DATA MODELS (7 Hours)

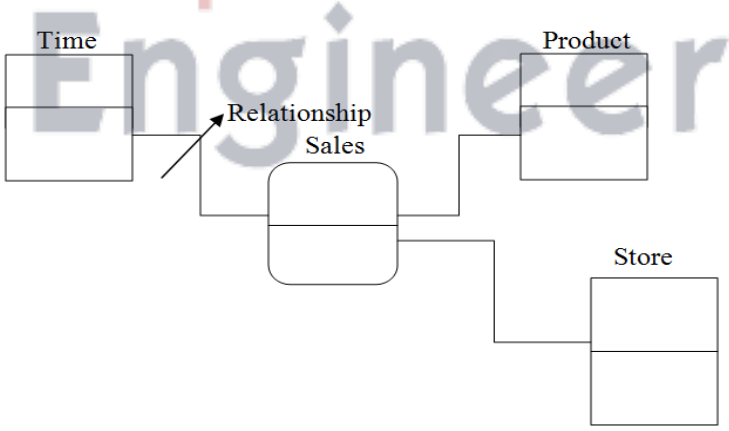
Data models define how the logical structure of database is modeled. It is the fundamental entities to introduce abstraction in a DBMS. It defines how data is connected to each other and how they are processed and stored inside the system. It is simply a diagram that displays a set of tables and the relation between them. Stages of data models are :

1. **Conceptual Data Model**
2. **Logical Data Model**
3. **Physical Data Model**

Conceptual, logical and physical model or ERD are three different ways of modeling data in a domain. While they all contain entities and relationships, they differ in the purposes they are created for and audiences they are meant to target. A general understanding to the three models is that, business analyst uses conceptual and logical model for modeling the data required and produced by system from a business angle, while database designer refines the early design to produce the physical model for presenting physical database structure ready for database construction

Conceptual Data Model

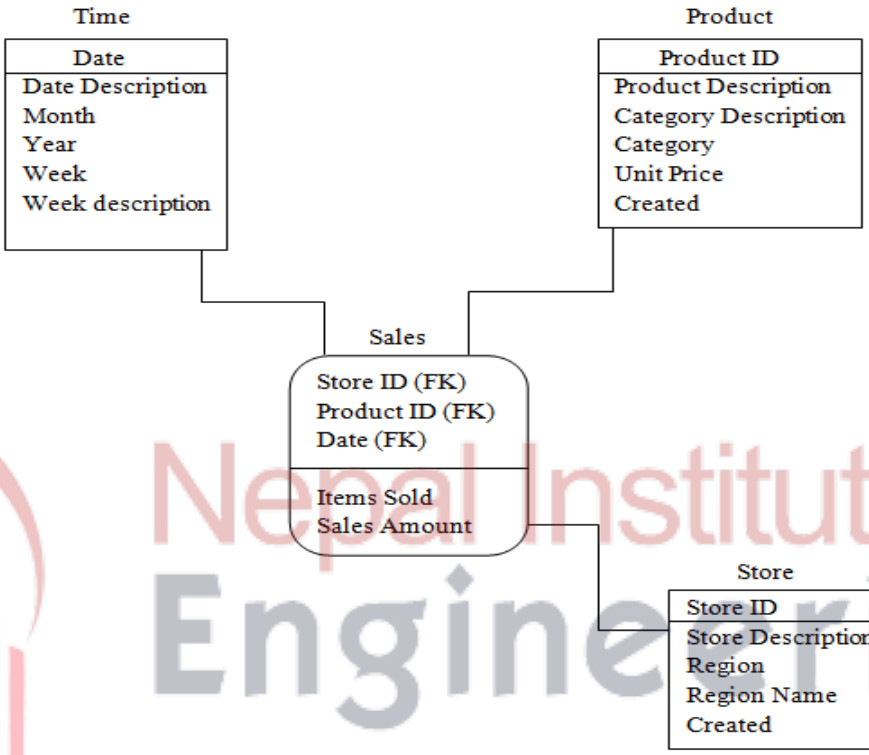
Entities : Time
Product
Sales
Store



- It is highly abstract i.e. we don't have much details.
- It can be understood easily by both technical and non technical user.
- It can be easily enhanced. (Note : From above diagram , entities Time, Product and Store have direct relationship with Sales entity, due to which lot of information can be obtained by looking at the conceptual data model. And since it is not a digital document , it can be easily enhanced.)
- Only 'entities' are visible.
- Relationship among entities are abstract too. (That is even relationship details are hidden.)

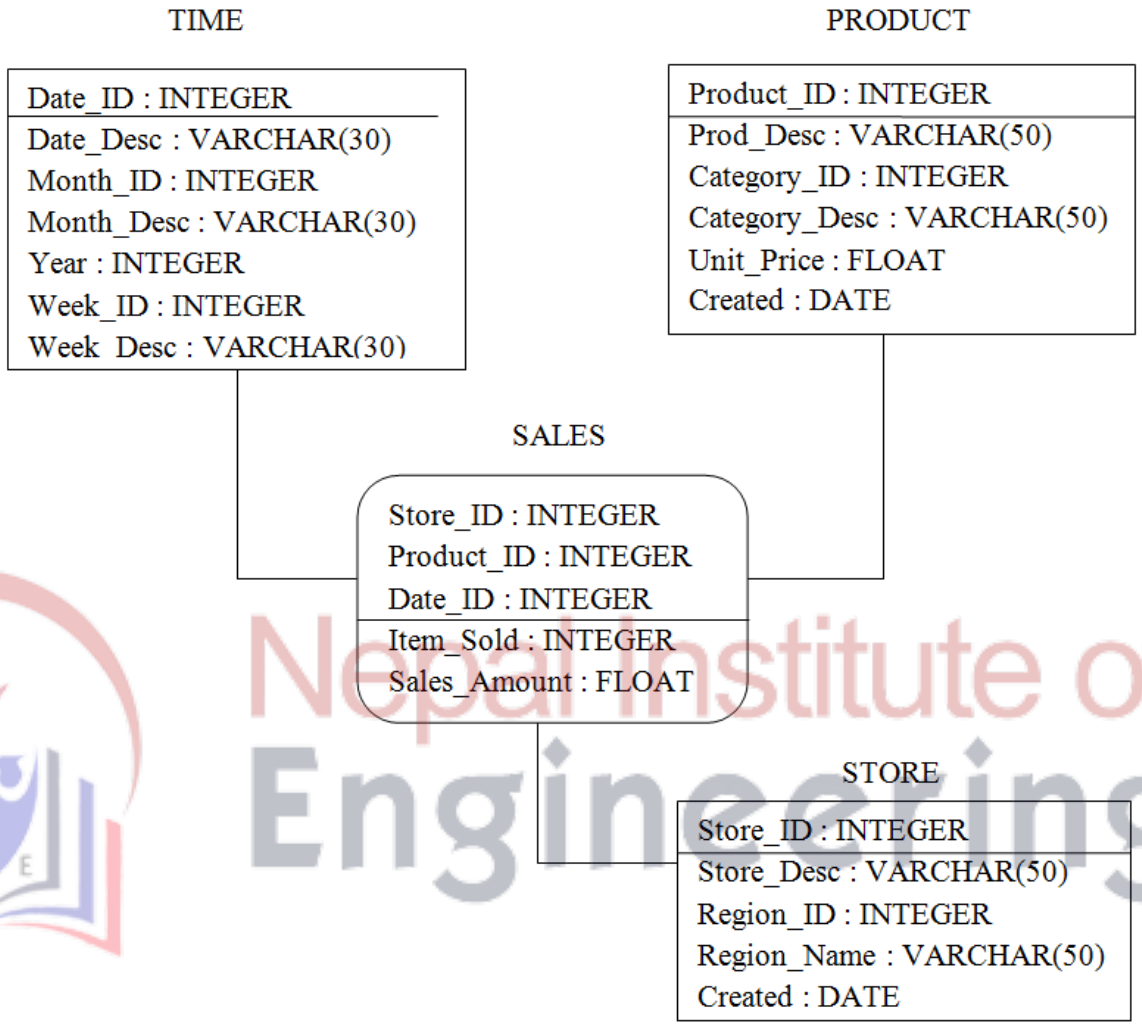
- Not any software tools are required to define data models. (Since , conceptual data models can be written on a piece of paper or white board, there is not any requirement of software tools.)

