into different types.

* **Simple Attribute**

These kinds of attributes have values which cannot be divided further. For example, STUDENT\_ID attribute which cannot be divided further. Passport Number is unique value and it cannot be divided.

* **Composite Attribute**

This kind of attribute can be divided further to more than one simple attribute. For example, address of a person. Here address can be further divided as Door#, street, city, state and pin which are simple attributes.

* **Derived Attribute**

Derived attributes are the one whose value can be obtained from other attributes of entities in the database.For example, Age of a person can be obtained from date of birth and current date. Average salary, annual salary, total marks of a student etc are few examples of derived attribute.

* **Stored Attribute**

The attribute which gives the value to get the derived attribute are called Stored Attribute. In example above, age is derived using Date of Birth. Hence Date of Birth is a stored attribute.

* **Single Valued Attribute**

These attributes will have only one value. For example, EMPLOYEE\_ID, passport#, driving license#, SSN etc have only single value for a person.

* **Multi-Valued Attribute**

These attribute can have more than one value at any point of time. Manager can have more than one employee working for him, a person can have more than one email address, and more than one house etc is the examples.

* **Simple Single Valued Attribute**

This is the combination of above four types of attributes. An attribute can have single value at any point of time, which cannot be divided further. For example, EMPLOYEE\_ID – it is single value as well as it cannot be divided further.

* **Simple Multi-Valued Attribute**

Phone number of a person, which is simple as well as he can have multiple phone numbers is an example of this attribute.

* **Composite Single Valued Attribute**

Date of Birth can be a composite single valued attribute. Any person can have only one DOB and it can be further divided into date, month and year attributes.

* **Composite Multi-Valued Attribute**

Shop address which is located two different locations can be considered as example of this attribute.

* **Descriptive Attribute**

Attributes of the relationship is called descriptive attribute. For example, employee works for department. Here ‘works for’ is the relation between employee and department entities. The relation ‘works for’ can have attribute DATE\_OF\_JOIN which is a descriptive attribute.

Keys

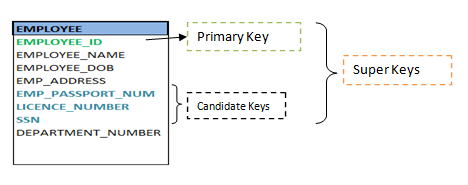
Keys are the attributes of the entity, which uniquely identifies the record of the entity. For example STUDENT\_ID identifies individual students, passport#, license # etc.

As we have seen already, there are different types of keys in the database.

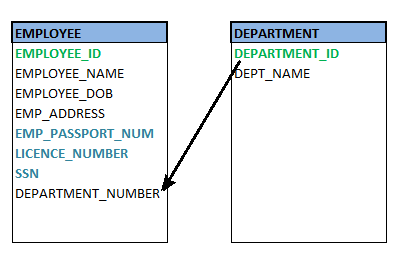
**Super Key** is the one or more attributes of the entity, which uniquely identifies the record in the database.

**Candidate Key** is one or more set of keys of the entity. For a person entity, his SSN, passport#, license# etc can be a super key.

**Primary Key** is the candidate key, which will be used to uniquely identify a record by the query. Though a person can be identified using his SSN, passport# or license#, one can choose any one of them as primary key to uniquely identify a person. Rest of them will act as a candidate key.



**Foreign Key** of the entity attribute in the entity which is the primary key of the related entity. Foreign key helps to establish the mapping between two or more entities.



Relationship

A relationship defines how two or more entities are inter-related. For example, STUDENT and CLASS entities are related as 'Student X **studies** in a Class Y'. Here 'Studies' defines the relationship between Student and Class. Similarly, Teacher and Subject are related as 'Teacher A **teaches** Subject B'. Here 'teaches' forms the relationship between both Teacher and Subject.

Degrees of Relationship

In a relationship two or more number of entities can participate. The number of entities who are part of a particular relationship is called degrees of relationship. If only two entities participate in the mapping, then degree of relation is 2 or binary. If three entities are involved, then degree of relation is 3 or ternary. If more than 3 entities are involved then the degree of relation is called n-degree or n-nary.

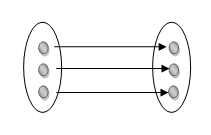
Cardinality of Relationship

How many number of instances of one entity is mapped to how many number of instances of another entity is known as cardinality of a relationship. In a ‘studies’ relationship above, what we observe is only one Student X is studying in on Class Y. i.e.; single instance of entity student mapped to a single instance of entity Class. This means the cardinality between Student and Class is 1:1.

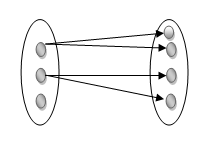
**Based on the cardinality, there are 3 types of relationship.**

* **One-to-One (1:1)**: As we saw in above example, one instance of the entity is mapped to only one instance of another entity.

Consider, HOD of the Department. There is only one HOD in one department. That is there is 1:1 relationship between the entity HOD and Department.

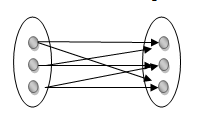


* **One-to-Many (1: M)**: As we can guess now, one to many relationship has one instance of entity related to multiple instances of another entity. One manager manages multiple employees in his department. Here Manager and Employee are entities, and the relationship is one to many. Similarly, one teacher teaches multiple classes is also a 1: M relationship.



* **Many-to-Many (M: N)**: This is a relationship where multiple instances of entities are related to multiple instances of another entity. A relationship between TEACHER and STUDENT is many to many. How? Multiple Teachers teach multiple numbers of Students.

Similarly, above example of 1:1 can be M:N !! Surprised?? Yes, it can be M:N relationship, provided, how we relate these two entities. Multiple Students enroll for multiple classes/courses makes this relationship M:N. The relationship 'studies' and 'enroll' made the difference here. That means, it all depends on the requirement and how we are relating the entities.



Participation Constraints

This represents how an entity is involved in the relation. That means, if all the entity values are participating in any relation, then it is called total participation. If only few values of an entity is part of relation, then it is a partial participation.

For example, ‘Employee Works for a Department’. Here all the employees work for one or the other department. No employees are left without department. Hence all the employees are participating in this relation. Thus, participation of employee is total participation.

But individual Department will not have all the employees. Each department will have only few groups of employees working. Hence partial participation of department is seen here.