**Unit – 4 Systems Design and modelling**

**System Design-**

System design is the process of designing the elements of a system such as the architecture, modules, and components, the different interfaces of those components, and the data that goes through that system. System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. It involves translating user requirements into a detailed blueprint that guides the implementation phase. The goal is to create a well-organized and efficient structure that meets the intended purpose while considering factors like scalability, maintainability, and performance.

## Objectives of System Design

1. **Practicality**: We need a system that should be targetting the set of audiences(users) corresponding to which they are designing.
2. **Accuracy**: Above system design should be designed in such a way it fulfills nearly all requirements around which it is designed be it functional o non-functional requirements.
3. **Completeness**: System design should meet all user requirements
4. **Efficient**: The system design should be such that it should not overuse surpassing the cost of resources nor under use as it will by now we know will result in low thorough put (output) and less response time(latency).
5. **Reliability**: The system designed should be in proximity to a failure-free environment for a certain period of time.
6. **Optimization**: Time and space are just likely what we do for code chunks for individual components to work in a system.
7. **Scalable(flexibility)**: System design should be adaptable with time as per different user needs of customers which we know will keep on changing on time.

## [**Components of System Design**](https://www.geeksforgeeks.org/what-are-the-components-of-system-design/)

Below are some of the major components of the System Design. discussed in brief. The detailed version of this will be discussed in different posts:

1. **Load balancers:**Most crucial component for scalability, availability, and performance measures for systems.
2. **Key Value Stores:** It is a storage system similar to hashtables where key-value stores are distributed hash tables.
3. **Blob Storage:** Blob stands for binary large objects, as the name suggests is storage for unstructured data such as YouTube, and Netflix.
4. **Databases:** It is an organized collection of data so that they can be easily accessed and modified.
5. **Rate Limiters:** These sets the maximum number of requests a service can fulfill.
6. **Monitoring System:** These are basically software where system administrator monitor infrastructures such as bandwidth, CPU, routers, switches, etc.
7. **Distributed System Messaging Queue:** Transaction medium between producers and consumers.
8. **Distributed Unique ID generator:** In the case of large distributed systems, every moment multiple tasks are occurring so in order to distinguish it assign a tag corresponding to every event.
9. **Distributed Search:** Over every website, crucial information that visitors will seek is put into the search bar.
10. **Distributed Logging Services:** Tracing sequences of events from end to end.
11. **Distributed Task Scheduler:**  Computational resources such as CPU, memory, storage, etc.

**Computer Aided Software Engineering (CASE)-**

**Computer-aided software engineering (CASE)** is the implementation of computer-facilitated tools and methods in software development. CASE is used to ensure high-quality and defect-free software. CASE ensures a check-pointed and disciplined approach and helps designers, developers, testers, managers, and others to see the project milestones during development.   
  
CASE can also help as a warehouse for documents related to projects, like business plans, requirements, and design specifications. One of the major advantages of using CASE is the delivery of the final product, which is more likely to meet real-world requirements as it ensures that customers remain part of the process.   
  
CASE illustrates a wide set of labor-saving tools that are used in software development. It generates a framework for organizing projects and to be helpful in enhancing productivity. There was more interest in the concept of CASE tools years ago, but less so today, as the tools have morphed into different functions, often in reaction to software developer needs. The concept of CASE also received a heavy dose of criticism after its release.

## **What is CASE Tools?**

The essential idea of CASE tools is that in-built programs can help to analyze developing systems in order to enhance quality and provide better outcomes. Throughout the 1990, CASE tool became part of the software lexicon, and big companies like IBM were using these kinds of tools to help create software.

