DBMS Database Models

A **data model** is a relatively simple representation, usually graphical, of more complex real-world data structures.

Within the database environment, a data model represents data structures and their characteristics,
relations, constraints, transformations, and other constructs with the purpose of supporting a specific problem domain.

A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system.

Basics

- The basic building blocks of all data models are entities, attributes, relationships, and constraints.
- An **entity** is anything (a person, a place, a thing, or an event) about which data are to be collected and stored. An entity represents a particular type of object in the real world. Because an entity represents a particular type of object, entities are “distinguishable”—that is, each entity occurrence is unique and distinct
- An **attribute** is a characteristic of an entity.
- A **relationship** describes an association among entities.

– One-to-many (1:M or 1..*) relationship
– Many-to-many (M:N or *..*) relationship
– One-to-one (1:1 or 1..1) relationship.

- A **constraint** is a restriction placed on the data. Constraints are important because they help to ensure data integrity. Constraints are normally expressed in the form of rules

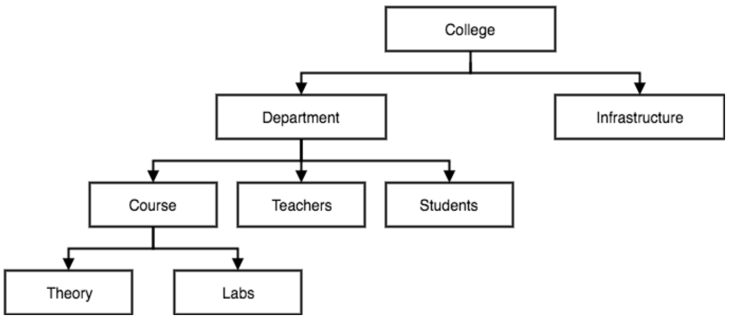
While the **Relational Model** is the most widely used database model, there are other models too:

- Hierarchical Model
- Network Model
- Entity-relationship Model
- Relational Model

Hierarchical Model

- The hierarchical model was developed in the 1960s to manage large amount of data for complex manufacturing project.
- This database model organizes data into a tree-like-structure, with a single root, to which all the other data is linked. The hierarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.
- This model is used to depict one-to-many(1:M) relationship between Parent and Child.
  o Each Parent may have child

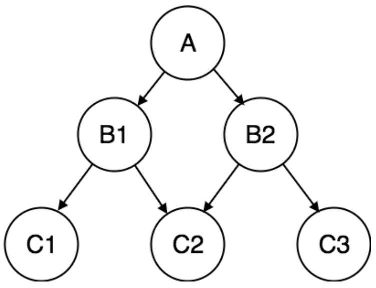
o Each child has only one Parent.

- This model efficiently describes many real-world relationships like index of a book, recipes etc.
- All attributes of a specific record are listed under an entity type.
- In hierarchical model, data is organized into tree-like structure with one **one-to-many relationship** between two different types of data, for example, one department can have many courses, many professors and of-course many students.
- 

- Advantages
  - Easy to design
  - Easy to share
  - Helps in data integrity

- Disadvantages:
  - Complex to implement
  - A clear picture of relations and attributes should be well planned beforehand.
  - Difficult to manage : Difficult to modify the structure as and when needed.
  - Lack of standards
  - Implementation limitation
  - Can be used in only 1-many relationships.

Network Model

- The network model was created to represent complex data relationship more effectively than hierarchical model, to improve database performance, and to impose a database standard.
- This is an extension of the Hierarchical model. In this model data is organized more like a graph, and are allowed to have more than one parent node.
- In this database model, data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used to map **many-to-many** data relationships.
- This was the most widely used database model, before Relational Model was introduced.
- 

While the network database model is generally not used today, the definitions of standard database *concepts* that emerged with the network model are still used by modern data models. Some important concepts that were defined at this time are:

- The **schema**, which is the conceptual organization of the entire database as viewed by the database administrator.
- The **subschema**, which defines the portion of the database "seen" by the application programs that actually produce the desired information from the data contained within the database.
- A **data management language** (**DML**), which defines the environment in which data can be managed and to work with the data in the database.
- A schema **data definition language** (**DDL**), which enables the database administrator to define the schema components.