Logical Data Model



- Attributes of each entity are present, which are further identified as keyed and non-keyed attributes.
- Keyed attributes define uniqueness of entity.
- All the keyed attributes are placed above line as shown in figure above and all non-keyed attributes are placed below line.
- Primary key - foreign key relationship are clearly defined in logical data model. In above example Date, Product ID, Store ID are primary key for Time, Product and Store entity respectively, and can also be foreign key for Sales entity.
- User friendly attributes are present i.e. it becomes easier for both technical and non-technical user to understand.
- It provides more detailed information as compared to conceptual data model.
- It does not depend on any specific database means you can take this logical model and implement on any database software i.e. ORACLE, SQLSERVER etc.

Physical Data Model



- In this model, entities referred to as tables and attributes are referred to as columns.
- It provides database compatible table name and database specific data types.
- It is difficult for user to understand
- It requires more effort to be enhanced in comparison to logical data model.
- It includes indexes, constraints, triggers and other DB objects.
- It is difficult to port to a different databases once design is finalized.

ER MODEL

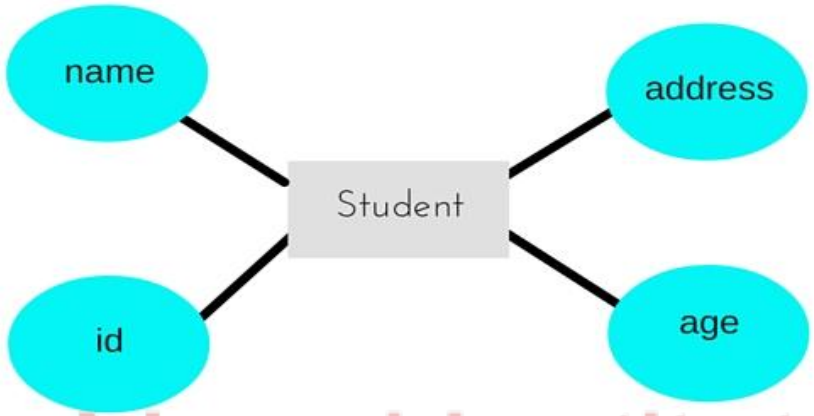
In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes.

Different entities are related using relationships.

E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand.

This model is good to design a database, which can then be turned into tables in relational model(explained below). While formulating real world scenario into the database model, ER model creates entity set, relationship set, general attributes and constraints.

Let's take an example, If we have to design a School Database, then **Student** will be an **entity** with **attributes** name, age, address etc. As **Address** is generally complex, it can be another **entity** with **attributes** street name, pincode, city etc, and there will be a relationship between them. Relationships can also be of different types.



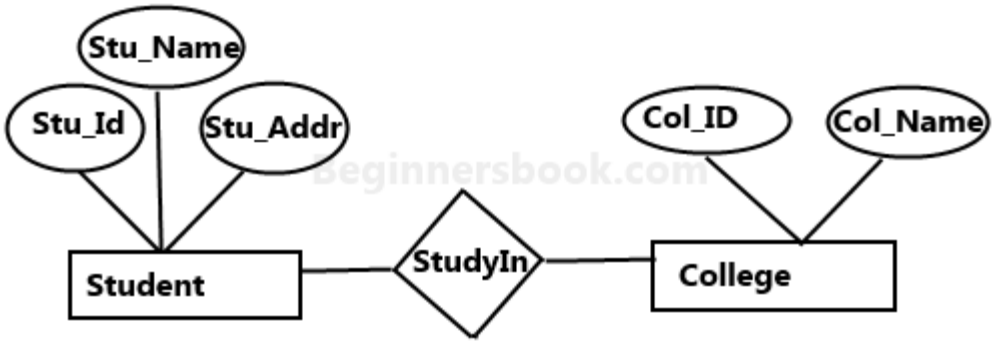
An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let's have a look at a simple ER diagram to understand this concept.

In the following diagram we have two entities Student and College and their relationship. The relationship between Student and College is many to one as a college can have many students however a student cannot study in multiple colleges at the same time. Student entity has attributes such as Stu_Id, Stu_Name & Stu_Addr and College entity has attributes such as

Col_ID

Col_Name



Sample E-R Diagram

Here are the geometric shapes and their meaning in an E-R Diagram. We will discuss these terms in detail in the next section(Components of a ER Diagram) of this guide so don't worry too much about these terms now, just go through them once.

Entity-relationship model is a model used for design and representation of relationships between data.

The main data objects are termed as Entities, with their details defined as attributes, some of these attributes are important and are used to identify the entity, and different entities are related using relationships.

In short, to understand about the ER Model, we must understand about:

- Entity & Entity Set
- Relationship & Relationship Set
- What are Attributes? And Types of Attributes.
- Keys

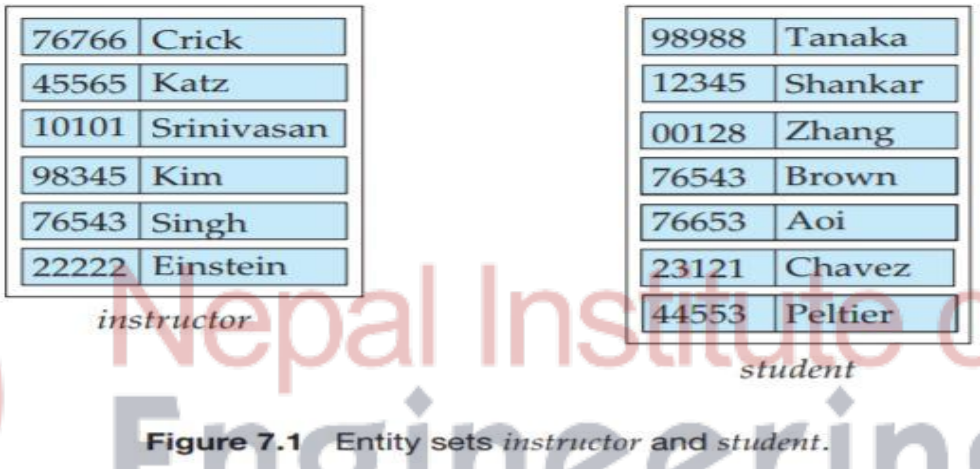
Entity & Entity Set

An entity is a "thing" or "object" in the real world that is distinguished from all other objects. An entity is represented as rectangle in an ER diagram. An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course. An entity has a set of properties, and the values for some set of properties may uniquely identify an entity. For example, a person may have ***person_id*** property whose value uniquely identifies that person. Thus, the value 677-89-9011 for ***person_id*** would uniquely identify one particular person in the university.

An entity set is a set of entities of the same type that share the same properties, or attributes. All entity set has a key. The set of all people who are instructors at a given university, for example, can be defined as the entity set ***instructor***. Similarly, the entity set ***student*** might represent the set of all students in the university. Entity sets do not need to be disjoint. For example, it is possible to define the entity set of all people in a university (person). A person entity may be an

instructor entity, a ***student*** entity, both, or neither. An entity is represented by the set of **attributes.** Attributes are descriptive properties possessed by each member of an entity set. The designation of an attribute for an entity set expresses that the database stores similar information concerning each entity in the entity set; however, each entity may have its own value for each attribute. Possible attributes for ***student*** entity set may be ID, name, roll, stream. In real life , there would be further attributes, such as address, age etc.

A database for university may include a number of other entity sets. For example, in addition to keeping track of instructors and students, the university also has information about courses, which are represented by the entity set ***course*** with attributes *course_id, title, dept_name* and *credits*. In a real setting, a university database may keep dozens of entity sets.



76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

instructor

98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

Figure 7.1 Entity sets *instructor* and *student*.

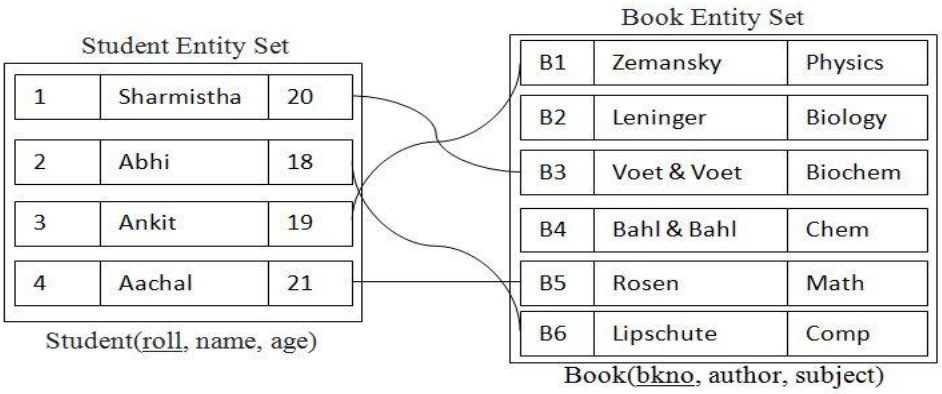
Relationship & Relationship Set

Relationship is an association among several entities. For example, If we define a relationship ***advisor*** that associates instructor **Suresh** with student **Shankar**, this relationship specifies that **Suresh** is an advisor to student **Shankar**.