## **Types of CASE Tools:**

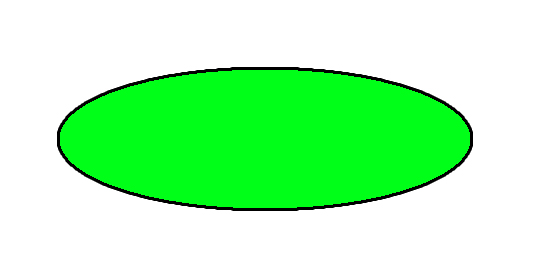
1. **Diagramming Tools:** It helps in diagrammatic and graphical representations of the data and system processes. It represents system elements, control flow and data flow among different software components and system structures in a pictorial form. For example, Flow Chart Maker tool for making state-of-the-art flowcharts.
2. **Computer Display and Report Generators:** These help in understanding the data requirements and the relationships involved.
3. **Analysis Tools:** It focuses on inconsistent, incorrect specifications involved in the diagram and data flow. It helps in collecting requirements, automatically check for any irregularity, imprecision in the diagrams, data redundancies, or erroneous omissions.   
   For example:  
   * (i) Accept 360, Accompa, CaseComplete for requirement analysis.
   * (ii) Visible Analyst for total analysis.
4. **Central Repository:** It provides a single point of storage for data diagrams, reports, and documents related to project management.
5. **Documentation Generators:** It helps in generating user and technical documentation as per standards. It creates documents for technical users and end users.   
   For example, Doxygen, DrExplain, Adobe RoboHelp for documentation.
6. **Code Generators:** It aids in the auto-generation of code, including definitions, with the help of designs, documents, and diagrams.
7. **Tools for Requirement Management:** It makes gathering, evaluating, and managing software needs easier.
8. **Tools for Analysis and Design**: It offers instruments for modelling system architecture and behaviour, which helps throughout the analysis and design stages of software development.
9. **Tools for Database Management:** It facilitates database construction, design, and administration.
10. **Tools for Documentation:**It makes the process of creating, organizing, and maintaining project documentation easier.

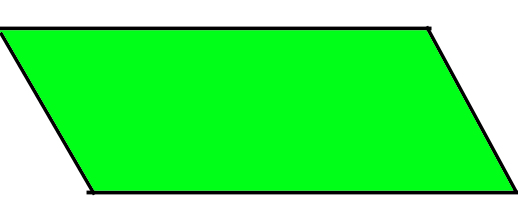
**Systems Flowcharts –**

**What is a Flowchart?**   
Flowchart is a graphical representation of an algorithm. Programmers often use it as a program-planning tool to solve a problem. It makes use of symbols which are connected among them to indicate the flow of information and processing.   
The process of drawing a flowchart for an algorithm is known as “flowcharting”.

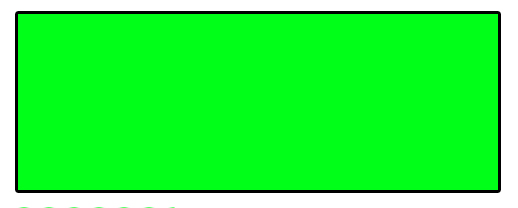
**Basic Symbols used in Flowchart Designs**

1. **Terminal:** The oval symbol indicates Start, Stop and Halt in a program’s logic flow. A pause/halt is generally used in a program logic under some error conditions. Terminal is the first and last symbols in the flowchart.

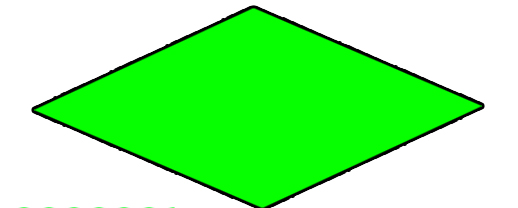
*   
  **Input/Output:** A parallelogram denotes any function of input/output type. Program instructions that take input from input devices and display output on output devices are indicated with parallelogram in a flowchart.



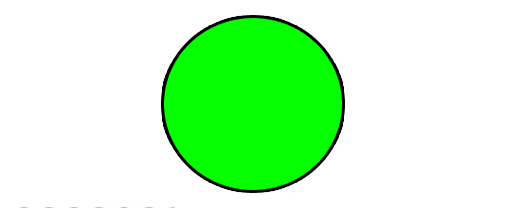
* **Processing:** A box represents arithmetic instructions. All arithmetic processes such as adding, subtracting, multiplication and division are indicated by action or process symbol.



* **Decision** Diamond symbol represents a decision point. Decision based operations such as yes/no question or true/false are indicated by diamond in flowchart.