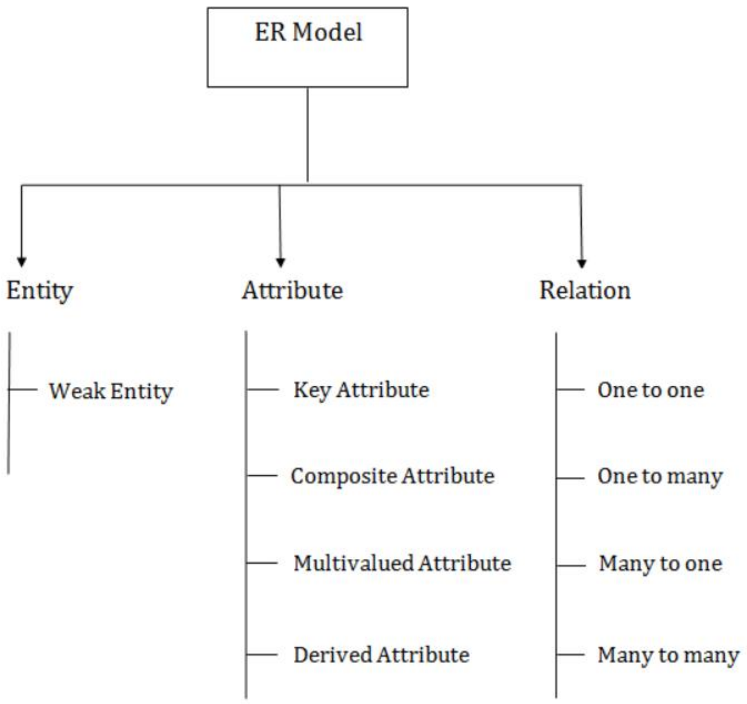
Advantages

- Easy to design
- Many to many relations
- Superior data access than hierarchial
- Has some standards.
- Integrity..

Disadvantages:

- Complex system due to too many connections.
- Too cumbersome
- Lack of ad-hoc query capability
- Any structural change need to change in whole application – no structural independence

CHECK THE NOTES.



The Entity-Relationship Model:-

 The entity-relationship (E-R) data model was developed to facilitate database design that represents the overall logical structure of a database.
 The E-R model is one of several semantic data model.
 The E-R model is very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema.

A. Entity Sets:-

 An **entity** is a “thing” or “object” in the real world that is distinguishable from all other objects.
 For example, each person in an enterprise is an entity. An entity has a set of properties, and the values for some set of properties may uniquely identify an entity.
 For instance, a person may have a person_id property whose value uniquely identifies that person.
 An entity might be

- An object with physical existence (e.g., a lecturer, a student, a car)
- An object with conceptual existence (e.g., a course, a job, a positio

 An **entity set** is a set of entities of the same data type that share same properties, or attributes.
 An entity is represented by a set of **attributes.** Attributes are descriptive properties possessed by each member of an entity set.
 Each entity has a **value** for each of its attributes.
 Weak Entity: An entity that cannot be uniquely identified by its own attributes and relies on the relationship with other entity is called weak entity. The weak entity is represented by a double rectangle. For example – a bank account cannot be uniquely identified without knowing the bank to which the account belongs, so bank account is a weak entity.

B. **Attributes:-**

 For each attribute, there is a set of permitted values, called the domain, or value set, of that attribute.

1. Key attribute:

- A key attribute can uniquely identify an entity from an entity set.
- Cannot have null or duplicate values.

 Simple and **composite** attributes:-
 The attributes have been simple; that is they have not been divided into subparts.
 Composite attributes on the other hand, can be divided into subparts.
 For example, an attribute name could be structured as a composite attribute consisting of first_name, middle_initial, and last_name.

2. Composite attribute:

An attribute that is a combination of other attributes is known as composite attribute. For example, In student entity, the student address is a composite attribute as an address is composed of other attributes such as pin code, state, country

 required and **optional** attributes:-
 A **required attribute** is an attribute that must have a value; in other words, it cannot be left empty\
 An **optional attribute** is an attribute that does not require a value; therefore, it can be left empty

 Single-valued and **mutlivalued** attributes:-
 There may be instance where an attributes has a sert of values for specific entity.
 Consider an employee entity set with the attribute phone_number.
 An employee may have zero, one, or several phone numbers.
 This type of attribute is said to be multivalued.