A relationship set is a set of relationships of same type. It is a mathematical relation of n>=2 (possibly non distinct) entity sets. If E1, E2, E3,...., En are entity sets, then a relationship set R is a subset of

$$\{(e1, e2, e3, ..., en) \mid e1 \in E1, e2 \in E2, ..., en \in En\}$$

Where (e1, e2, e3, ..., en) is a relationship.



So we have two different entity sets such as ***Student entity set*** and ***Book entity set***. From above figure it is clear that we have schema name ***Student*** with attributes ***roll, name, age*** and another schema ***Book*** with attributes ***bkno, author and subject***. Here ***roll*** is primary key attribute and ***bkno*** is another primary key attribute of schema ***Student*** and ***Book*** respectively. There is relationship between ***Student*** and ***Book***, where students has taken different books. And this relationship(mapping) information can be kept in respective relationship. So here mapping information has been kept in relationship name ***issue***. And relationship ***issue*** will be having prime attributes i.e., ***roll*** and ***bkno*** of ***Student*** and ***Book*** schema respectively. That is all the attributes which will be coming in the relationship should be the attributes of respective connected entities.

So from above figure , we have 4 records in a relationship table.

issue table

Roll	bkno
1	B3
2	B6
3	B1
4	B5

Here from table it is clear that , we have alike relationships, that is, (1 , B3), (2 , B6), (3 , B1) and (4 , B5) are 4 alike relationship and these relationships form relationship set.

Set of attributes in a relationship R will be

- Primary_key(E1) ∪ Primary_key(E2) ∪ ..... ∪ Primary_Key(En)

If a relationship has descriptive attributes associated with it and if these attributes are {a1, a2, a3, ..., am} then the relationship will have following set of attributes

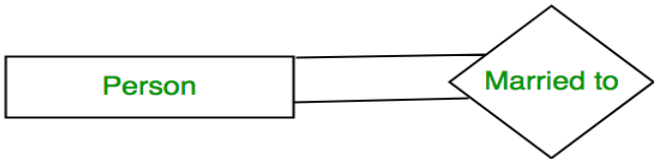
- Primary_Key(E1) ∪ Primary_Key(E2) ∪ .... ∪ Primary_Key(En) ∪ {a1, a2, ..., am}
  issue(roll, bkno, access_date)

(If we have descriptive attributes like ***access_date*** in a relationship. Descriptive properties means ***access_date*** can't be kept in ***Student*** or ***Book*** schema.)

Degree of a relationship set:

The number of different entity sets **participating in a relationship** set is called as degree of a relationship set.

1. **Unary Relationship**
   When there is **only ONE entity set participating in a relation**, the relationship is called as unary relationship. For example, one person is married to only one person.



2. **Binary Relationship**
   When there are **TWO entities set participating in a relation**, the relationship is called as binary relationship. For example, Student is enrolled in Course.

3. **n-ary Relationship**
   When there are n entities set participating in a relation, the relationship is called as n-ary relationship.

Constraints in Data Models

An ER enterprise schema may define certain constraints to which the contents of a database must confirm. We have **mapping cardinalities** and **participation cardinalities**.

Mapping Cardinalities: Mapping Cardinalities, or cardinality ratios express the number of entities to which another entity can be associated via a relationship set. Or, **the number of times an entity of an entity set participates in a relationship** set is known as cardinality.

Mapping cardinalities are most useful in describing binary relationship sets, although they can contribute to the description of relationship sets that involve more than two entity sets. For Binary relationship set R between entity sets A and B, the mapping cardinality must be of the following :

One-to-one. An entity in *A* is associated with *at most* one entity in *B*, and an entity in *B* is associated with *at most* one entity in *A*.

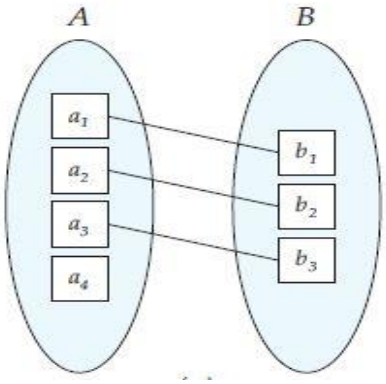


figure : One - To - One

For example, a person has only one passport and a passport is given to one person.

One-to-many. An entity in *A* is associated with any number (zero or more) of entities in *B*. An entity in *B*, however, can be associated with *at most* one entity in *A*.

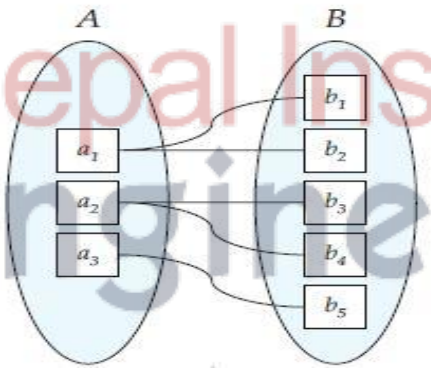


Figure : One - To - Many

For example – a customer can place many orders but a order cannot be placed by many customers.

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 (zero or more) of entities in *A*.

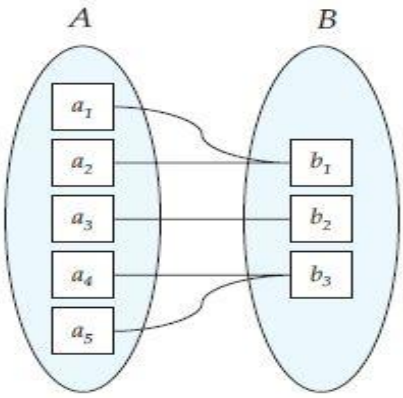


Figure : Many - To - One

For example – many students can study in a single college but a student cannot study in many colleges at the same time.

Many-to-many. An entity in *A* is associated with any number (zero or more) of entities in *B*, and an entity in *B* is associated with any number (zero or more) of entities in *A*.

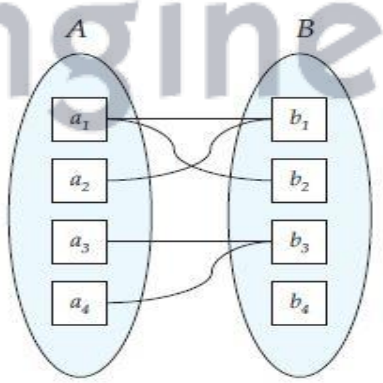


Figure : Many - To - Many

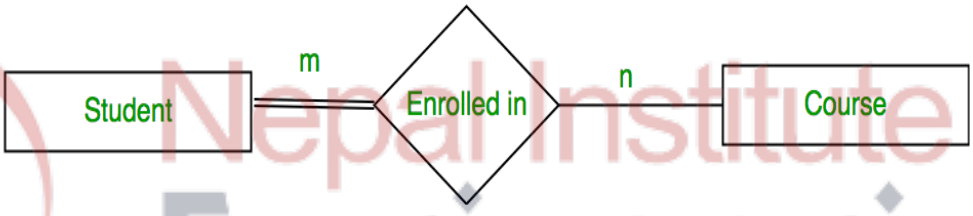
For example, a student can be assigned to many projects and a project can be assigned to many students.

Participation Constraints

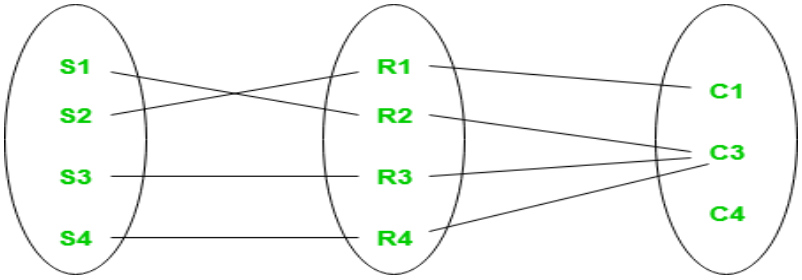
Participation Constraint is applied on the entity participating in the relationship set.