* **Connectors:** Whenever flowchart becomes complex or it spreads over more than one page, it is useful to use connectors to avoid any confusions. It is represented by a circle.



* **Flow lines:** Flow lines indicate the exact sequence in which instructions are executed. Arrows represent the direction of flow of control and relationship among different symbols of flowchart.

**Rules For Creating Flowchart :**

A flowchart is a graphical representation of an algorithm.it should follow some rules while creating a flowchart  
Rule 1: Flowchart opening statement must be ‘start’ keyword.  
Rule 2:  Flowchart ending statement must be ‘end’ keyword.  
Rule 3: All symbols in the flowchart must be connected with an arrow line.  
Rule 4: The decision symbol in the flowchart is associated with the arrow line.

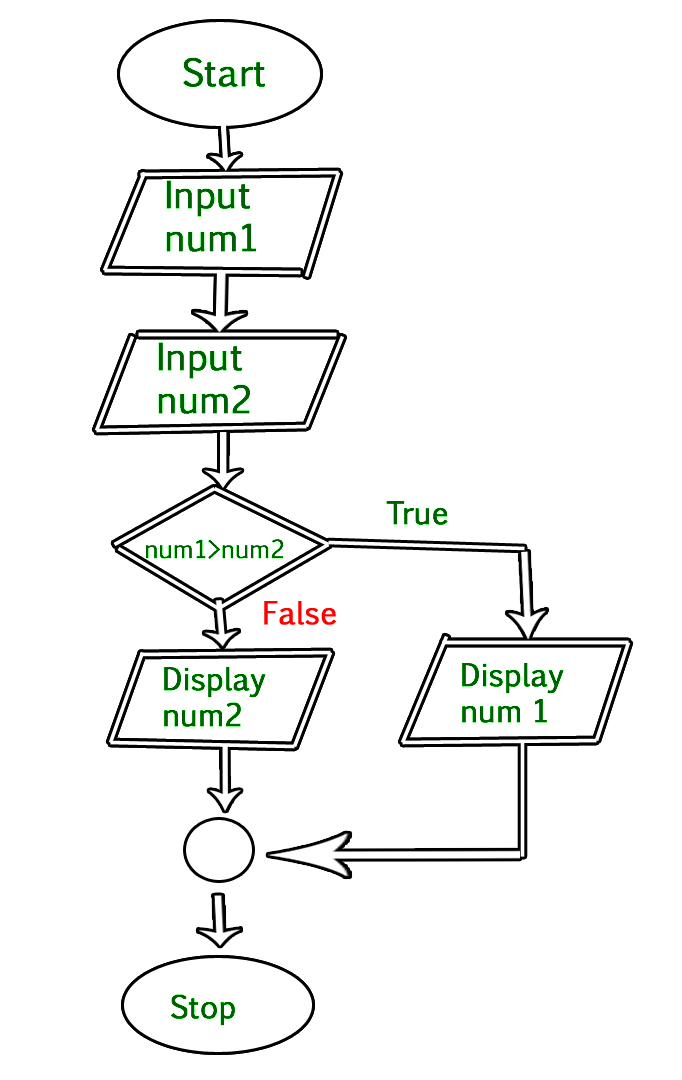
**Advantages of Flowchart:**

* Flowcharts are a better way of communicating the logic of the system.
* Flowcharts act as a guide for blueprint during program designed.
* Flowcharts help in debugging process.
* With the help of flowcharts programs can be easily analyzed.
* It provides better documentation.
* Flowcharts serve as a good proper documentation.
* Easy to trace errors in the software.
* Easy to understand.
* The flowchart can be reused for inconvenience in the future.
* It helps to provide correct logic.