3. Multivalued attribute:

- An attribute that can hold multiple values is known as multivalued attribute. It is represented with **double ovals** in an ER Diagram. For example – A person can have more than one phone numbers so the phone number attribute is multivalued.

 Derived Attributes:-
 The value for this type of attribute can be derived from the values of other related attributes or entities.
 Customer entity set has an attribute age that indicates the customer's age.
 If the customer entity set also has an attribute date_of_birth, we can calculate age from date_of_birth and current date. Thus, age is derived attribute.
 An attribute takes a **null** value when an entity does not have a value for it.
 The null value may indicate "not-applicable"- that is, that the value does not exist for the entity.

4. Derived attribute:

- A derived attribute is one whose value is dynamic and derived from another attribute. It is represented by **dashed oval** in an ER Diagram. For example – Person age is a derived attribute as it changes over time and can be derived from another attribute (Date of birth).

Read RDBMS TextBook for this.

C. Relationship Sets:-
 A **relationship** is an association among several entities.
 A **relationship set** is a set of relationships of the same type.
 Consider the two entity sets customer and loan. We define the relationship set borrower to denote the association between customers and the bank loans that the customers have.
 The association between entity sets is referred to as participation; that is, the entity sets E1, E2…, En **Participate** in relationship set R.
 The function that an entity plays in a relationship is called that entity's **role**.
 Since entity set participating in a relationship set are generally distinct, roles are implicit and are not usually specified.
 However, they are useful when the meaning a **recursive** relationship set.
 A relationship may also have attributes called **descriptive attributes**.

Mapping Cardinalities/Relationship Constraints:-
 Mapping cardinalities, or cardinality rations, express the number of entities to which another entity can be associated via a relationship set.

 For a binary relationship set R between entity sets A and B, the mapping cardinality must be one of the following:
 Oneto-one. An entity is A is associated with at most one entity in B, an entity in B is associated with at most one entity in A.
 One-to-many. An entity in A is associated with any number of entities in B. An entity in B, however, can be associated with at most one entity in A.
 Many-to-one. An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number of entities in A.
 Many-to-many. An entity in A is associated with any number of entities in B, and an entity in B, is associated with any number of entities in A.

Keys:-
 A **key** allows us to identify a set of attributes that suffice to distinguish entities from each other.
 Keys also help uniquely identify relationships, and thus distinguish relationships from each other.
 A **superkey** is a set of one or more attributes that, taken collectively, allow us to identify uniquely an entity in the entity set.
 For example, the customer_id attribute of the entity set customer is sufficient to distinguish one customer entity from another.
 Thus, customer_is is a superkey.

- Similarly, the combination of customer_name and customer_id is a superkey for the entity set customer.
- The customer_name attribute of customer is not a superkey, because several people might have the same name.
- Such minimal superkeys are called **candidate keys**.
- Suppose that combination of customer_name and customer_street is sufficient to distinguish among members of the customer entity sets.
- Then, both {customer_id} and {cus tomer_name, cutomer_street} are candidate keys.

- **Primary key** to donate a candidate key that is chosen by the database designer as the principal means of identifying entities within an entity set.

Participation Constraints:-

- The participation of an entity set E in a relationship set R is said to be a total if every entity in E participates in at least one relationship in R.
- If only some entities in E participate in relationship in R, the participation of entity set E in relationship R is said to be partial.
- For example, we expect every loan entity to be related to at least one customer through the borrower relationship.
- Therefore the participation of loan in the relationship set borrower is total.
- An individual can be a bank customer whether or not she has a loan with the bank.
- Participation off customer in the borrower relationship set is therefore partial.

Entity-Relationship Diagrams:-

- **Rectangles,** which represent **entity set**
- **Ellipses,** which represent **attribute**
- **Diamonds,** which represent **relationship sets**
- **Lines,** which **link attributes to entity sets** and **entity sets to relationship sets**
- **Double ellipses,** which represent **multivalued attributes**
- **Dashed ellipses,** which denote **derived attributes**
- **Double lines,** which indicate **total participation of an entity in a relationship set**
- **Double rectangles,** which represent **weak entity set**
- **Underline** represents **primary key**

See mam notes for how to draw.