1. **Total Participation –** Each entity in the entity set **must participate** in the relationship. That is, the participation of an entity set E in a relationship set R is said to be ***total*** if every entity in E participates in at least one relationship in R. If each student must enroll in a course, the participation of student will be total. Total participation is shown by double line in ER diagram.

2. **Partial Participation –** The entity in the entity set **may or may NOT participat**e in the relationship. That is, if only some entities in E participates in relationships in R, the participation of entity set E in relationship R is said to be ***partial.*** If some courses are not enrolled by any of the student, the participation of course will be partial.
   The diagram depicts the 'Enrolled in' relationship set with Student Entity set having total participation and Course Entity set having partial participation.



Using set, it can be represented as,



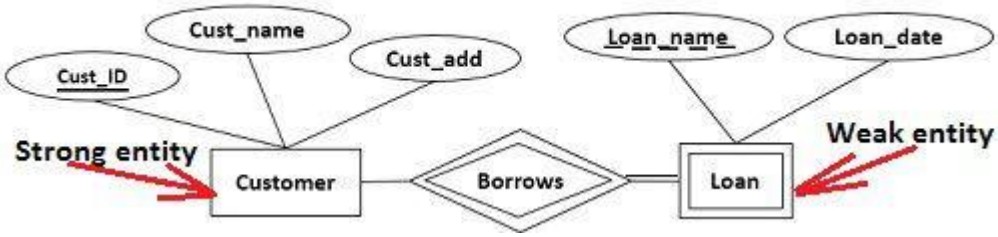
Every student in Student Entity set is participating in relationship but there exists a course C4 which is not taking part in the relationship.

Strong And Weak Entity Sets

Strong Entity : The Strong Entity is the one whose existence does not depend on the existence of any other entity in a schema. It is denoted by a single rectangle. A strong entity always has the primary key in the set of attributes that describes the strong entity. It indicates that each entity in a strong entity set can be uniquely identified.

Set of similar types of strong entities together forms the Strong Entity Set. A strong entity holds the relationship with the weak entity via an Identifying Relationship, which is denoted by double

diamond in the ER diagram. On the other hands, the relationship between two strong entities is denoted by a single diamond and it is simply called as a relationship.
Let us understand this concept with the help of an example; a customer borrows a loan. Here we have two entities first a customer entity, and second a loan entity.



Observing the ER-diagram above, for each loan, there should be at least one borrower otherwise that loan would not be listed in Loan entity set. But even if a customer does not borrow any loan it would be listed in Customer entity set. So we can conclude that a customer entity does not depend on a loan entity.

Cust_ID	Cust_name	Cust_add
101	John	Xyzzy
103	Ruby	Pqr
109	John	Uvfw

Customer Entity Set

The second thing you can observe that the Customer entity has as primary key Cust_ID which uniquely identify each entity in Customer Entity set. This makes Customer entity a strong entity on which a loan entity depends.

Weak Entity: A Weak entity is the one that depends on its owner entity i.e. a strong entity for its existence. A weak entity is denoted by the double rectangle. Weak entity do not have the primary key instead it has a partial key that uniquely discriminates the weak entities. The primary key of a weak entity is a composite key formed from the primary key of the strong entity and partial key of the weak entity.

The collection of similar weak entities is called Weak Entity Set. The relationship between a weak entity and a strong entity is always denoted with an Identifying Relationship i.e. double diamond.

For further illustration let us discuss the above example, this time from weak entity's point of view. We have Loan as our weak entity, and as I said above for each loan there must be at least one borrower. You can observe in the loan entity set, no customer has borrowed a car loan and hence, it has totally vanished from loan entity set. For the presence of car loan in loan entity set, it must have been borrowed by a customer. In this way, the weak Loan entity is dependent on the strong Customer entity.

Loan_name	Loan_date	Amount
Home	20/11/2015	20000
Education	5/10/2015	10000
Home	20/11/2015	20000

Loan Entity Set

The second thing, we know is a weak entity does not have a primary key. So here Loan_name, the partial key of the weak entity and Cust_ID primary key of customer entity makes primary key of loan entity.

In the Loan entity set, we have two exactly same entities i.e. a **Home loan on date 20/11/2015 with amount 20000.** Now how to identify who had borrowed them this can be done with the help of primary key of the weak entity (Loan_name + Cust_ID). So, it will be determined that one home loan is borrowed by Customer 101 Jhon and other by Customer 103 Ruby. This is how the composed primary key of weak entity identify each entity in weak entity set.

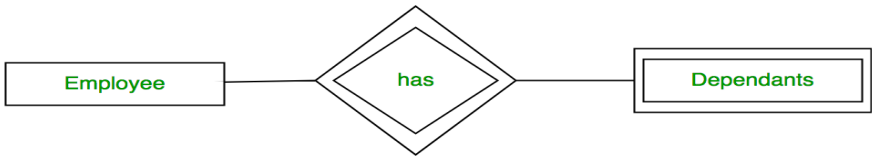
BASIS FOR COMPARISON	STRONG ENTITY	WEAK ENTITY
Basic	The Strong entity has a primary key.	The weak entity has a partial discriminator key.
Depends	The Strong entity is independent of any other entity in a schema.	Weak entity depends on the strong entity for its existence.
Denoted	Strong entity is denoted by a single rectangle.	Weak entity is denoted with the double rectangle.
Relation	The relation between two strong entities is denoted by a single diamond simply called relationship.	The relationship between a weak and a strong entity is denoted by Identifying Relationship denoted with double diamond.
Participation	Strong entity may or may not have total participation in the relationship.	Weak entity always has total participation in the identifying relationship shown by double line.

Weak Entity Type and Identifying Relationship

As discussed before, an entity type has a key attribute which uniquely identifies each entity in the entity set. But there exists **some entity type for which key attribute can't be defined**. These are called Weak Entity type.

For example, A company may store the information of dependants (Parents, Children, Spouse) of an Employee. But the dependents don't have existence without the employee. So Dependent will be weak entity type and Employee will be Identifying Entity type for Dependant.

A weak entity type is represented by a double rectangle. The participation of weak entity type is always total. The relationship between weak entity type and its identifying strong entity type is called identifying relationship and it is represented by double diamond.



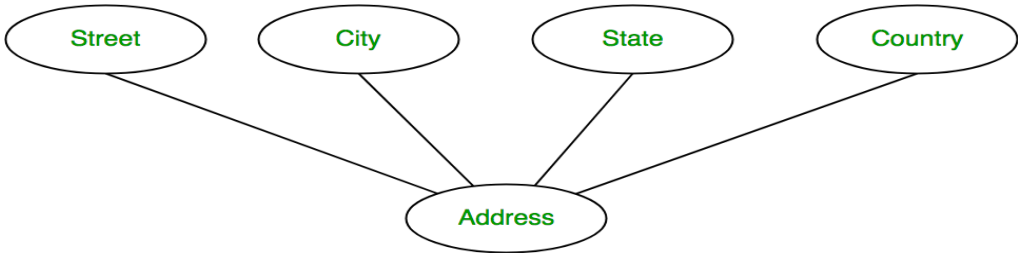
Attributes And Keys

Attribute(s) : Attributes are the **properties which define the entity type**. For example, Roll_No, Name, DOB, Age, Address, Mobile_No are the attributes which defines entity type Student. In ER diagram, attribute is represented by an oval.

1. Key Attribute – The attribute which **uniquely identifies each entity** in the entity set is called key attribute.For example, Roll_No will be unique for each student. In ER diagram,

key attribute is represented by an oval with underlying lines.