**Disadvantages of Flowchart:**

* It is difficult to draw flowcharts for large and complex programs.
* There is no standard to determine the amount of detail.
* Difficult to reproduce the flowcharts.
* It is very difficult to modify the Flowchart.
* Making a flowchart is costly.
* Some developer thinks that it is waste of time.
* It makes software processes low.
* If changes are done in software, then the flowchart must be redrawn

**Example :** **Draw a flowchart to input two numbers from the user and display the largest of two numbers** 



**Data Flow Diagrams –**

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

It shows how data enters and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

**The following observations about DFDs are essential:**

1. All names should be unique. This makes it easier to refer to elements in the DFD.
2. Remember that DFD is not a flow chart. Arrows is a flow chart that represents the order of events; arrows in DFD represents flowing data. A DFD does not involve any order of events.
3. Suppress logical decisions. If we ever have the urge to draw a diamond-shaped box in a DFD, suppress that urge! A diamond-shaped box is used in flow charts to represents decision points with multiple exists paths of which the only one is taken. This implies an ordering of events, which makes no sense in a DFD.
4. Do not become bogged down with details. Defer error conditions and error handling until the end of the analysis.

Standard symbols for DFDs are derived from the electric circuit diagram analysis and are shown in fig:



**Circle:** A circle (bubble) shows a process that transforms data inputs into data outputs.

**Data Flow:** A curved line shows the flow of data into or out of a process or data store.

**Data Store:** A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.

**Source or Sink:** Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.

## **Levels in Data Flow Diagrams (DFD)**

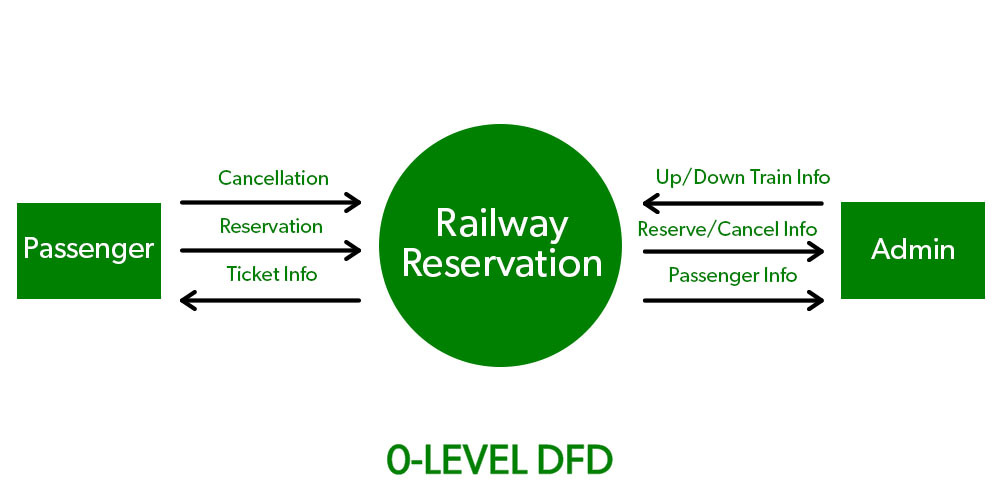
In Software engineering DFD(data flow diagram) can be drawn to represent the system of different levels of abstraction. Higher-level DFDs are partitioned into low levels-hacking more information and functional elements. Levels in DFD are numbered 0, 1, 2 or beyond. Here, we will see mainly 3 levels in the data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.

Data Flow Diagrams (DFD) are graphical representations of a system that illustrate the flow of data within the system. DFDs can be divided into different levels, which provide varying degrees of detail about the system. The following are the four levels of DFDs:

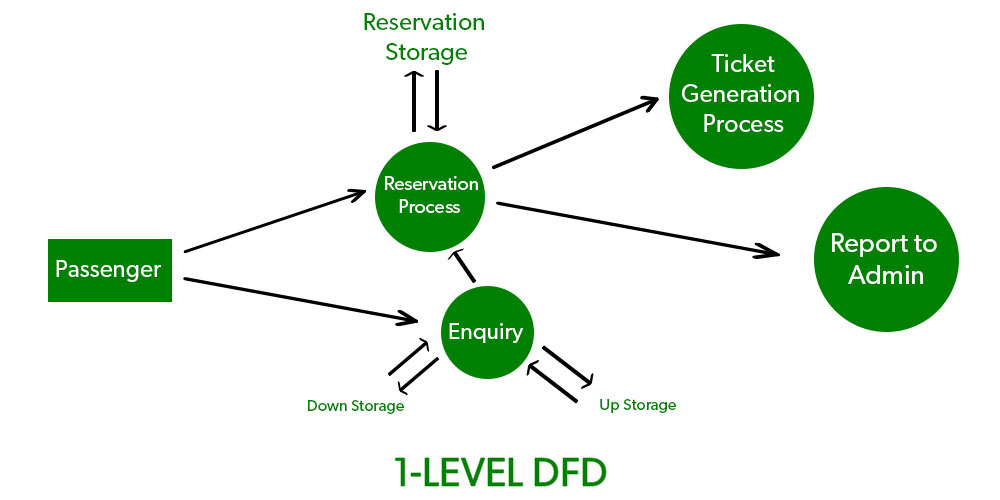
1. **Level 0 DFD:**This is the highest-level DFD, which provides an overview of the entire system. It shows the major processes, data flows, and data stores in the system, without providing any details about the internal workings of these processes.
2. **Level 1 DFD:** This level provides a more detailed view of the system by breaking down the major processes identified in the level 0 DFD into sub-processes. Each sub-process is depicted as a separate process on the level 1 DFD. The data flows and data stores associated with each sub-process are also shown.
3. **Level 2 DFD:**This level provides an even more detailed view of the system by breaking down the sub-processes identified in the level 1 DFD into further sub-processes. Each sub-process is depicted as a separate process on the level 2 DFD. The data flows and data stores associated with each sub-process are also shown.
4. **Level 3 DFD:**This is the most detailed level of DFDs, which provides a detailed view of the processes, data flows, and data stores in the system. This level is typically used for complex systems, where a high level of detail is required to understand the system. Each process on the level 3 DFD is depicted with a detailed description of its input, processing, and output. The data flows and data stores associated with each process are also shown.

The choice of DFD level depends on the complexity of the system and the level of detail required to understand the system. Higher levels of DFD provide a broad overview of the system, while lower levels provide more detail about the system’s processes, data flows, and data stores. A combination of different levels of DFD can provide a complete understanding of the system.

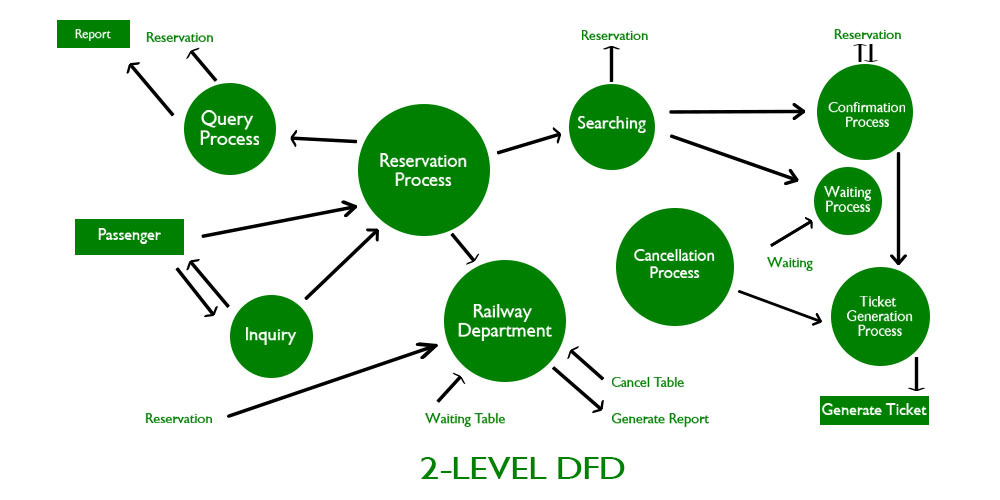
* **0-level DFD:** It is also known as a context diagram. It’s designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.



* **1-level DFD:** In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into subprocesses.



* **2-level DFD:** 2-level DFD goes one step deeper into parts of 1-level DFD. It can be used to plan or record the specific/necessary detail about the system’s functioning.