Weak and Strong Entity Sets:-
 An entity set may not have sufficient attributes to form a primary key.
 Such an entity set is termed a **weak entity**.
 An entity set that has a primary key is termed a strong entity set.
 For a weak entity set to be meaningful it must be associated with another entity set, called the **identifying** or **owner entity set.**
 Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be **existence dependent** on the identifying entity set.
 The identifying entity set is said to **own** the weak entity set that it identifies.
 The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship.**

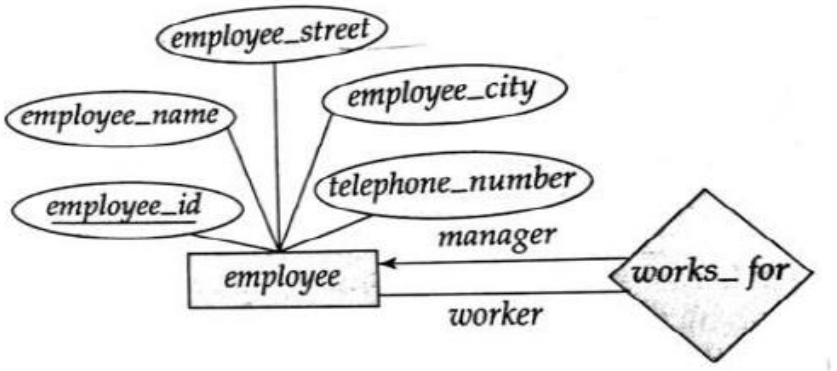
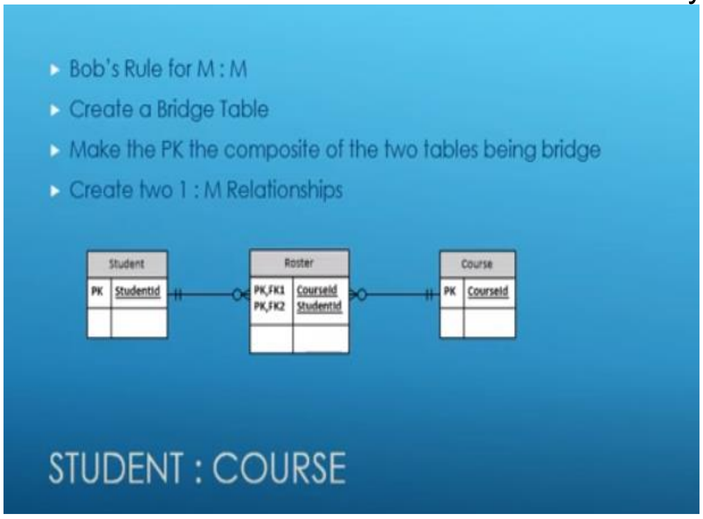


Figure 6.11 E-R diagram with role indicators.

Composite Entity:
A composite entity is also known as a "bridge" entity. This "bridge" is used to handle the many-to-many relationships that the traditional entity could not handle. This entity lies between the two entities that are of interest and this composite entity shares the primary keys from both the connecting tables. This composite entity is also known as a "gerund" because it has the characteristics of an entity and a relationship


- Bob's Rule for M : M
- Create a Bridge Table
- Make the PK the composite of the two tables being bridge
- Create two 1 : M Relationships

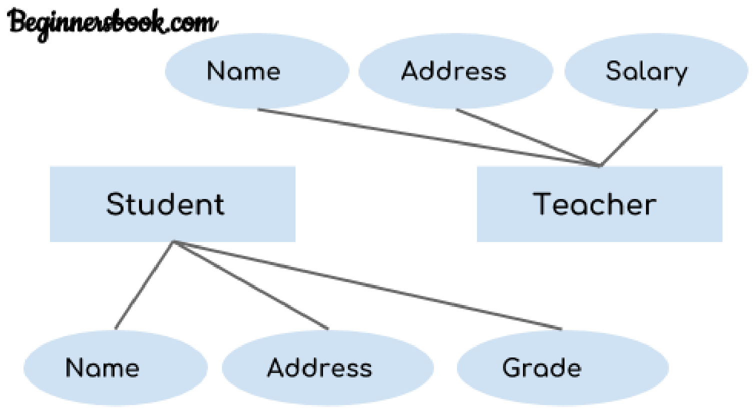
STUDENT : COURSE

Generalization

Generalization is a process in which the common attributes of more than one entities form a new entity. This newly formed entity is called generalized entity.

