

### Contents

- What is ER Model and ER Diagrams?
- Different elements used in ER Diagram
- Cardinality Constraints
- Participation Constraints
- Recursive Relationships
- Weak Entities
- Ternary Relationships and Ternary

### Entity-Relationship Model

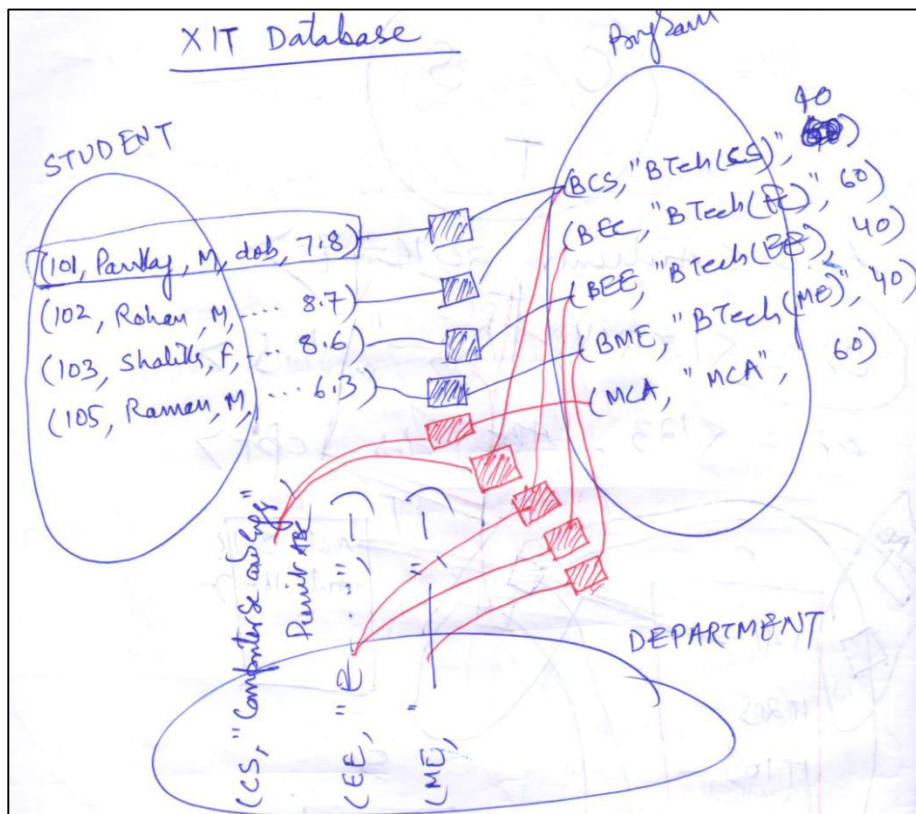
Entity-Relationship is a popular modeling technique at Conceptual Level. Since it is not implementation model, it does not define database manipulation operations.

In this model, database is represented as collection of entities and their relationship instances.

Recall: XIT-DB = {S, P, D, SP, PD}

Where S is set of student entities, P is set of program entities, D is set of Department entities, SP is set of instances of relationship between student and programs, and PD set of instances of relationship between programs and department entities. Figure below depicts the same.

In this model database schema is represented as Entity Relationship Diagrams, often referred as ERD. ER Diagrams are primarily used as first hand sketch for database design – aims to capture required data-items with their semantics.



ER model provide set of primitives that can be used to depict schema; that is ER Diagram. An ER Diagram typically uses following three concepts -

- Entity Types – contains their name and attributes, and key attribute.
- Relationship Types between entities
- Cardinality Constraints and Participation Constraints on relationships between entities

### Entity Set

- In ER model first term we introduce is Entity Set. It is a set of entities belonging to a particular “entity type”, for example, below is set of student entities. Each element in the set is “entity” or entity instance.

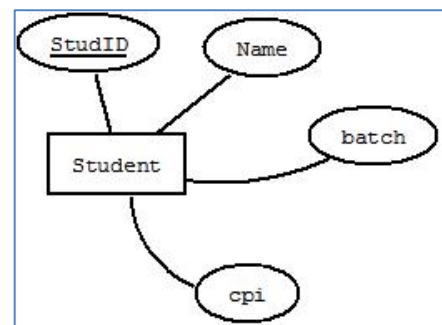
```
{ID:200701001,Name:Charu,Batch:2007, CPI:6.12},  
{ID:200711002,Name:Amit Khanna,Batch:2007,CPI:7.12},  
{ID:200711003,Name:Kamla Kiran,Batch:2007,CPI:7.50},  
{ID:200711004,Name:Raj Kumar,Batch:2007,CPI:4.00},  
{ID:200711005,Name:Raj Tiwari,Batch:2007,CPI:5.56},  
{ID:200811001,Name:Rama Kant,Batch:2008,CPI:8.12},  
{ID:200811002,Name:Akshya,Batch:2008,CPI:9.22},  
{ID:200811003,Name:Unnati Gupta,Batch:2008,CPI:5.52}
```

### Entity Type

- Entity Type is most important thing in ER model.
- An Entity Type describes schema for a “type of entity set”.
- In ERD, we describe entity types.

### Describing Entity Type in ERD

- Here is notation for describing an entity type.
- A rectangle for Entity Type\*, its name is mentioned within it.
- Attribute names are encircled
- Key attribute is underlined. (Note: value of key attribute is unique, and distinguishes an entity in its set).
- Key attribute might involve more than one attribute.



\*Note: Notations that are used here for describing ER Diagrams are Chen’s notation, were originally suggested by Dr Peter Chen in ’76.

### Attribute Types:

- Atomic – does not have more than one value in an attribute.
- Key Attributes: are used to identify an entity in its set.
- Composite
- Multi-value
- Stored and Derived attributes

## Multi-value attribute

Entity set with entities having multi-value attributes. Student entities below have email as multi-value attribute.

