**UNIT 1 : INTRODUCTION**

Database systems have become an essential component of life in modern society, in that many frequently occurring events trigger the accessing of at least one database: bibliographic library searches, bank transactions, hotel/airline reservations, grocery store purchases, etc.,

**Traditional vs. more recent applications of databases**:

The applications like bank transactions, hotel/airline reservations, grocery store purchases, etc., are all "traditional" ones for which the use of rigidly-structured textual and numeric data suffices.

Recent advances have led to the application of database technology to a wider class of data. Examples include **multimedia** databases (involving pictures, video clips, and sound messages) and **geographic** databases (involving maps, satellite images).

**Database :**

Database is a collection of related data.

**Properties of Database :**

A database has the following implicit properties:

■ A database represents some aspect of the real world, sometimes called the miniworld or the universe of discourse (UoD). Changes to the miniworld are reflected in the database.

■ A database is a logically coherent collection of data with some inherent meaning.

■ A database is designed, built, and populated with data for a specific purpose.It has an intended group of users and some preconceived applications in which these users are interested.

**Database Management System :**

A **database management system (DBMS)** is a collection of programs that enables users to create and maintain a database.

The DBMS is a *general-purpose software system* that facilitates the processes of ***defining, constructing, manipulating,* and *sharing***databases among various users and applications.

1. **Defining** a database involves specifying the data types, structures, and constraints of the data to be stored in the data base. The database definition or descriptive information is also stored by the DBMS in the form of a database catalog or dictionary; it is called **meta-data**.

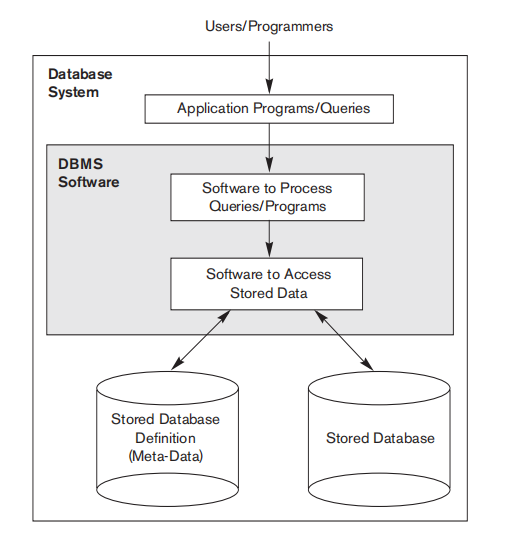
1. **Constructing** the database is the process of storing the data on some storage medium that is controlled by the DBMS.
2. **Manipulating** a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data.
3. **Sharing** a database allows multiple users and programs to access the database simultaneously.

Other important functions provided by the DBMS includes :

1. **Protecting the database : Protection** includes ***system protection***against hardware or software malfunction (or crashes) and ***security protection***against unauthorized or malicious access.
2. M**aintaining it over a long period of time** : The DBMS must be able to **maintain** the database system by allowing the system to evolve as requirements change over time.

**Database System**

The database and DBMS software together a **database system**. The below Figure shows the database system.



**Fig : A Simplified Database Environment**

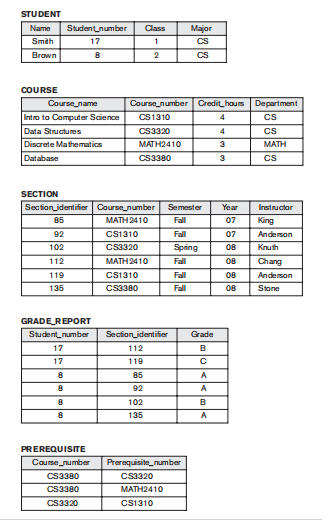
**Example :**

Let us consider a ***UNIVERSITY database*** for maintaining information concerning students, courses, and grades in a university environment.

Below Figure shows the database structure and a few sample data for such a database.

The database is organized as five files, each of which stores **data records** of the same type. The STUDENT file stores data on each student, the COURSE file stores data on each course, the SECTION file stores data

on each section of a course, the GRADE\_REPORT file stores the grades that students receive in the various sections they have completed, and the PREREQUISITE file stores the prerequisites of each course.