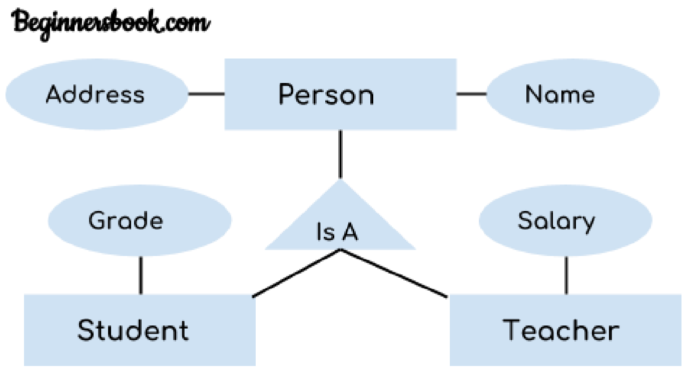
1. Generalization uses bottom-up approach where two or more lower level entities combine together to form a higher level new entity.
2. The new generalized entity can further combine together with lower level entity to create a further higher level generalized entity.

Example:
Lets say we have two entities Student and Teacher. Attributes of Entity Student are: Name, Address & Grade. Attributes of Entity Teacher are: Name, Address & Salary
The ER diagram before generalization looks like this:



Before Generalization

These two entities have two common attributes: Name and Address, we can make a generalized entity with these common attributes. Lets have a look at the ER model after generalization.



Generalization

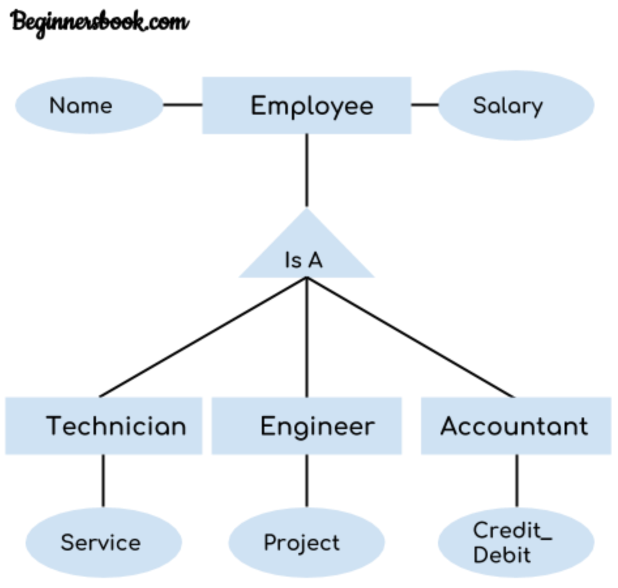
The ER diagram after generalization:
We have created a new generalized entity Person and this entity has the common attributes of both the entities. As you can see in the following ER diagram that after the generalization process the entities Student and Teacher only has the specialized attributes Grade and Salary respectively and their common attributes (Name & Address) are now associated with a new entity Person which is in the relationship with both the entities (Student & Teacher).

Specialization

Specialization is a process in which an entity is divided into sub-entities. You can think of it as a reverse process of generalization, in generalization two entities combine together to form a new higher level entity. Specialization is a top-down process.

The idea behind Specialization is to find the subsets of entities that have few distinguish attributes. For example – Consider an entity employee which can be further classified as sub-entities Technician, Engineer & Accountant because these sub entities have some distinguish attributes.