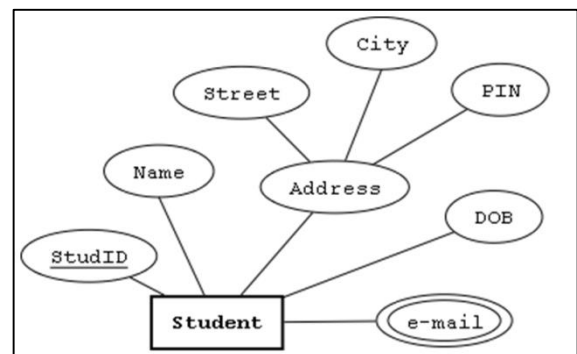
{ID:200701001, Name:Charu, Batch:2007, email:{charu@gmail.com, charu\_x@abc.in}, CPI:6.8},  
{ID:200711002, Name:Amit Khanna, Batch:2007, email:{amil@yahoo.com}, CPI:7.12},  
{ID:200711003, Name:Kamla Kiran, Batch:2007, email:{}, CPI:7.50},  
...

## Composite attributes

Composite attributes are composition of more than one value, for example Address composed of Street, City, and PIN.

{ID:200701001, Name:{Fname:Charu, Minit:K, Lname:Chawla}, Batch:2007, CPI:6.8},  
{ID:200711002, Name:{Fname:Amit, Minit:C, Lname:Patel}, Batch:2007, CPI:7.12},  
{ID:200711003, Name:{Fname:Kamla, Minit:S, Lname:Kiran}, Batch:2007, CPI:7.50},  
...

Diagram side by depicts multi-value and composite attributes for student entity type. Address is composite attribute while email is multi-value attribute.



==>

Observe semantics captured in following two representation of same entity. In first Name is composite attribute, where second is three different attributes for name.

{ID:200701001, Name:{Fname:Charu, Minit:K, Lname:Chawla}, Batch:2007, CPI:6.8}

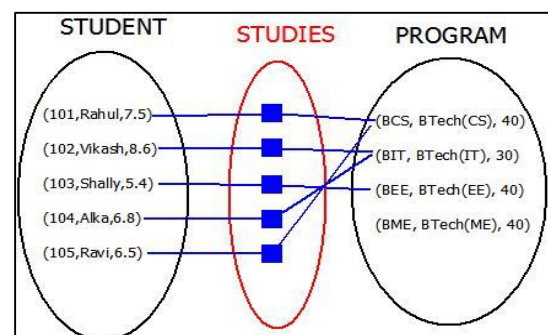
{ID:200701001, Fname:Charu, Minit:K, Lname:Chawla, Batch:2007, CPI:6.8}

## Relationships

### Relationship instance set

Diagram here shows that student entities are associated with program entities.

Set of little boxes (encircled in red) represents instances of associations between student and program entities. This is called “relationship instance set” in ER modeling.



This set captures facts of “association between student and program entities”.

This association also characterizes data model, and a name is given to a association, let us call this association as “STUDIES”

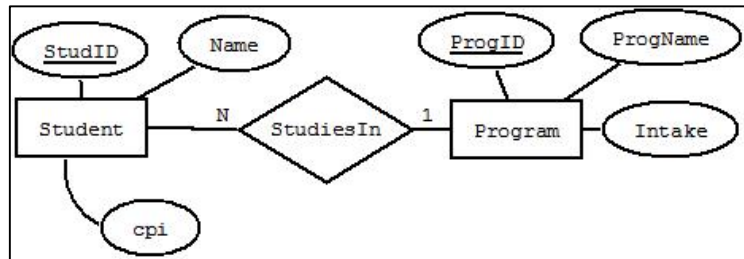
A relationship instance is identified (in its set) jointly by key attributes values of “participating entities”.

### Relationship [Type] in ER Diagram

Relationships types are shown as diamond in ER Diagram. The diagram below captures the fact that a student studies in a program.

It can be read as following-  
STUDENT and PROGRAM entity types are related by “STUDIES-IN” relationship type.

The label within the diamond gives name to the relationship.



It is a binary relationship. Binary relationship associates two entity types. Entities are said to be participating in the relationship.

Relationship can also have attributes. In the example here, suppose we want to say a student studies in a program from this year to that year, and can be depicted as following.

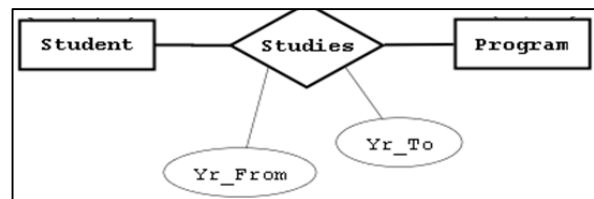
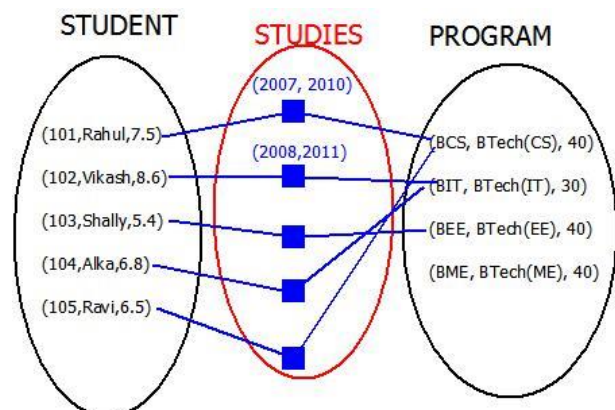


Diagram below shows relationship instance for the same.



### Cardinality Constraints in ERD

One of main constraint specified in ERD is Cardinality Constraint. It specifies number of instance an entity from one side can associate with entities on other side. For instance a student (entity from one side) can be associated with at-most (and at-least) one department (entity on other side).

There are three types of cardinality constraints we have -

- One to One (1:1)
- One to Many (1:N) [flipped is Many to One (N:1)]
- Many to Many (M:N)

In some texts, this is also referred as Cardinality Ratio (for example Elmasri/Navathe); but this is not really ratio in precise terms. Probably people call it ratio as it is expressed in ratio form (1:N)!

We will use the term “Cardinality Constraints”

You can use following decision flow to figure out cardinality constraint for a relationship.