**Fig (b) : A database that stores student and course information.**

**Requirements Specification and Analysis:**

* Design of a new application for an existing database or design of a brand new database starts off with a phase called **requirements specification and analysis**.

1. **Conceptual design:** These requirements (which include design interactions, experience and strategies) are documented in detail.
2. **Logical design:** The design is then translated to a **logical design** that can be expressed in a data model implemented in a commercial DBMS. (Logical relationship, constraints etc,).
3. The final stage is **Physical design**: during which further specifications are provided for storing and accessing the database. The database design is implemented, populated with actual data, and continuously maintained to reflect the state of the mini world.

**Characteristics of the Database Approach:**

The main characteristics of the database approach versus the file-processing approach are :

* A fundamental characteristic of the database approach is that the database system contains not only the database itself but also a complete definition or description of the database structure and constraints.
* This definition is stored in the *DBMS catalog, which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data.*
* *The information stored in the catalog is called meta-data, and it describes the structure of the primary database (Figure a).*
* The catalog is used by the DBMS software and also by database users who need information about the database structure. A general-purpose DBMS software package is not written for a specific database application.
* Therefore, it must refer to the catalog to know the structure of the files in a specific database, such as the type and format of data it will access.
* The DBMS software must work equally well with any number of database applications—for example, a university database, a banking database, or a company database—as long as the database definition is stored in the catalog.

**DATABASE USERS:**

1. **Database Administrators :**

* In a database environment, the primary resource is the database itself, and the secondary resource is the DBMS and related software.
* Administering these resources is the responsibility of the **database administrator (DBA)**.
* The DBA is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed.
* The DBA is accountable for problems such as security breaches and poor system response time.
* In large organizations, the DBA is assisted by a staff that carries out these functions.

1. **Database Designers :**

* They are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data.
* It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements and to create a design that meets these requirements.
* The final database design must be capable of supporting the requirements of all user groups.

1. **End Users :**

* **End users** are the people whose jobs require access to the database for querying,

updating, and generating reports.

The several categories of end users:

■ **Casual end users** occasionally access the database, but they may need different information each time. They use a sophisticated database query language to specify their requests and are typically middle- or high-level managers or other occasional browsers.

■ **Naive** or **parametric end users** Their main job function revolves around constantly querying and updating the database, called **canned transactions—**that have been carefully programmed and tested.

**■ Standalone users** maintain personal databases by using ready-made program packages that provide easy-to-use menu-based or graphics-based interfaces.

1. **System Analysts and Application Programmers (Software Engineers) :**

* **System analysts** determine the requirements of end users, especially naive and parametric end users, and develop specifications for standard canned transactions

that meet these requirements.

* **Application programmers** implement these specifications as programs; then they test, debug, document, and maintain these canned transactions. Such analysts and programmers—commonly referred to as **software developers** or **software engineers.**

**Workers behind the Scene :**

* **DBMS system designers/implementers:** provide the DBMS software that is at the foundation of all this!
* **Tool developers:** design and implement software tools facilitating database system design, performance monitoring, creation of graphical user interfaces, prototyping, etc.
* **Operators and maintenance personnel**: responsible for the day-to-day operation of the system.

**Advantages of Using the DBMS Approach**

1. Controlling Redundancy
2. Restricting Unauthorized Access
3. Providing Persistent Storage for Program Objects
4. Providing Storage Structures for Efficient Query Processing
5. Providing Backup and Recovery.
6. Providing Multiple User Interfaces
7. Representing Complex Relationships Among Data
8. Enforcing Integrity Constraints
9. Permitting Inference and Actions Via Rules
10. Potential for enforcing standards
11. Reduced application development time
12. Flexibility to change data structures
13. Availability of up-to-date information
14. Economies of scale.

**Database System Concepts and Architecture**

One fundamental characteristic of the database approach is that it provides some level of ***data abstraction***.

***Data abstraction*** generally refers to the suppression of details of data organization and storage, and the highlighting of the essential features for an improved understanding of data.

***Data model***—a collection of concepts that can be used to describe the structure of a database.

**Categories of Data Models:**

Data models have been categorize according to the types of concepts they use to describe the database structure.

1. ***High-level*** *or* ***conceptual data models*** provide concepts that are close to the way many users perceive data

* Conceptual data models use concepts such as entities, attributes, and relationships.