### **Advantages of using Data Flow Diagrams (DFD) include:**

1. Easy to understand: DFDs are graphical representations that are easy to understand and communicate, making them useful for non-technical stakeholders and team members.
2. Improves system analysis: DFDs are useful for analyzing a system’s processes and data flow, which can help identify inefficiencies, redundancies, and other problems that may exist in the system.
3. Supports system design: DFDs can be used to design a system’s architecture and structure, which can help ensure that the system is designed to meet the requirements of the stakeholders.
4. Enables testing and verification: DFDs can be used to identify the inputs and outputs of a system, which can help in the testing and verification of the system’s functionality.
5. Facilitates documentation: DFDs provide a visual representation of a system, making it easier to document and maintain the system over time.

### **Disadvantages of using DFDs include:**

1. Can be time-consuming: Creating DFDs can be a time-consuming process, especially for complex systems.
2. Limited focus: DFDs focus primarily on the flow of data in a system, and may not capture other important aspects of the system, such as user interface design, system security, or system performance.
3. Can be difficult to keep up-to-date: DFDs may become out-of-date over time as the system evolves and changes.
4. Requires technical expertise: While DFDs are easy to understand, creating them requires a certain level of technical expertise and familiarity with the system being analyzed.
5. Overall, the benefits of using DFDs outweigh the disadvantages. However, it is important to recognize the limitations of DFDs and use them in conjunction with other tools and techniques to analyze and design complex software systems.

**Entity Relationship Diagram –**

ER-modeling is a data modeling method used in software engineering to produce a conceptual data model of an information system. Diagrams created using this ER-modeling method are called Entity-Relationship Diagrams or ER diagrams or ERDs.

## **Purpose of ERD**

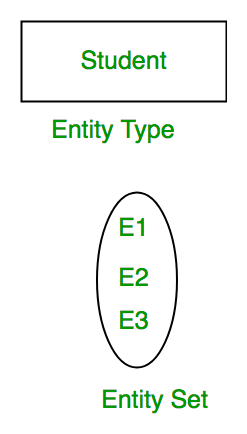
* The database analyst gains a better understanding of the data to be contained in the database through the step of constructing the ERD.
* The ERD serves as a documentation tool.
* Finally, the ERD is used to connect the logical structure of the database to users. In particular, the ERD effectively communicates the logic of the database to users.

## **Components of an ER Diagrams**

### **1. Entity**

An entity can be a real-world object, either animate or inanimate, that can be merely identifiable. An entity is denoted as a rectangle in an ER diagram. For example, in a school database, students, teachers, classes, and courses offered can be treated as entities. All these entities have some attributes or properties that give them their identity.

**Entity Set:** An Entity is an object of Entity Type and a set of all entities is called an entity set. For Example, E1 is an entity having Entity Type Student and the set of all students is called Entity Set. In ER diagram, Entity Type is represented as:



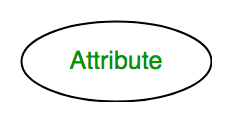
An entity set is a collection of related types of entities. An entity set may include entities with attribute sharing similar values. For example, a Student set may contain all the students of a school; likewise, a Teacher set may include all the teachers of a school from all faculties. Entity set need not be disjoint.



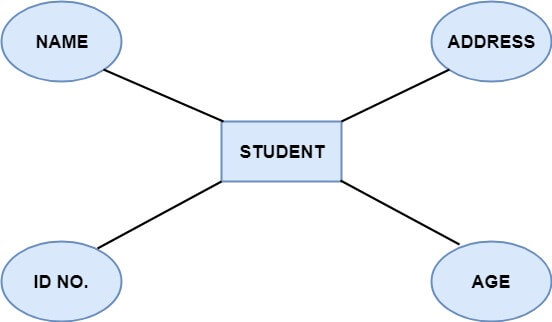
### **Attributes**

Entities are denoted utilizing their properties, known as attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes.

[Attributes](https://www.geeksforgeeks.org/types-of-attributes-in-er-model/) are the properties that define the entity type. For example, Roll\_No, Name, DOB, Age, Address, and Mobile\_No are the attributes that define entity type Student. In ER diagram, the attribute is represented by an oval.