Take an entity from one side (side1), and see how many entities, it can be associated on other side (side2)

If answer is 1

see how many entities on side1, an entity from side2 can be associated with

If answer is 1

It is 1:1

else if answer is more than 1

It is 1:N

else if answer is more than 1

see how many entities on side1, an entity from side2 can be associated with

If answer is 1

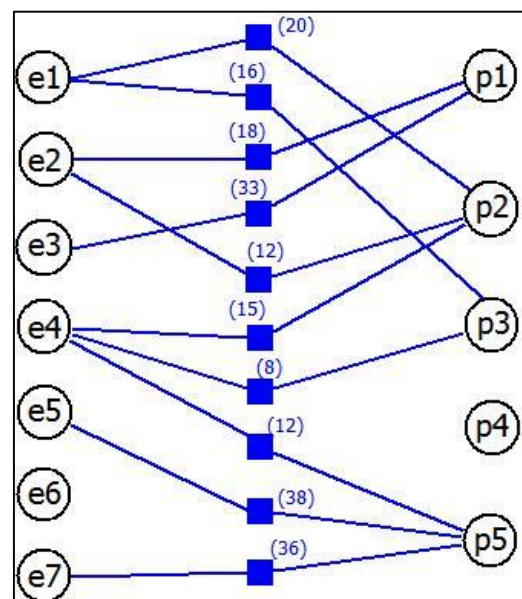
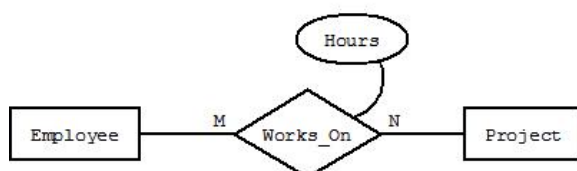
It is 1:N (actually N:1 but same as 1:N)

else if answer is more than 1

It is M:N

Examples:

- A student gets allocated only one room, and one room can accommodate only one student (1:1).
- An employee works for only one department, where as a department can have number of employee (1:N).
- A product category can have many products but one product will have only one category (1:N).
- A product category can have many products and one product can belong to many categories (M:N).
- A bill has many items, and an item can occur in many bills.
- A book is published by only one publisher but a publisher can have many books(1:N)
- A book can have many authors, and an author can author many books.
- An employee can work on any number of Projects, and a Project can have any number of employees working on it (M:N)
  - When employee works on a project, we also store for how many hours the employee works on a project
  - Diagram side by and shows instances of employees working on projects.
  - Below is schema depiction of the relationship



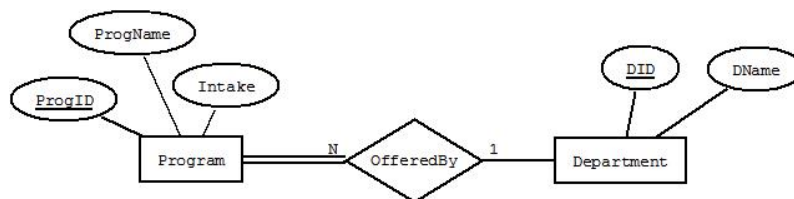
- A property can be owned by many persons, and a person can own many properties.
  - When a person owns a property, we require an additional attribute share\_percentage; that what percentage of share the person has for the property.
- Student has one of Professor as Mentor (?:?)
- Course uses Text Book(s), and one text book could be used in multiple courses (?:?)
- A city is in a Country, and Each Country has one of its city as its capital (?:?)

### Participation Constraints

Participation constraint specifies whether participation of an entity in a relationship is mandatory or not. Mandatory means an entity cannot exist in its set without participating in the relationship. That is it is necessary that an entity needs to be associated with some entity on the other side. Such a participation constraint is referred as “**Total**”. If it is not mandatory (or total) then it is “optional”.

For example, a program entity (XIT database) necessarily needs to be associated with some Department. Then Program entity has mandatory (or total) participation in Offered-By relationship. (say we have this between program and department). However it is not necessary that a department offers a program. Therefore participating this entity in **offered-by** is not mandatory.

Mandatory participations are shown by double lines while optional participations are shown by single in ERD. Below is depiction of Offer-By relationship.

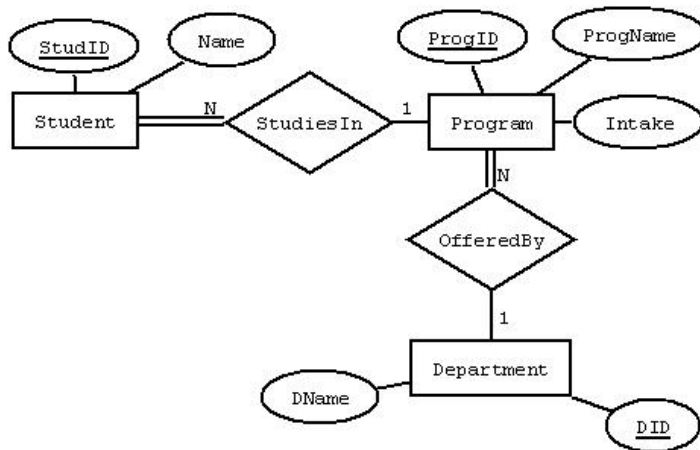


### Examples

- A student entity may not exist without getting associated with a Program (or in other words without participating in studies-in relationship. However a program may exist.
- An employee may not exist without associated with a department as an employee, however a department may exist.
- An employee may exist without becoming manager of some department
- An employee may exist without working on a project and project having no employee working at all.