***Entity* : R**epresents a real-world object or concept.

Ex : Project, Car, Employee form the miniworld that is described in the database.

***Attribute:* P**roperty that describes an entity, such as the employee’s name or salary.

***Relationship:*** Association betweentwo or more entities.

Ex : a works-on relationship between an employee and a project.

1. ***low-level*** *or* ***physical data models*** provide concepts that describe the details of how data is stored on the computer storage media.
2. ***Representational*** *(or* ***implementation****)* ***data models*** which provide concepts that may be easily understood by end users. Representational data models hide many details of data storage on disk but can be implemented on a computer system directly.

**Data Models:**

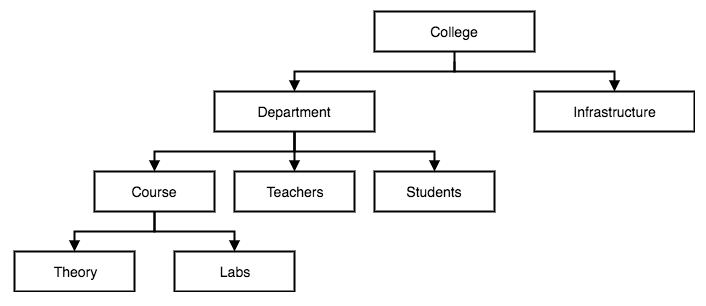
A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system.

**Types of Data Models:**

* + - 1. **Hierarchical Model**
      2. **Network Model**
      3. **Relational Model**
      4. **Object oriented Model**
      5. **E-R Model.**

**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 heirarchy 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.

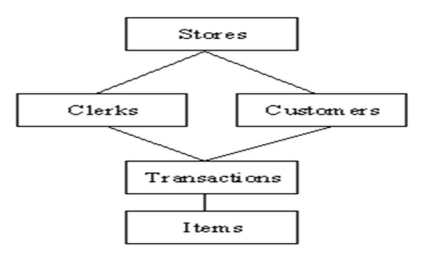


**Network Model:**

This is an extension of the Hierarchical model. In this model data is organised 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.



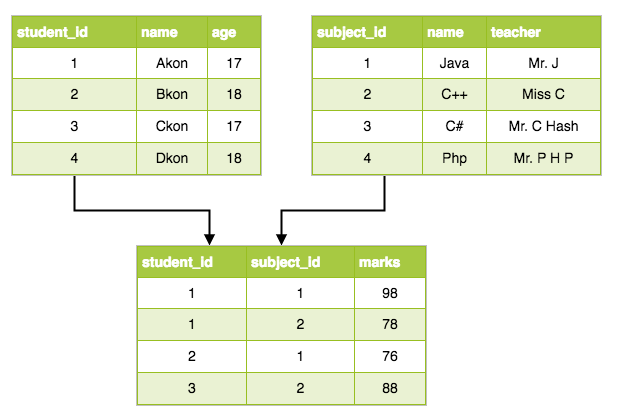
**Relational Model :**

In this model, data is organised 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, infact, 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.



**Object Oriented Data Model:**

In the object oriented data model (OODM), both data and their relationships are contained in a single structure known as an object.

In turn, the OODM is the basis for the object-oriented database management system (OODBMS).

**The Components of the Object Oriented Data Model**

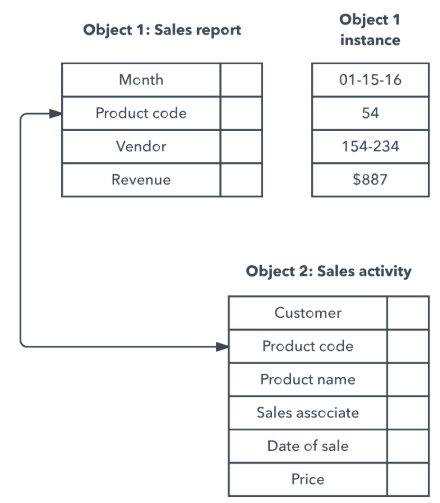
• An object is an abstraction of a real-world entity.

• Attributes describe the properties of an object.

• Objects that share similar characteristics are grouped in classes. A class is a collection of similar objects with shared structure (attributes) and behavior (methods).