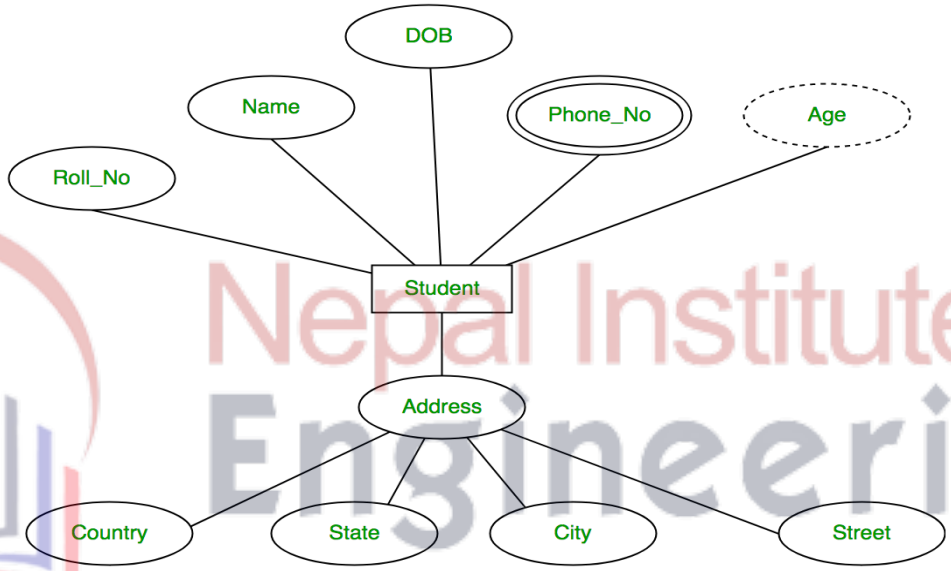
2.**Composite Attribute –** An attribute **composed of many other attribute** is called as composite attribute. For example, Address attribute of student Entity type consists of Street, City, State, and Country. In ER diagram, composite attribute is represented by an oval comprising of ovals.



3. Multivalued Attribute – An attribute consisting **more than one value** for a given entity. For example, Phone_No (can be more than one for a given student). In ER diagram, multivalued attribute is represented by double oval.

4. Derived Attribute – An attribute which can be **derived from other attributes** of the entity type is known as derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, derived attribute is represented by dashed oval.

The complete entity type **Student** with its attributes can be represented as:



Database Keys

Keys are very important part of Relational database model. They are used to establish and identify relationships between tables and also to uniquely identify any record or row of data inside a table.

A Key can be a single attribute or a group of attributes, where the combination may act as a key.

Why we need a Key?

In real world applications, number of tables required for storing the data is huge, and the different tables are related to each other as well.

Also, tables store a lot of data in them. Tables generally extends to thousands of records stored in them, unsorted and unorganised.

Now to fetch any particular record from such dataset, you will have to apply some conditions, but what if there is duplicate data present and every time you try to fetch some data by applying certain condition, you get the wrong data. How many trials before you get the right data?

To avoid all this, **Keys** are defined to easily identify any row of data in a table.

Let's try to understand about all the keys using a simple example.

student_id	name	phone	age
1	Akon	9876723452	17
2	Akon	9991165674	19
3	Bkon	7898756543	18
4	Ckon	8987867898	19
5	Dkon	9990080080	17

Let's take a simple **Student** table, with fields student_id, name, phone and age.

Super Key : It is defined as a set of attributes within a table that can uniquely identify each record within a table. Super Key is a superset of Candidate key.

In the table defined above super key would include student_id, (student_id, name), phoneetc.

Confused? The first one is pretty simple as student_id is unique for every row of data, hence it can be used to identity each row uniquely.

Next comes, (student_id, name), now name of two students can be same, but their student_idcan't be same hence this combination can also be a key.

Similarly, phone number for every student will be unique, hence again, phone can also be a key.

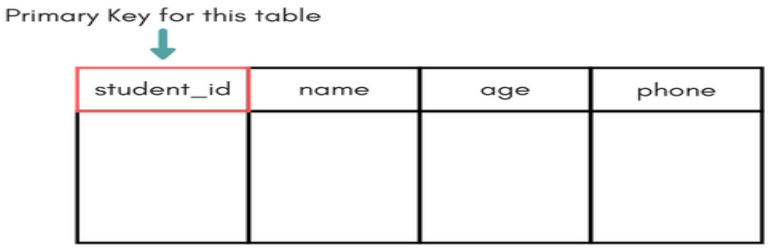
So they all are super keys.

Candidate Key : Candidate keys are defined as the minimal set of fields which can uniquely identify each record in a table. It is an attribute or a set of attributes that can act as a Primary Key for a table to uniquely identify each record in that table. There can be more than one candidate key.

In our example, student_id and phone both are candidate keys for table **Student**.

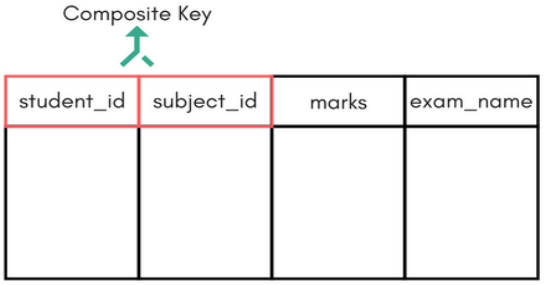
- A candiate key can never be NULL or empty. And its value should be unique.
- There can be more than one candidate keys for a table.
- A candidate key can be a combination of more than one columns(attributes).

Primary Key : Primary key is a candidate key that is most appropriate to become the main key for any table. It is a key that can uniquely identify each record in a table.



For the table **Student** we can make the student_id column as the primary key.

Composite Key : Key that consists of two or more attributes that uniquely identify any record in a table is called **Composite key**. But the attributes which together form the **Composite key** are not a key independently or individually.



Score Table - To save scores of the student for various subjects.

In the above picture we have a **Score** table which stores the marks scored by a student in a particular subject.
In this table student_id and subject_id together will form the primary key, hence it is a composite key.

Secondary or Alternative key : The candidate key which are not selected as primary key are known as secondary keys or alternative keys.

Non-key Attributes : Non-key attributes are the attributes or fields of a table, other than **candidate key** attributes/fields in a table.

Non-prime Attributes : Non-prime Attributes are attributes other than **Primary Key attribute(s).**.

E-R Diagram

ER Diagram is a visual representation of data that describes how data is related to each other. In ER Model, we disintegrate data into entities, attributes and setup relationships between entities, all this can be represented visually using the ER diagram.

Components of ER Diagram

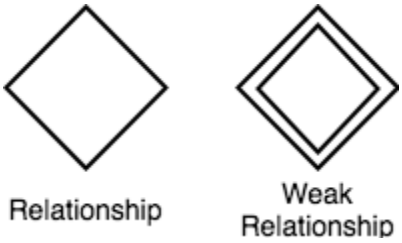
Entitiy, Attributes, Relationships etc form the components of ER Diagram and there are defined symbols and shapes to represent each one of them.
Let's see how we can represent these in our ER Diagram.

Entity

Simple rectangular box represents an Entity.

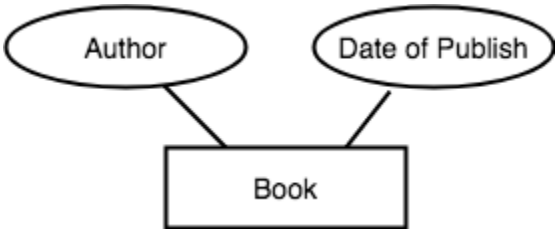
Relationships between Entities - Weak and Strong

Rhombus is used to setup relationships between two or more entities.



Attributes for any Entity

Ellipse is used to represent attributes of any entity. It is connected to the entity.



Weak Entity

A weak Entity is represented using double rectangular boxes. It is generally connected to another entity.

Key Attribute for any Entity

To represent a Key attribute, the attribute name inside the Ellipse is underlined.

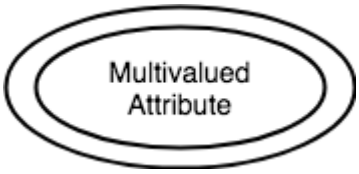
Derived Attribute for any Entity

Derived attributes are those which are derived based on other attributes, for example, age can be derived from date of birth.

To represent a derived attribute, another dotted ellipse is created inside the main ellipse.

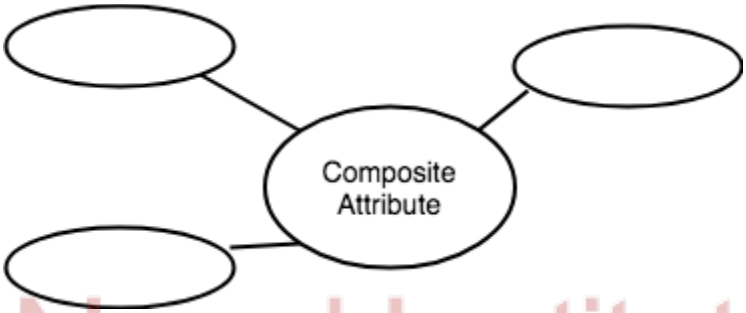
Multivalued Attribute for any Entity

Double Ellipse, one inside another, represents the attribute which can have multiple values.