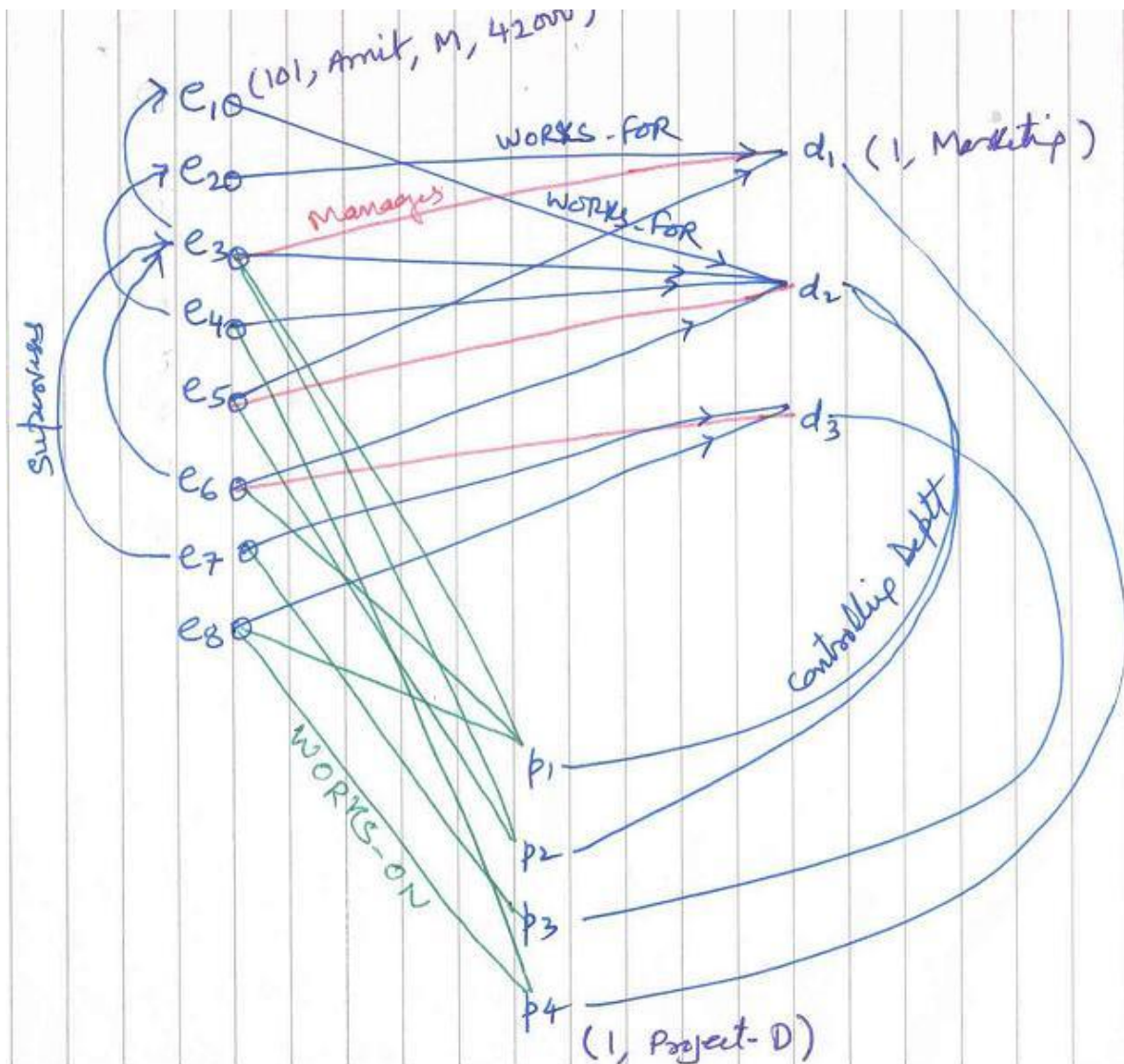


Here is complete ERD for XIT schema:



See what sense following sketch containing few instances from employee, department, and project having some interactions.

Attempt sketching ERD for the same-

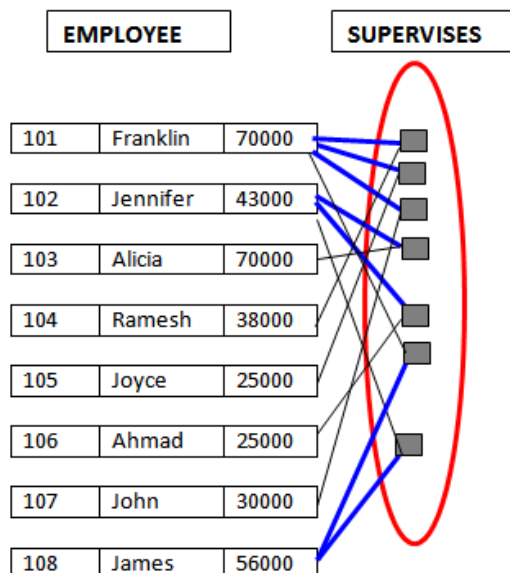


Case Studies#1: TGMG (description given in separate document)

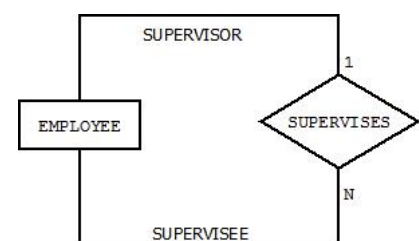
Case Studies#2: Library (description given in separate document)

## Recursive Relationship

- An entity can be associated with another entity from its own set (same type of entity). For example, as figure below shows, Franklin (101) supervises Ramesh (104); both entities are from set of employee entities, and hence name “recursive relationship”.



- Set of little boxes encircled in red forms relationship instances set; let us give name to this relationship type “SUPERVISES”
- Note that there is an issue here – when you have participating entities from different sets (types), roles of entities in relationship is implicit and clear. For example- MANAGES relationship between EMPLOYEE and DEPARTMENT (figure previous page); a “manages” instance where e3 and d1 are interacting (related), expresses the fact that employee e3 manages department d1; and we say that employee e3 plays role of manager while department d1 is managed.
- However this is not the case for recursive relationship. For example, we have instance of e101 and e104 associated in “SUPERVISES” relationship, it is not clear who is supervisor and who is supervisee. Therefore in recursive relationship instance, it is also specified what role an entity plays in the relationship. In this case we say e101 plays role of supervisor and e104 is supervisee. In the diagram blue and thicker line indicates supervisor role while thinner line indicates supervisee role.
- Here is how recursive relationship is depicted in ER Diagram -
- Note that roles are labeled, and Cardinality and participations constraints are specified for Roles. An employee can play a supervisee role only once, that is can be associate as supervisee with at-most one other employee, where as an





employee can act as supervisor with multiple other employees, therefore cardinality is 1:N, supervisee being in many side. For participation both roles are optional; not necessary every employee acts as supervisee, and not necessary an employee supervises any.

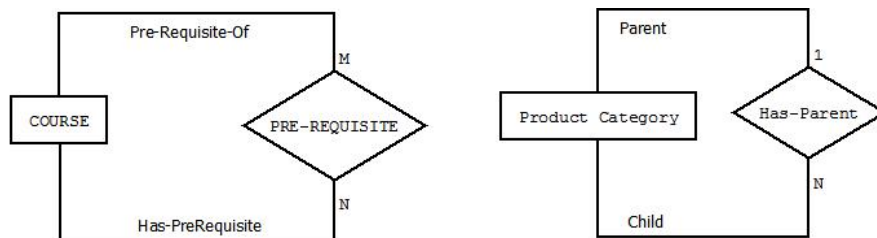
## More examples of Recursive relationship

(1) A course has another course as pre-requisite (figure below left depicts its ERD).