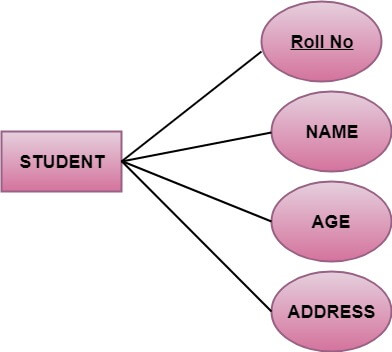
There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.



**There are four types of Attributes:**

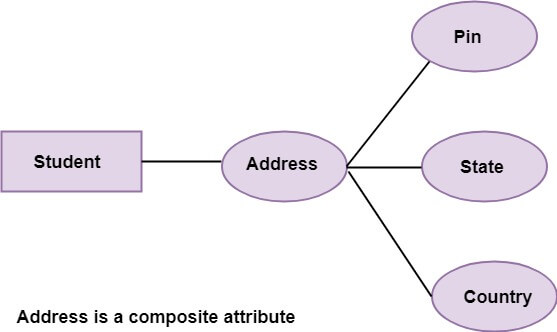
1. Key attribute
2. Composite attribute
3. Single-valued attribute
4. Multi-valued attribute
5. Derived attribute

**1. Key attribute:** Key is an attribute or collection of attributes that uniquely identifies an entity among the entity set. For example, the roll\_number of a student makes him identifiable among students.



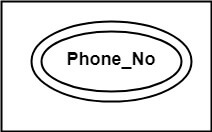
**There are mainly three types of keys:**

1. **Super key:** A set of attributes that collectively identifies an entity in the entity set.
2. **Candidate key:** A minimal super key is known as a candidate key. An entity set may have more than one candidate key.
3. **Primary key:** A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.
4. **Composite attribute:** An attribute that is a combination of other attributes is called a composite attribute. For example, In student entity, the student address is a composite attribute as an address is composed of other characteristics such as pin code, state, country.

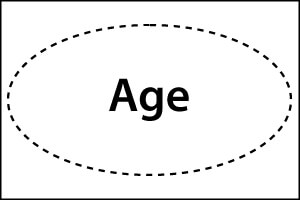


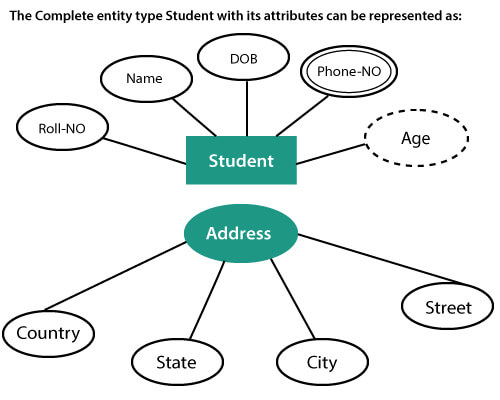
**3. Single-valued attribute:** Single-valued attribute contain a single value. For example, Social\_Security\_Number.

**4. Multi-valued Attribute:** If an attribute can have more than one value, it is known as a multi-valued attribute. Multi-valued attributes are depicted by the double ellipse. For example, a person can have more than one phone number, email-address, etc.



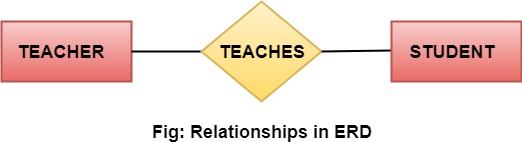
**5. Derived attribute:** Derived attributes are the attribute that does not exist in the physical database, but their values are derived from other attributes present in the database. For example, age can be derived from date\_of\_birth. In the ER diagram, Derived attributes are depicted by the dashed ellipse.





## **3. Relationships**

The association among entities is known as relationship. Relationships are represented by the diamond-shaped box. For example, an employee works\_at a department, a student enrolls in a course. Here, Works\_at and Enrolls are called relationships.