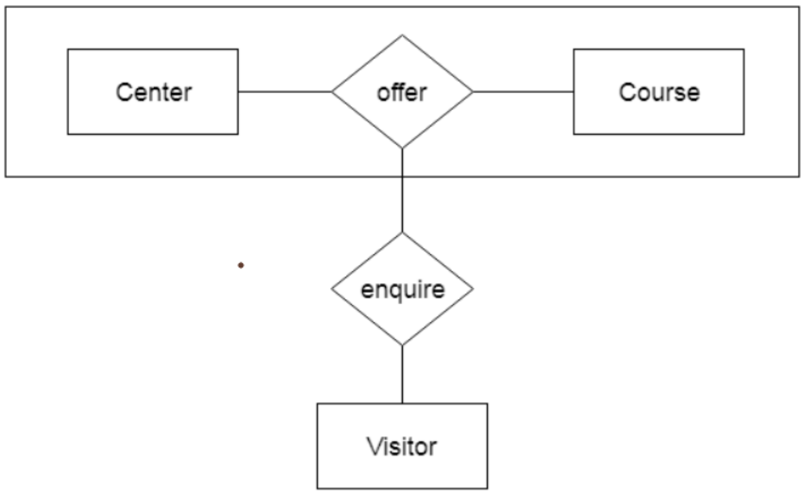
Specialization Example



Specialization

In the above diagram, we can see that we have a higher level entity "Employee" which we have divided in sub entities "Technician", "Engineer" & "Accountant". All of these are just an employee of a company, however their role is completely different and they have few different attributes. Just for the example, I have shown that Technician handles service requests, Engineer works on a project and Accountant handles the credit & debit details. All of these three employee types have few attributes common such as name & salary which we had left associated with the parent entity "Employee" as shown in the above diagram.

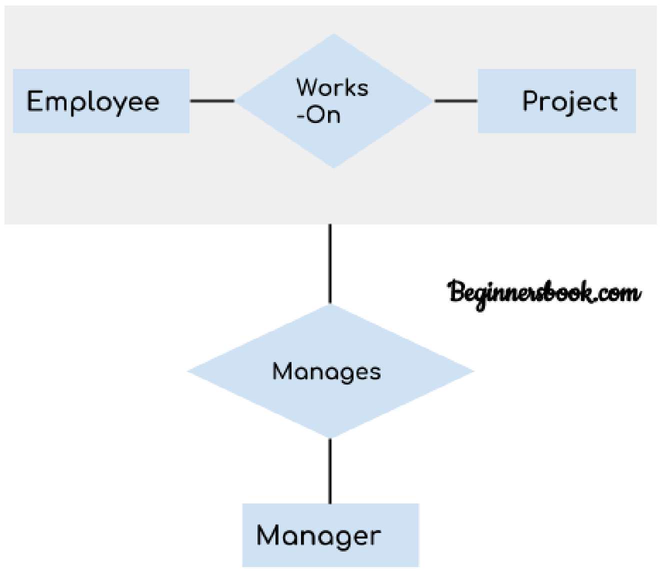
For example: Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.