Cardinality: there can be any of courses as pre-requisite of a course, and a course can be pre-requisite of many courses, therefore M:N. Participation: Not necessary every course is pre-requisite of any course, and not necessary every course will have a pre-requisite.

(2) A Product category has another category as parent category. Cardinality: A category can have at most one parent; but a parent can have many multiple children, therefore 1:N.

Participation: Not necessary every category will have a parent, and not necessary that a category will have children (figure below right depicts its ERD).



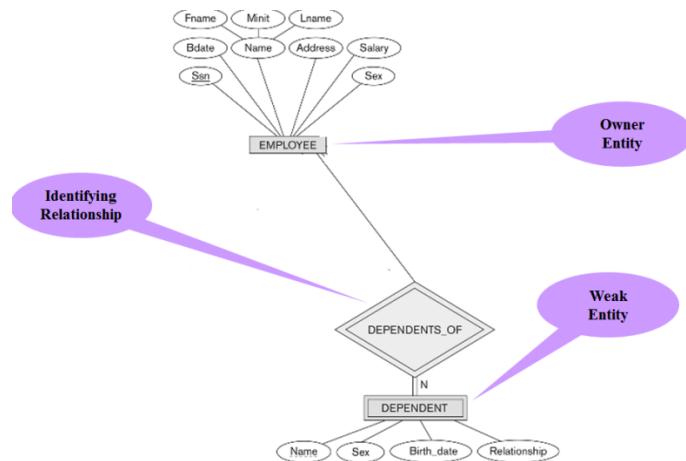
## Weak Entity

- When finding a key for an entity is harder, then possibly it is a weak entity. Consider company schema; suppose the following details of dependents of the employee are to be recorded in the database – name, gender, birth\_date, and relationship with the employee. Let us say below is a set of dependents, can you figure out the key attribute for the set?

101	{	Alice	F	1976-	DAUGHTER
		Theodore	M	1973-	SON
		Joy	F	1948-	SPOUSE
106	{	Abner	M	1932-	SPOUSE
		Michael	M	1978-	SON
102	{	Alice	F	1978-	DAUGHTER
		Elizabeth	F	1957-	SPOUSE

- Probably “name” within an employee’s dependents. This is fair assumption.
- When we put dependents from all employees together in a single set, name is no more a key.
- We require adding key of employee to make name distinct in this combined set.
- This is repeating phenomenon in many situations, where an entity has a key but that is key is valid within a subset belonging to one other entity.
- This situation is handled using a notion of *weak entity*; weak entities are always “identified by” some *strong entity*, called as *owner entity*. In the current example of dependents, dependents is a weak entity identified by “Employee”.

- In ER Diagram, weak entity type is modeled as following-



### Characteristics of weak entity

- A weak entity is identified by some strong “Owner Entity”.
- A weak entity is depicted by double line rectangle; is related with owner entity through “*Identifying Relationship*”.
- Identifying relationship is depicted by double line diamond. Cardinality wise, it is always 1:N, weak entity being in many side.
- Participation of weak entity is always total/mandatory.
- Weak entities have *partial key*. Value of partial key is “unique within a single “owner entity””. In the example here: dependent name is partial key.

### Other examples #1: Player as Weak Entity

- Consider you are organizing a inter college cricket meet; where you have different teams. Each team has different players. While attempting to model this you have a set of team entities, and set of player entities. Let us say we have identified following attributes for Team (name, city, state), and assume that team-names are unique, and hence key for team entity.
- For players, let us say, we have following attributes- name, runs, wickets, sixes, fours). Now what is the key for a player? Player is modeled as “weak entity” identified by Team.

### Other examples #2: Room as Weak Entity

- Class Rooms in a University campus? Let us say we have following attributes for room- Room-No, Seating capacity, is\_ac, etc. What can be the key for ROOM?
- Room needs to be modeled as Weak Entity identified by Building.
- Remember Room number like CEP-110, KRB-123 captures the same notion; CEP identifies building name while 110 is room no. This kind of representation can be enough for human mind but not enough for computer databases. We need to use building name and room number as separate attributes. In absence of such a separation how do we answer a query like “Give me rooms that having capacity of 200 in CEP building”?

A strong entity in some context may become weak in broadened context (and vice versa)

- In DAIICT scenario course is strong entity having course\_no as key. IT214 is unique within DA-IICT. Now consider a database scenario, where courses numbers from different universities are merged for some purpose? Then we need to say IT214 of DA-IICT ==> Course requires an “owner entity” institute, and becomes weak?

In some cases, a weak entity may alternatively be represented as multi-value composite attribute

- Can we model dependent as multi-value composite attribute? Possibly yes, and in many such cases this stands out to be an alternative. However, in cases where weak entity participates in some relationship, we may not be alternatively representing it using multi-value composite attribute. For example Player may not be modeled as composite attribute of Team – player is an important entity in sports meet, and interact (have relationships) with other entities like Match.
- Similarly Room may not be modeled as attribute of building – room may be associated with courses or so in a time-tabling problem.

Exercise#: Complete ER Diagram for the company schema.