Composite Attribute for any Entity

A composite attribute is the attribute, which also has attributes.



ER Diagram: Entity

An **Entity** can be any object, place, person or class. In ER Diagram, an **entity** is represented using rectangles. Consider an example of an Organisation- Employee, Manager, Department, Product and many more can be taken as entities in an Organisation.

The yellow rhombus in between represents a relationship.

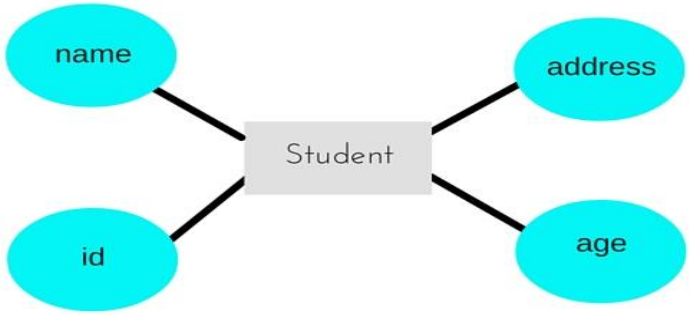
ER Diagram: Weak Entity

Weak entity is an entity that depends on another entity. Weak entity doesn't have anay key attribute of its own. Double rectangle is used to represent a weak entity.

ER Diagram: Attribute

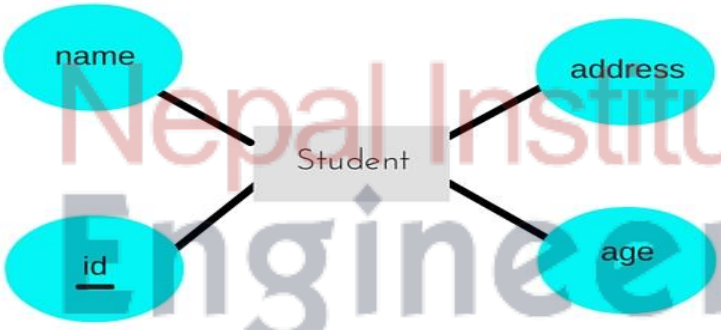
An Attribute describes a property or characteristic of an entity.

For example, Name, Age, Address etc can be attributes of a Student. An attribute is represented using eclipse.



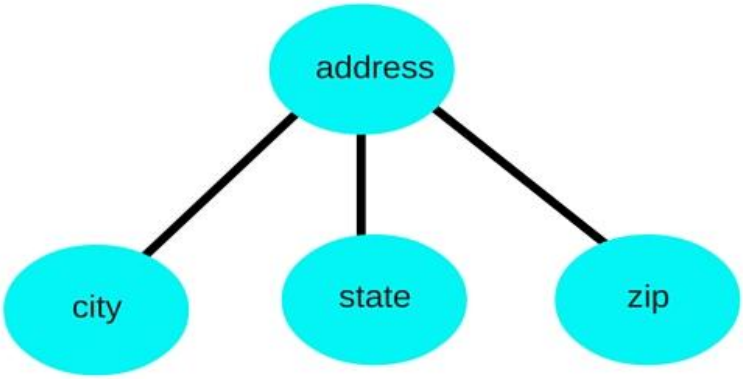
ER Diagram: Key Attribute

Key attribute represents the main characteristic of an Entity. It is used to represent a Primary key. Ellipse with the text underlined, represents Key Attribute.



ER Diagram: Composite Attribute

An attribute can also have their own attributes. These attributes are known as **Composite** attributes.



ER Diagram: Relationship

A Relationship describes relation between **entities**. Relationship is represented using diamonds or rhombus.

There are three types of relationship that exist between Entities.

- Binary Relationship
- Recursive Relationship
- Ternary Relationship

ER Diagram: Binary Relationship

Binary Relationship means relation between two Entities. This is further divided into three types.

One to One Relationship

This type of relationship is rarely seen in real world.

The above example describes that one student can enroll only for one course and a course will also have only one Student. This is not what you will usually see in real-world relationships.

One to Many Relationship

The below example showcases this relationship, which means that 1 student can opt for many courses, but a course can only have 1 student. Sounds weird! This is how it is.

Many to One Relationship

It reflects business rule that many entities can be associated with just one entity. For example, Student enrolls for only one Course but a Course can have many Students.

Many to Many Relationship

The above diagram represents that one student can enroll for more than one courses. And a course can have more than 1 student enrolled in it.

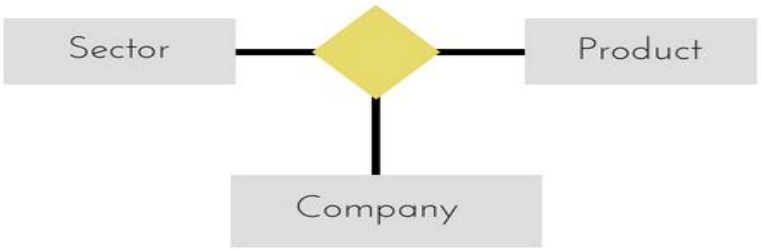
ER Diagram: Recursive Relationship

When an Entity is related with itself it is known as **Recursive** Relationship.

ER Diagram: Ternary Relationship

Relationship of degree three is called Ternary relationship.

A Ternary relationship involves three entities. In such relationships we always consider two entities together and then look upon the third.



- The above relationship involves 3 entities.
- Company operates in Sector, producing some Products.

For example, in the diagram above, we have three related entities, **Company**, **Product** and **Sector**. To understand the relationship better or to define rules around the model, we should relate two entities and then derive the third one.

A **Company** produces many **Products**/ each product is produced by exactly one company. A **Company** operates in only one **Sector** / each sector has many companies operating in it. Considering the above two rules or relationships, we see that although the complete relationship involves three entities, but we are looking at two entities at a time.

The Enhanced ER Model

As the complexity of data increased in the late 1980s, it became more and more difficult to use the traditional ER Model for database modelling. Hence some improvements or enhancements were made to the existing ER Model to make it able to handle the complex applications better.

Hence, as part of the **Enhanced ER Model**, along with other improvements, three new concepts were added to the existing ER Model, they were:

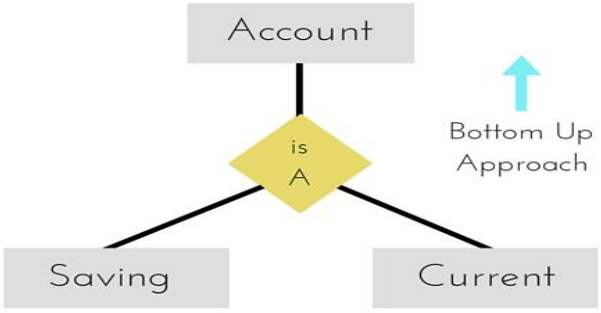
- Generalization
- Specialization
- Aggregation

Let's understand what they are, and why were they added to the existing ER Model.

Generalization

Generalization is a bottom-up approach in which two lower level entities combine to form a higher level entity. In generalization, the higher level entity can also combine with other lower level entities to make further higher level entity.

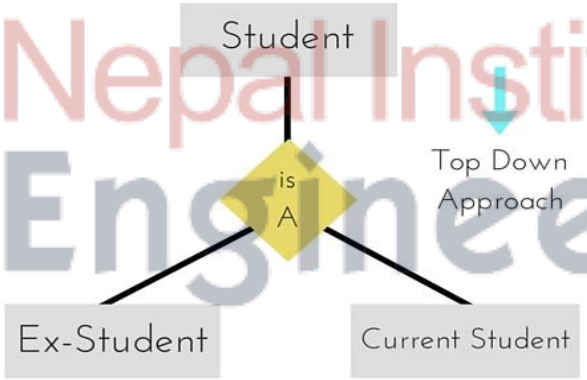
It's more like Superclass and Subclass system, but the only difference is the approach, which is bottom-up. Hence, entities are combined to form a more generalized entity, in other words, sub-classes are combined to form a super-class.



For example, **Saving** and **Current** account types entities can be generalized and an entity with name **Account** can be created, which covers both.