• Classes are organized in a class hierarchy. The class hierarchy resembles an upside-down tree in which each class has only one parent.

• Inheritance is the ability of an object within the class hierarchy to inherit the attributes and methods of the classes above it.



**Schemas, Instances,**

**Database Schema** : The description of a database is called the **database schema**, which is specified during database design and is not expected to change frequently.

**Schema Diagram** A displayed schema is called a schema diaram. Figure 2a shows a schema diagram for the database shown in Figure (b) in chap 1, the diagram displays the

**Schema Construct**: structure of each record type but not the actual instances of records.

Ex : each object in the schema—such as STUDENT or COURSE

**Snapshot / Database State:** The data in the database at a particular moment in time is called a **database state** or **snapshot**. (or) It is also called the *current* set of **occurrences** or **instances** in the Database.

The **schema** is sometimes called the **intension**.

**Database state** is called an **extension** of the schema.

**Schema Evolution**: It is the ability of a database system to respond to changes in the real world by allowing the schema to evolve.

For example, we may decide that another data item needs to be stored for each record in a file, such as adding the Date\_of\_birth to the STUDENT schema in Figure 2a. This is known as **schema evolution**.

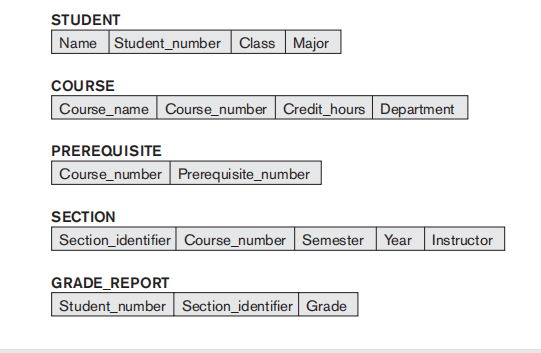


Fig 2a: Schema diagram for the database in fig(a) in chapter 1.

**Three-Schema Architecture and Data Independence :**

**The Three-Schema Architecture :**

The goal of the three-schema architecture, illustrated in Figure 2b, is to separate the user applications from the physical database.

In this architecture, schemas can be defined at the following three levels:

1. ***The internal level*** has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

2. ***The conceptual level*** has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.

3. ***The external or view level*** includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.

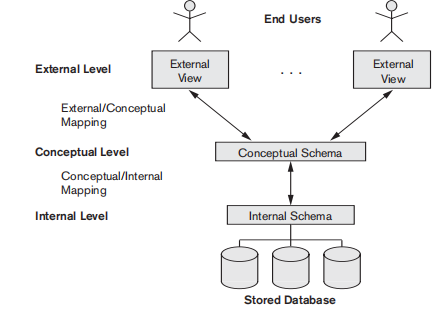


Fig 2b : Three Schema Architecture.

* The three schemas are only descriptions of data
* The stored data that actually exists is at the physical level only.
* In a DBMS based on the three-schema architecture, each user group refers to its own external schema.
* Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database.
* If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user’s external view.
* The processes of transforming requests and results between levels are called **mappings.**

**Data Independence**

***Data Independence*** is defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

We can define two types of data independence:

1. **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item).
2. **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well. Changes to the internal schema may be needed because some physical files were reorganized.

**Database Languages and Interfaces**

***Database languages:***

1. **Data Definition Language** (**DDL**): is used by the DBA and by database designers to define both schemas.
2. **Storage Definition Language** (**SDL**): is used to specify the internal schema. The mappings between the two schemas may be specified in either one of these languages.
3. **View Definition Language** (**VDL**): To specify user views and their mappings to the conceptual schema, but in most DBMSs *the DDL is used to define both conceptual and external schemas*. In relational DBMSs, SQL is used in the role of VDL to define user or application **views** as results of predefined queries
4. **Data Manipulation Language** (**DML**): Once the database schemas are compiled and the database is populated with data, users must have some means to manipulate the database. Typical manipulations include retrieval, insertion, deletion, and modification of the data. The DBMS provides a set of operations or a language called the (**DML**) for these purposes.