Caution while creating ER Diagrams

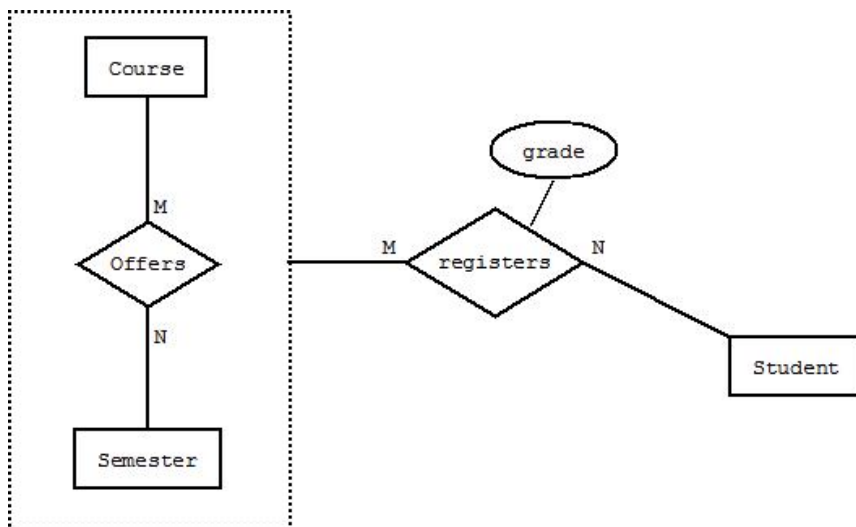
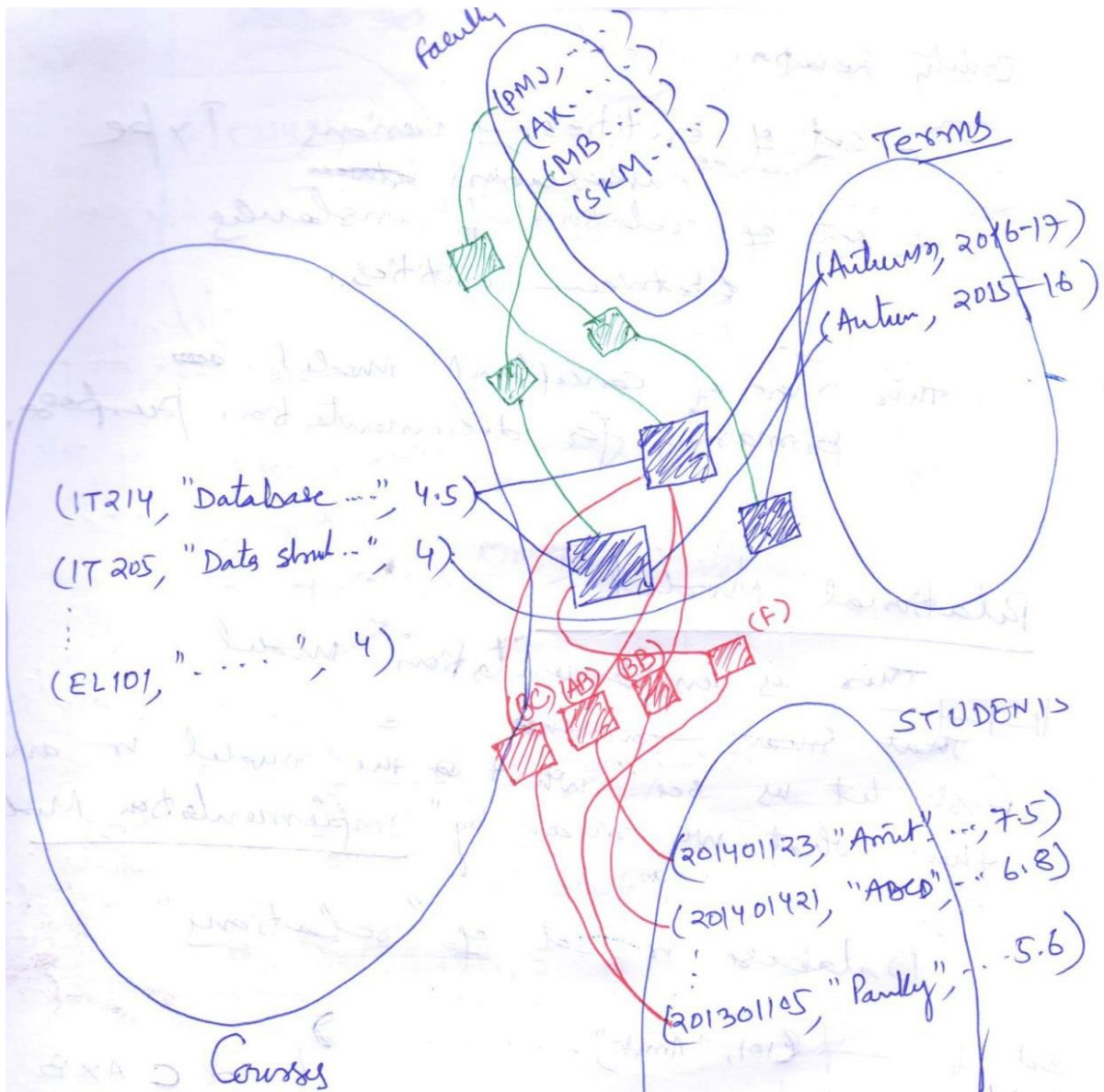
- Attributes like DNO in employee entity, mgr\_eno in department sound like correct; but really not, these attributes basically captures a relationships; and relationships are represented differently in ER Diagrams.

Aggregation

Let us begin with our example of the ACAD database. Consider the following depiction, discussed very early in the course. Here we have the following entity sets: COURSE, TERM, and STUDENT. Here it is important to note that a student’s association is with a relationship instance of course getting offered in a term. Do you agree?

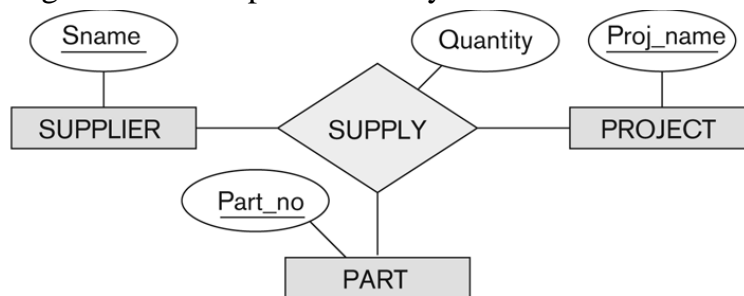
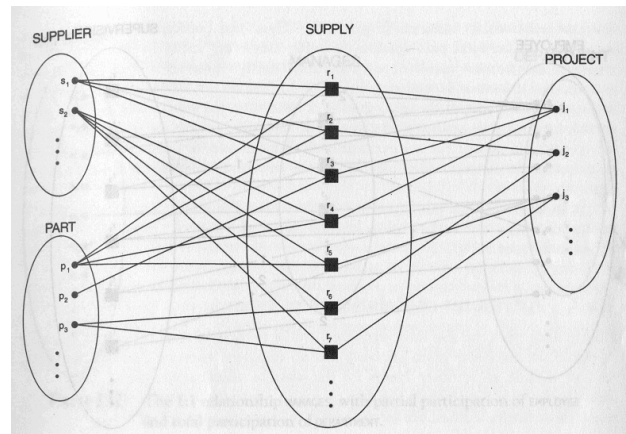
ER Model allows us to associate a relationship instance with an entity instance and this is called Aggregation. In aggregation, a “relationship instance” is treated like any other entity (instance) and allowed to get associated with another entity (instance). For example, we have a binary relationship instance (IT214, Autumn’16); this “relationship instance” can be associated (to represent registration instance) with a student entity 201501123. The figure above shows this.