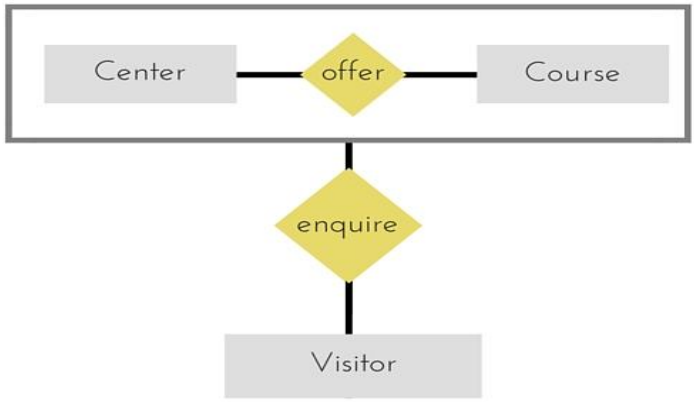
Specialization

Specialization is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entity. In specialization, a higher level entity may not have any lower-level entity sets, it's possible.

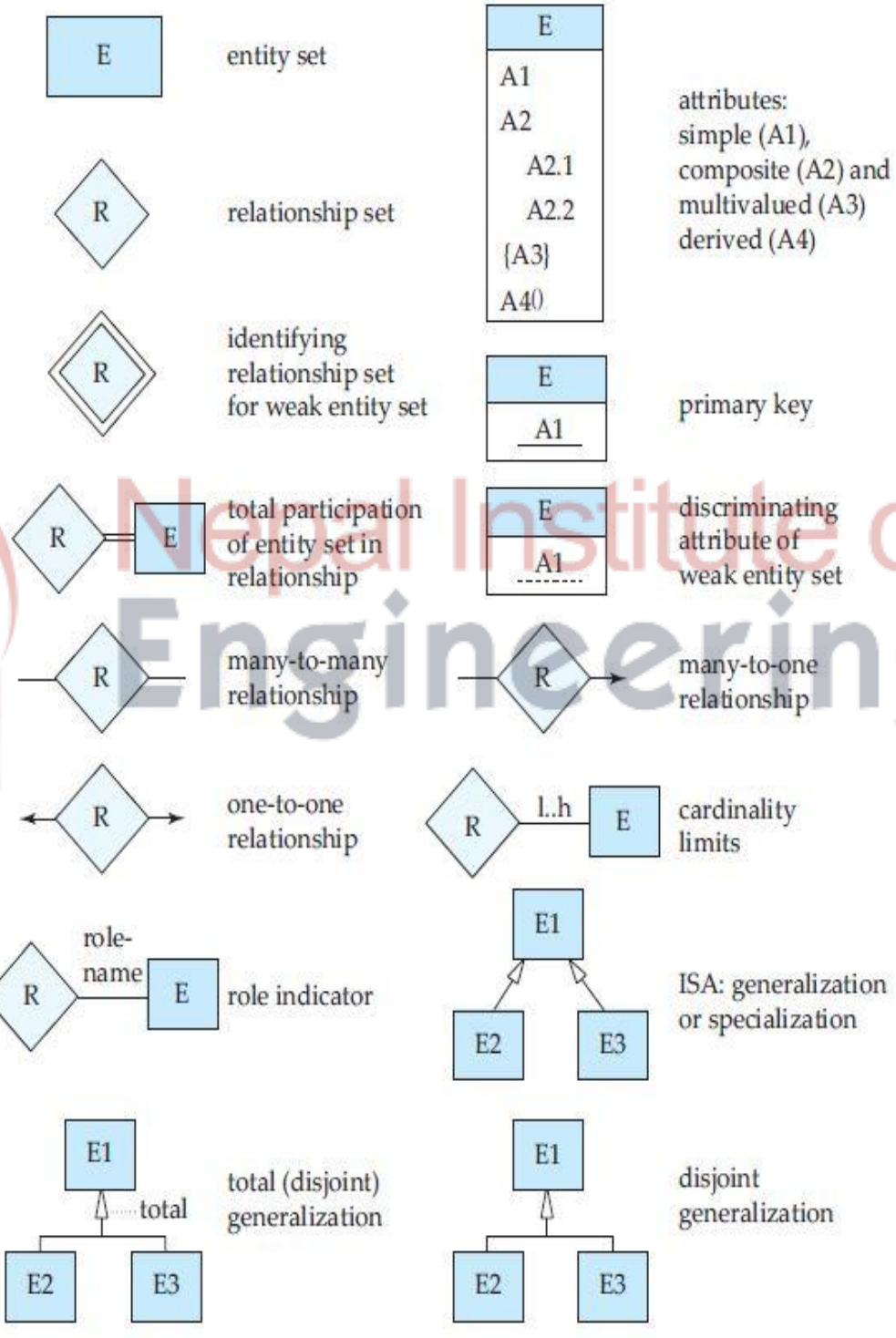


Aggregation

Aggregation is a process when relation between two entities is treated as a **single entity**.



In the diagram above, the relationship between **Center** and **Course** together, is acting as an Entity, which is in relationship with another entity **Visitor**. Now in real world, if a Visitor or a Student visits a Coaching Center, he/she will never enquire about the center only or just about the course, rather he/she will ask enquire about both.

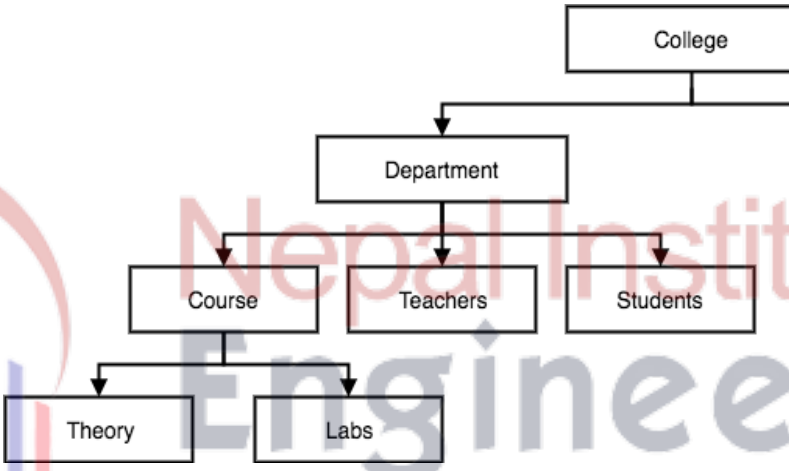


Alternate Data Model (hierarchical, network, graph)

Record based logical Models – Like Object based model, they also describe data at the conceptual and view levels. These models specify logical structure of database with records, fields and attributes.

- **Hierarchical Model**

  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. In this model, a child node will only have a single parent node. This model efficiently describes many real-world relationships like index of a book, recipes etc. 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

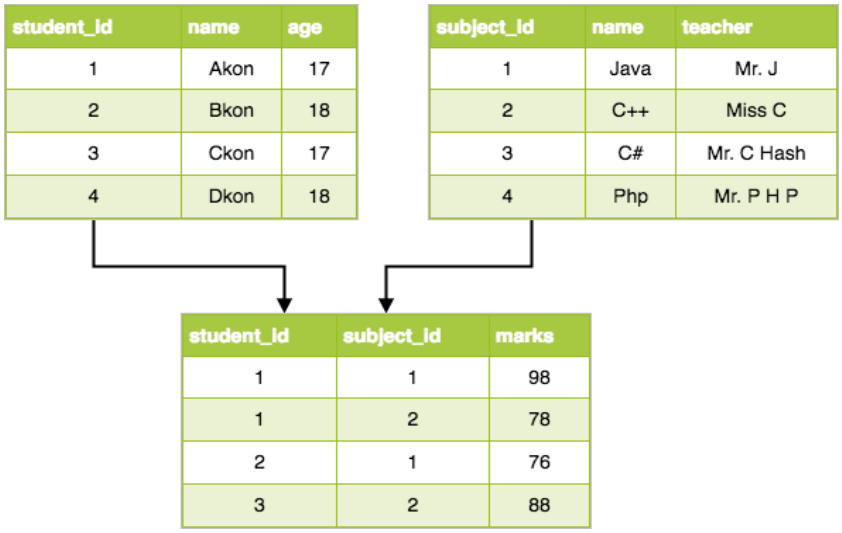
- Simplicity
- Data security & Data integrity
- Efficiency

Disadvantages

- Implementation complexity
- Lack of structural independence

- **Relational Model**

  In this model, data is organized in two-dimensional **tables** and the relationship is maintained by storing a common field. This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, in-fact, we can say the only database model used around the world. The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table. Hence, tables are also known as **relations** in relational model. In the coming tutorials we will learn how to design tables, normalize them to reduce data redundancy and how to use Structured Query language to access data from tables.



student_id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

student_id	subject_id	marks
1	1	98
1	2	78
2	1	76
3	2	88

Advantages

- Flexible & efficiency
- Relationship many-to-many
- Table diagram
- Good management
- Less redundancy
- Searching speed in fast

Disadvantages

- very complex
- not user friendly
- less secure

In relational model, the data & relationships are represented by collection of inter-related tables.