Aggregation

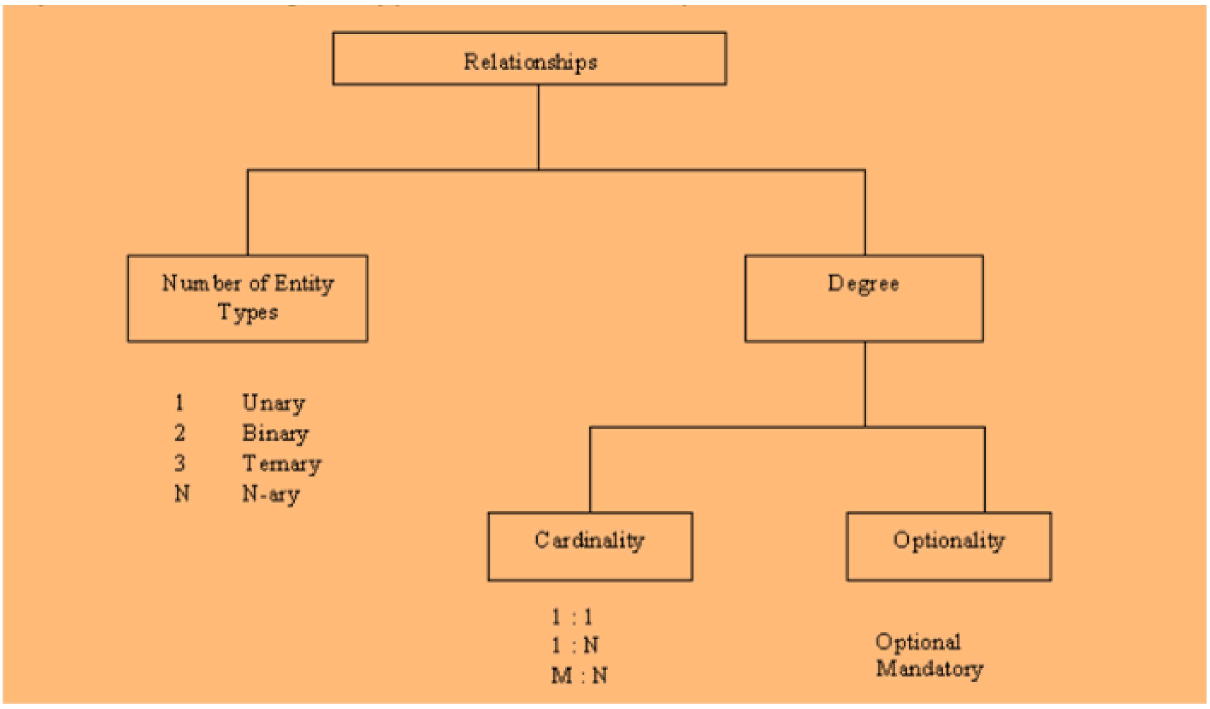
Aggregation is a process in which a single entity alone is not able to make sense in a relationship so the relationship of two entities acts as one entity. I know it sounds confusing but don't worry the example we will take, will clear all the doubts.

Aggregration Example



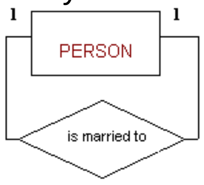
In real world, we know that a manager not only manages the employee working under them but he has to manage the project as well. In such scenario if entity "Manager" makes a "manages" relationship with either "Employee" or "Project" entity alone then it will not make any sense because he has to manage both. In these cases the relationship of two entities acts as one entity. In our example, the relationship “Works-On” between “Employee” & “Project” acts as one entity that has a relationship “Manages” with the entity “Manager”

Relationships

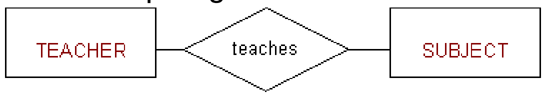


The ***degree*** of a relationship is the number of entity types that participate in the relationship. The three most common relationships in ER models are ***Binary***, ***Unary*** and ***Ternary***

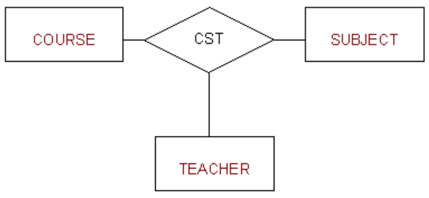
A **unary relationship** is when both participants in the relationship are the same entity.



A **binary relationship** is when two entities participate and is the most common relationship degree



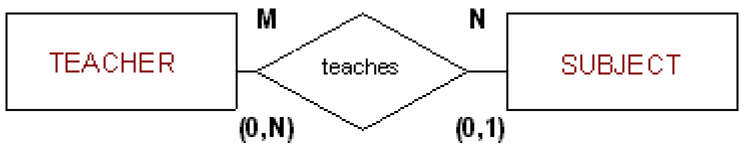
A **ternary relationship** is when three entities participate in the relationship



Cardinality expresses the minimum and maximum number of entity occurrences associated with one occurrence of the related entity. In the ERD, cardinality is indicated by placing the appropriate numbers beside the entities, using the format (x,y). The first value represents the minimum number of associated entities, while the second value represents the maximum number of associated entities

However, keep in mind that the DBMS cannot handle the implementation of the cardinalities at the table level—that capability is provided by the application software or by triggers

connectivity is used to describe the relationship classification. Just to show the relation existence and type of relation between entities.



Mandatory/Optional Relationships:

Participation by an entity in a relationship may be **optional** or **mandatory**.

REFER MAM’S NOTES FOR DRAWING ER DIAGRAM