In ER Diagram it is depicted as shown below.



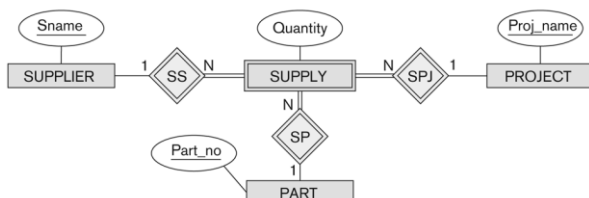
## Ternary Relationship

- A relationship in which two entities participate is called a binary relationship. An employee entity e1 and department entity d2 join together and form a binary relationship instance. If a relationship instance has three entities participating into a relationship, then it is ternary.
- The diagram here shows a classical example of ternary. There are three entity sets here- supplier, parts, and projects. This set of associations captures a recordable fact that “a supplier s1 supplies part p1 to project j1”, and forms an instance of supply r1; let us name this relationship as SUPPLY.
- It is worth pointing out again that in a ternary relationship instance, we necessarily have three entities from respective sets joining together.
- The diagram below depicts a ternary in ERD -



- Cardinality and Participation constraints for ternary: we will skip this. Constraints (1:1, 1:N, M:N) are for binary. Very few discussions about; and non-standard. Possible reasons are (1) Most ternary relationships can be represented using binary and “aggregated binary” relationships; (2) Ternary (and more) rarely occur in real situations.
- For participation, we can always extend the same principle for binary to ternary and higher relationship orders.

PS: Ternary relationships can always be represented as weak entities identified jointly by three owner entities (as shown in the diagram below)-

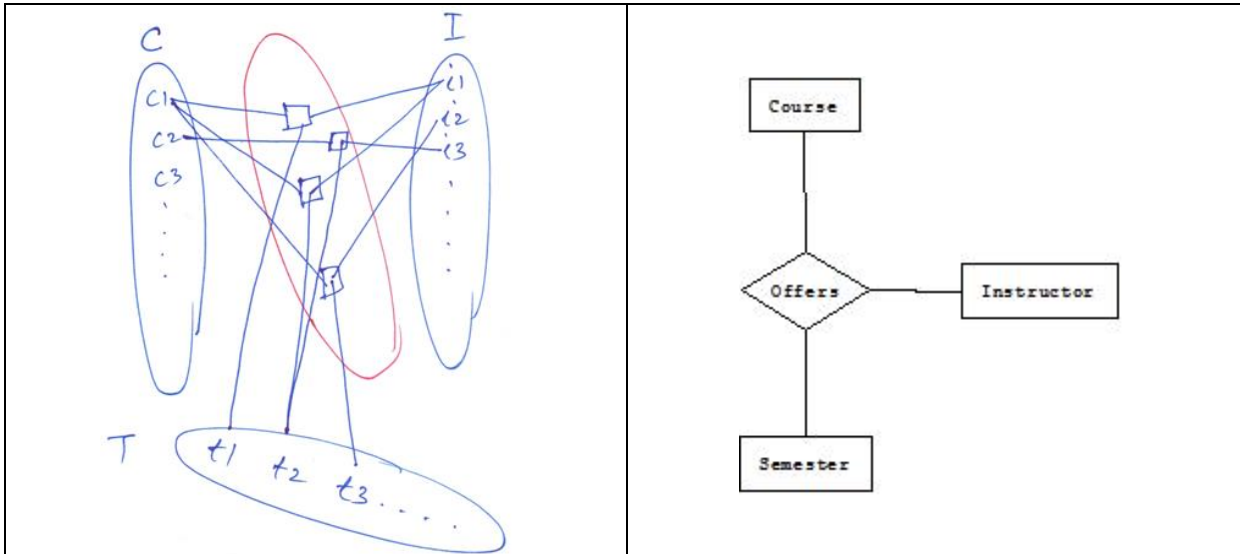


Good news is often we do not need ternary and higher degree relationships.

Other example – “OFFERS: Course-Term-Instructor”



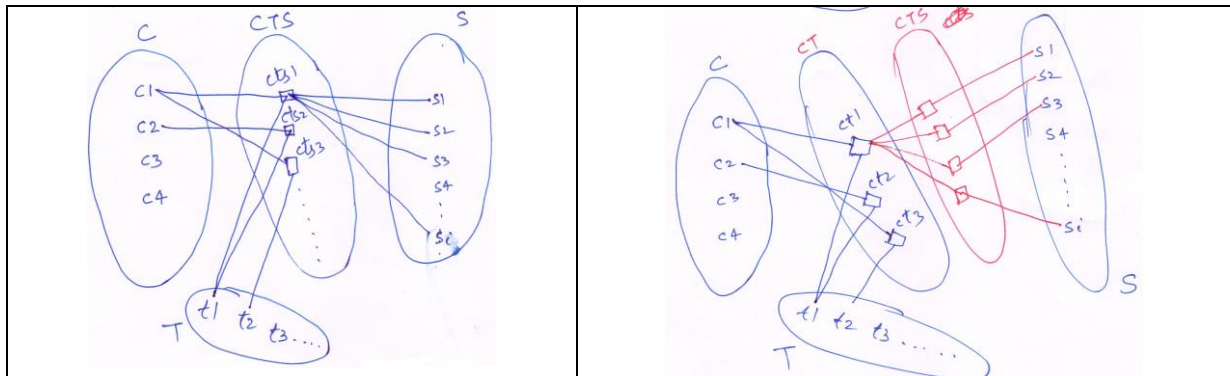
- In DA-Acad scenario: three entities a course (IT214), a term (Autumn'2016), and an instructor (PMJ) join together to form an instance of “course offer” relationship instance. This way a number of such ternary interactions, form set of “course offers”. A natural name for this relationship is “Course-Offers”, or simply “Offers”. Below is an instance set and ERD for “Offers” relationship –