How we design the table in database?

Relation : Defined as table with columns and rows.

It is used for data storage and processing.

It is proposed by E. F. Codd of IBM.

Informal Terms	**Formal Terms**
Table	Relation
Columns or field	Attributes
All possible columns values	Domain values
Rows	Tuples
Table Definition	Schema of relation

How to find the degree of relation?

No of attributes in a relation is called degree.

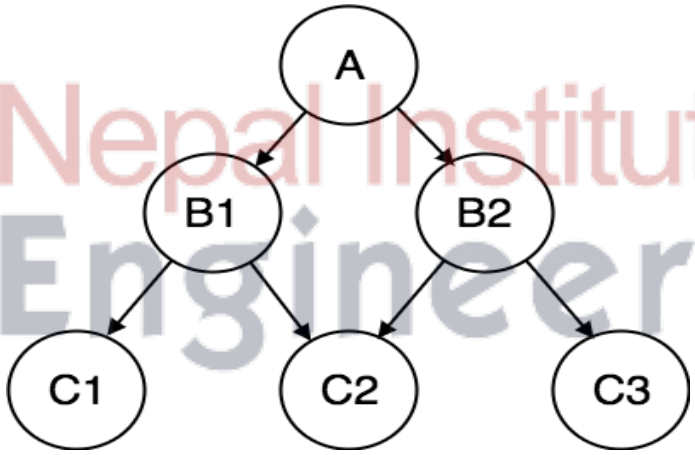
How to find the cardinality of a relation?

No of tuples in a relation is called cardinality.

Properties of Relation

Relation has name(distinct) from all the names in the relationship schema.
Each shell of relation contains exactly one value.
Each attribute has distinct name.

- **Network Model** – Network Model is same as hierarchical model except that it has graph-like structure rather than a tree-based structure. Unlike hierarchical model, this model allows each record to have more than one parent record. 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.



OLD QUESTIONS ASKED PREVIOUSLY(Marks/Year)

1. Identify relevant attributes and construct an ER diagram with proper mapping constraints for a university which has many departments and each department has multiple instructors; one among them is the head of the department. An instructor belongs to only one department, each department offers multiple courses, each of which is taught by a single instructor. A student may enroll for many courses offered by different departments. (6/2075 Bhadra)

2. Define unary relationship along with example. How you convert an ER relationship into relation schema? Explain with examples of different cardinalities. (2+4/2075 Bhadra)

3. Construct an ER-Diagram for the following NFL database. You are given the requirement for a simple database for the National Football league (NFL). The NFL has many teams, and each team has a name, a city, a coach, a captain and a set of players. Each player belongs to only one team and each player has a name, a position (such as left wing, mid fielder or a goalkeeper) a skill level, and a set of injury records. A team captain is also a player and a game is played between 2 teams (referred as host team and guest team) and has a match date (such as June 11,2018) and score (such as 2 to 5).
Explain strong and weak entity sets along with example. (8+4/2075 Baisakh)

4. Define discriminator in ER diagram. Explain different keys used in database design. (4/2074 Bhadra)

5. Draw the Entity-Relationship Diagram (ERD) with appropriate mapping cardinalities for the following scenario.
A production company consists of a machining, fabrication and assembly department. Employees are assigned to different departments. Each department is managed by a manager. Each employee has at most one recognized skill, but a given skill may be possessed by several employees. An employee is able to operate a given machine-type (e.g. lathe, grinder, welding) of each department. Some of the employees are paid overtime and some of them are paid with daily basis. According to their designation (example: mechanic, welder) are supposed to maintain at least one machine-type of their department. Raw materials are bought from different vendors and fetched to the machining department. Parts from machining department are fetched to fabrication department and so on. Many parts are assembled together to form a product. The final products from assembly department are stored in the ware house. Products are labeled with different specifications (eg, Product_Id, Product_type, MRP, etc).
(8/2074 Bhadra)

6. What are data models? Explain various types of data models. (1+3/2073 Magh)

7. Design an E-R diagram for a database for an airlines system. The database must keep track of customers and their reservations, flights and their stratus, seat assignments on individual flights and the schedule and routing of future flights. Apply all the database design constraints as much as possible. (8/2073 Magh)

8. Differentiate total and partial participation with suitable example and draw an ER diagram for the airport database. Be sure to indicate the various attributes of each entity. Every airplane has a registration number and each airplane is of a specific model. The airport accommodates a number of airplane models and each model is identified by a model number (eg. DC- 10) and has a capacity and a weight. A number of technician works at the airport. You need to store the name, SSN, address, phone number and salary of each technician. Each technician is an expert on one or more plane model(s) and his or her expertise may overlap with that of other technicians. This information about technicians must also be recorded. Traffic controllers must have an annual medical examination. For each traffic controller you must store the data of the most recent exam. (4+8/2073 Bhadra)

9. Draw an ER-diagram for the following mini-case. What is the difference between strong and weak entity sets?
Patients are treated in a single ward by the doctors assigned to them. Healthcare assistants also attend to the patients; a number of these are associated with each ward. Each patient is required to take a variety of drugs a certain number of times per day and for varying lengths of time. The system must record details concerning patient treatment and staff payment. Some staffs are paid part time and doctors and healthcare assistants work varying amounts of overtime at varying rates, the system will also need to track what treatments are required for which patients. (8+4/2072 Ashwin)

10. Draw a complete ER-diagram for the following case.
"A Bus Company owns a number of busses. Each bus is allocated to o particular route, although some routes may have several busses. Each road passes through a number of towns. One or more drivers are allocated to each stage of o route, which corresponds to a journey through some or all of the towns on o route. Some of the towns have a garage where busses are kept and each of the buses ore identified by the registration number and con carry different numbers of passengers, since the vehicles vary in size and can be single or double decked. Each route is identified by a route number and information is available on the average number of passengers carried per day for each route. Drivers hove an employee number, name, address, and sometimes o telephone number."
What is the difference between the degree and cardinality of a relationship?
(8+4/2071 Bhadra)

11. Assume that at Pine Valley Furniture each product (described by Product No., Description, and Cost) is comprised of at least three components {described by Component No.. Description, and Unit of Measure) and components are used to make one or many products {i.e,, must be used in al le.rst one product). In-addition, assume that components are used to make other components and that raw materials are also considered to be components. In both cases of components being used to make other components, we need to keep track of how many components go into making something else. Draw an ER diagram for this case. Describe what is total participation using an ER-diagram example. (8+4/2070 Magh)

12. Draw an ER-diagram for the following mini-case. What is the difference between strong and weak entity sets? (8+4/2070 Bhadra)
Each employee in an engineering company has at most one recognized skill, but a given skill may be possessed by several employees. An employee is able to operate given machine-type (e.g., lathe, grinder) if he has one of several skills. but each skill is associated with the operation of only one machine type. Possession of a given skill (e.g., mechanic, electrician) allows an employee to maintain several machine-types, although maintenance of any given machine-type requires a specific skill (e.g., a lathe must be maintained by a mechanic).

13. Draw an ER-diagram for the following mini-case. What is the difference between cardinality and degree of a relationship?
A university registrar's maintains data about the following entities: (a) Courses, including number, title, credits, syllabus and prerequisites; (b) Course offerings, including course number, year, semester, section number, instructor(s), timings and classroom; (c) Students, including student-id, name, and program; and (d) instructors, including identification number, name, department, and title. Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled. (8+4/2069 Bhadra)

14. Draw an ER-diagram for the following mini-case
Procurement department of the Ministry of Transportation (MOT)keeps track of all the items (furniture and equipment such as a chair or printer) in the Ministry offices. There are several MOT buildings and each one is given a different name to identify it. Each item is assigned a unique ID when it is purchased. This ID is used to keep track of the item, which is assigned to a room within a building. Each room within a building is assigned to a department, and each department has a single employee as it's manager. (8/2068 Bhadra)