**Entity Types, Entity Sets, Attributes, and Keys :**

**Entities and Attributes :**

***Entity :*** is a *thing* in the real world with an independent existence. An entity may be an object with a physical existence (for example, a particular person, car, house, or employee) or it may be an object with a conceptual existence (for instance, a company, a job, or a university course).

***Attribute :*** The properties that describe entity.

For example, an EMPLOYEE entity may be described by the employee’s name, age, address, salary, and job.

**Types of Attributes** :

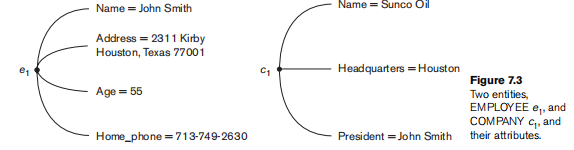
Several types of attributes occur in the ER model:

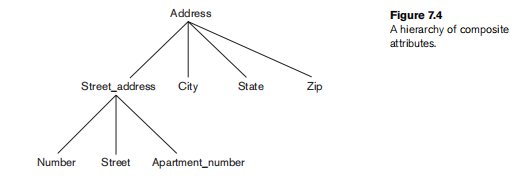
1. **Simple versus Composite,**
2. **SingleValued versus Multivalued, &**
3. **Stored versus Derived.**
4. **Null Value for an Attribute.**
5. **Composite versus Simple (Atomic) Attributes :**

**Composite attributes :** can be divided into smaller subparts.

For example, the Address attribute of the EMPLOYEE entity shown in below Figure can be subdivided into Street\_address, City, State, and Zip,3 with the values ‘2311 Kirby’, ‘Houston’, ‘Texas’, and ‘77001.’

**Simple** Or **Atomic Attributes:** The attributes that are not divisible are called Simple Or Atomic Attributes.





1. **Single-Valued versus Multivalued Attributes**.

**Single Valued Attributes**: The attributes have a single value for a particular entity; such attributes are called **single-valued**.

For example, Age is a single-valued attribute of a person.

**Multivalued Attributes:** A multivalued attribute may have a multiple value for a particular entity; such attributes are called **multi-valued**.

For example, the degree of a person.

1. **Stored versus Derived Attributes**.

In some cases, two (or more) attribute values are related—for example, the Age and Birth\_date attributes of a person.

For a particular person entity, the value of Age can be determined from the current (today’s) date and the value of that person’s Birth\_date.

The Age attribute is hence called a **derived attribute (attribute derived from other attribute)** and is said to be **derivable from** the Birth\_date attribute, which is called a **stored attribute**.

1. **NULL Values**.

It is a value to be considered when a particular entity may not have an applicable value for an attribute.

For example, the Apartment\_number attribute of an address applies only to addresses that are in apartment buildings and not to other types of residences, such as single-family homes.

Similarly, a College\_degrees attribute appliesonly to people with college degrees. For such situations, a special value called NULL is created.

The unknown category of NULL can be further classified into two cases.

The first case arises when it is known that the attribute value exists but is missing—for instance, if the Height attribute of a person is listed as NULL.

The second case arises when it is not known whether the attribute value exists—for example, if the Home\_phone attribute of a person is NULL.

**Cardinality Ratios for Binary Relationships :** The cardinality ratio for a binary relationship specifies the maximum number of relationship instances that an entity can participate in.

The possible cardinality ratios for binary relationship types are **1:1, 1:N, N:1, and M:N.**

1. **1:1 (One to One) cardinality ratio** : If an entity [a record] of one entity set is associated with maximum of one entity of the other entity set, then the relationship type is said to be

one-to-one.

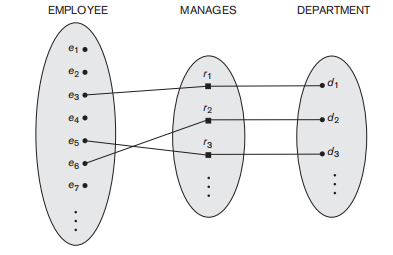
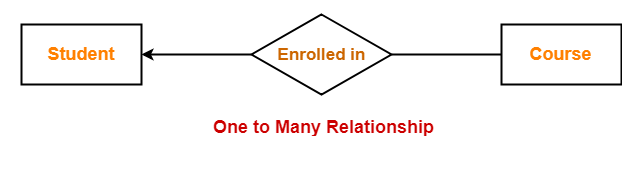
.