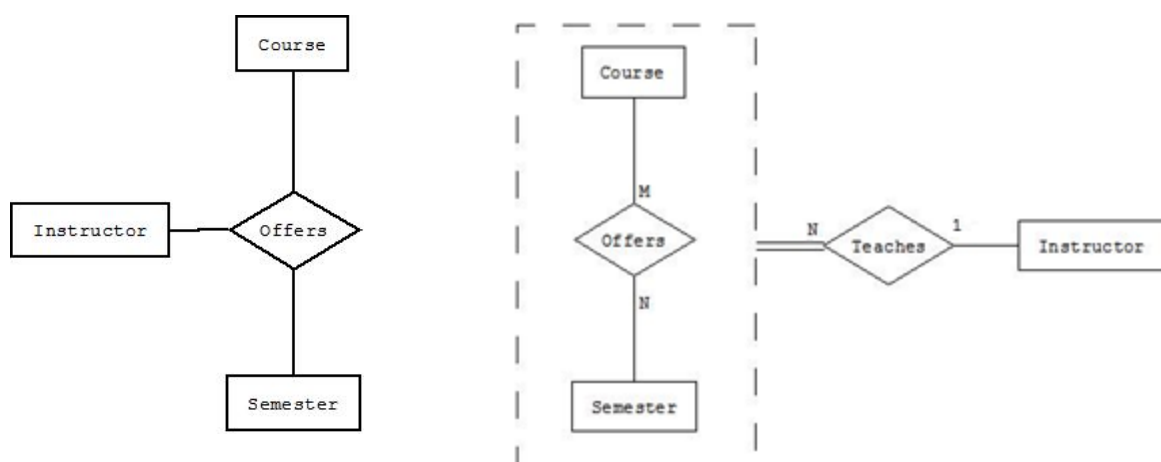
- If a student also joins above offers relationship (to register), will lead to having four entities participating in a relationship, a *Quaternary* relationship? However later on we will establish that this does not correctly capture the situation.

## Aggregation and Ternary

The diagram below should capture the differences; “Course-Term-Student” represented both ways.



- In aggregation, association of Course (C) and term (T) is independent of Student (S). Once an instance of CT (c1, t1) is there, it can interact with any student s1 from S, finally forming CTS instance (c1, t1, s1). Assumption in aggregation is that relationship instance (c1, t1) is already there before s1 interacts with it. Whereas in case of ternary, joining of (c1, t1, s1) happens all-together.
- For the purpose of “Registration” where one instance from each set course, term, and student joins to form a registration instance, ternary does not correctly captures the requirement. Aggregation is what correctly represents this situation. First, we have “offering”, as a independent binary instance of C and T, then student s1 joins instance (c1,t1). Ternary does not serve required purpose here; offering of a course is independent of student registering into it. Ternary does not correctly capture CTS situation.
- What about COURSE, TERM, INSTRUCTOR? Do we model it using ternary or aggregation? Diagram below depicts it both ways?



Both are identical here and should be fine in this case, assuming that there is always need to associate an instructor with a Course-Term instance, i.e. course offering. However, cardinality of an instructor’s association with a “course offering” can be better represented using aggregation.

Ternary and Aggregation may turn out to be the same when the participation of the third entity is mandatory.

This can be summarized as follows:

1. If any of the participating entities is participating optional then it is not ternary but aggregation
  - a. Note that entity being optional in a relationship and being optional participation are two different things. Example Employee in “MANAGES” relationship and Student in “REGISTERS” relationship.
2. Specifying cardinality and participation constraints are more clear in aggregation!

**Final Words:** Aggregation is always preferred over ternary wherever possible!

- Ternary and higher may not be very popular in OLTP applications but could be quite common in data warehouse/OLAP applications:
- Supply-Part-Project (qty being relationship attribute) should also be drawn from some OLAP application.
- As another example, we have Doctor, Drug, Ailment, a ternary relationship, and #no-of-cases being the relationship attribute; useful of some OLAP. It could built from OLTP entities and relationships as follows-
  - Doctor and Patient in binary relationship “consults”– attributes could be consultation fees etc
  - Consults then “aggregately” participates with drugs and ailments

### Creating ER Models:

For a given database project, you can follow the steps and guidelines enumerated here.

#### Steps

- Understand Database Requirements and Document them
- Identifying Entities and Relationships
- It is an iterative process and may take several iterations before you can reach a correct ER Diagram

#### Understand Database Requirement

- Describe database requirements.
- Also have a representative list of queries that your database is expected to answer [refer to sample description for bookstore and other descriptions shared with you as problem description]

#### Entity Identification:

- Common technique is Noun Analysis. In this approach we consider nouns appearing in requirement description of database scenario, as candidate Entities.

- Note down all entities that comes to your mind.
- Try finding out for all listed entity types
  - Attributes, and Key attributes
  - Sets of instances for each entity type that you have noted
- If we are not able to find out certain sets of attributes or attributes that you are seeing in an entity actually belong to some other entity or relationship. If not able to find key, or not able to find an instance set for some entity type. Such entities shall be ruled out.
- An entity normally has more than one attributes and “identifying attributes”. One attribute entity can be better attached with another entity as an additional attribute.
- Entities are often loudly visible in requirement description.
- Following are a few examples of entities
  - People - Employees, Students, Customers
  - Places: sites or locations - Cities, Offices, Routes
  - Things: tangible physical objects - Equipment, Products, Buildings
  - Organizations: Teams, Suppliers, Departments
  - Concepts: intangible ideas that track business or other activities - Projects, Accounts, Complaints, Invoice
- If an entity type has only one instance, THEN it is not modeled as entity, for example DAICT in DA-Acad; Challenge in TGMS, or so
- Relationships
  - Verbs Analysis. Relationships are potentially found in verbs. Try identifying various interactions entities take place.
  - A relationship instance should be identifiable by the keys of its participating entities; if not then the relationship may need to be modeled as entity/weak entity. For example Issues in the library scenario; billed in the billing scenario.
  - Some verbs in the requirement description may sound like relationships but actually may not be relationships, for example
    - “Student Fills Enrollment Form”. Here neither the “Enrollment form” is an entity nor “FillsIn” is a relationship
    - Similarly, “registers” is not a relationship in the statement “Team registers in the competition”
  - Some relationships may look like ternary or higher degree but in most cases, it is not. Explore binary alternatives, and aggregation before really making a relationship ternary.