Fig : 1:1 Relationship,Manages

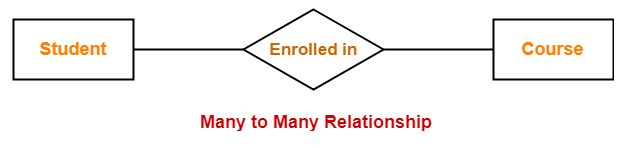
1. **1:N (One to Many) :** If an entity of one entity set is associated with zero or more entities of the other entity set, then the cardinality ratio is said to be one-to-many from one side entity set to the many side entity set.

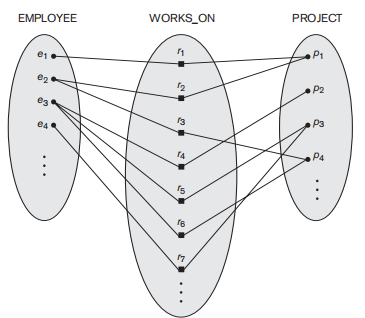


1. **N:1 (Many to One) :** An entity in set A can be associated with at most one entity in set B and an entity in set B can be associated with any number (zero or more) of entities in set A.



1. **M:N (Many to Many)** : An entity in set A can be associated with any number (zero or more) of entities in set B and an entity in set B can be associated with any number (zero or more) of entities in set A.





**Weak Entity Types**

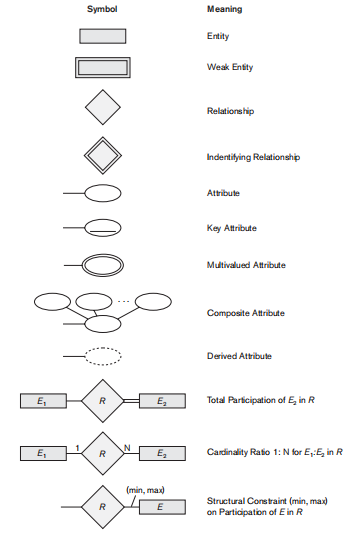
**Weak entity types** : Entity types that do not have key attributes of their own.

**Regular entity types :** Entity typesthat do have a key attribute.

A weak entity type always has a *total participation constraint* (existence dependency) with respect to its identifying relationship because a weak entity can not be identified without an owner entity.

**Partial Key:** A weak entity type normally has a partial key, which is the attribute that can uniquely identify weak entities that are *related to the same owner entity*.

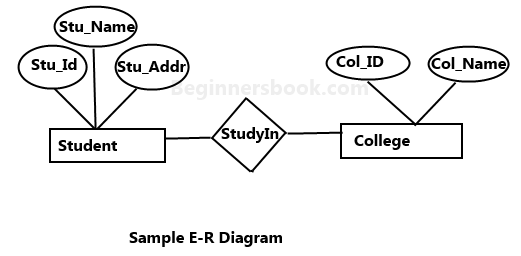
**Notations for ER Diagram :**



**ER (Entity Relationship) Diagram**

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.

**Example : A simple ER Diagram**



In the above 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.

**Notations and and their meaning in an E-R Diagram:**

**Rectangle:** Represents Entity sets.

**Ellipses**: Attributes

**Diamonds**: Relationship Set

**Lines:** They link attributes to Entity Sets and Entity sets to Relationship Set

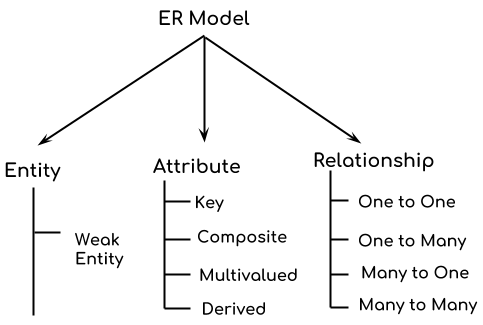
**Double Ellipses:** Multivalued Attributes

**Dashed Ellipses**: Derived Attributes

**Double Rectangles**: Weak Entity Sets

**Double Lines**: Total participation of an entity in a relationship set.

**Components of ER Diagram**



**Example : ER diagram for Company Database**