## **Relationship set**

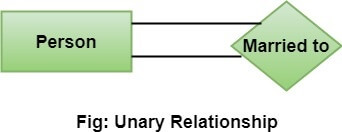
A set of relationships of a similar type is known as a relationship set. Like entities, a relationship too can have attributes. These attributes are called descriptive attributes.

## **Degree of a relationship set**

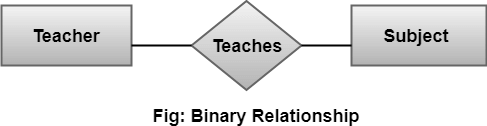
The number of participating entities in a relationship describes the degree of the relationship. The three most common relationships in E-R models are:

1. Unary (degree1)
2. Binary (degree2)
3. Ternary (degree3)

**1. Unary relationship:** This is also called recursive relationships. It is a relationship between the instances of one entity type. For example, one person is married to only one person.

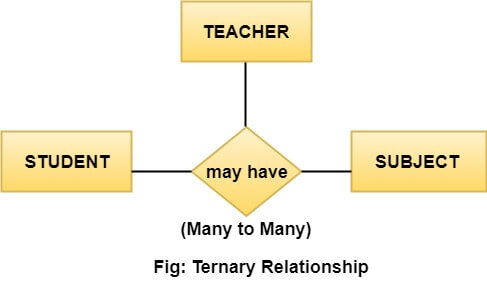


**2. Binary relationship:** It is a relationship between the instances of two entity types. For example, the Teacher teaches the subject.



**3. Ternary relationship:** It is a relationship amongst instances of three entity types. In fig, the relationships "**may have**" provide the association of three entities, i.e., TEACHER, STUDENT, and SUBJECT. All three entities are many-to-many participants. There may be one or many participants in a ternary relationship.

In general, "**n**" entities can be related by the same relationship and is known as **n-ary** relationship.



**• Tools for Structured Analysis :**

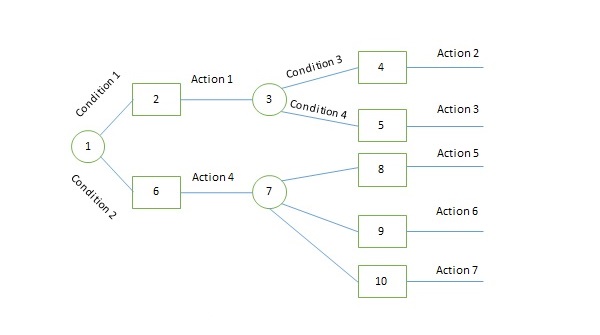
Structured Analysis is a development method that allows the analyst to understand the system and its activities in a logical way.

It is a systematic approach, which uses graphical tools that analyze and refine the objectives of an existing system and develop a new system specification which can be easily understandable by user.

**• Decision Tree**

Decision trees are a method for defining complex relationships by describing decisions and avoiding the problems in communication. A decision tree is a diagram that shows alternative actions and conditions within horizontal tree framework. Thus, it depicts which conditions to consider first, second, and so on.

Decision trees depict the relationship of each condition and their permissible actions. A square node indicates an action and a circle indicates a condition. It forces analysts to consider the sequence of decisions and identifies the actual decision that must be made.

  
The major limitation of a decision tree is that it lacks information in its format to describe what other combinations of conditions you can take for testing. It is a single representation of the relationships between conditions and actions.

## **Decision Tables**

Decision tables are a method of describing the complex logical relationship in a precise manner which is easily understandable.

* It is useful in situations where the resulting actions depend on the occurrence of one or several combinations of independent conditions.
* It is a matrix containing row or columns for defining a problem and the actions.

### **Components of a Decision Table**

* **Condition Stub** − It is in the upper left quadrant which lists all the condition to be checked.
* **Action Stub** − It is in the lower left quadrant which outlines all the action to be carried out to meet such condition.
* **Condition Entry** − It is in upper right quadrant which provides answers to questions asked in condition stub quadrant.
* **Action Entry** − It is in lower right quadrant which indicates the appropriate action resulting from the answers to the conditions in the condition entry quadrant.

The entries in decision table are given by Decision Rules which define the relationships between combinations of conditions and courses of action. In rules section,

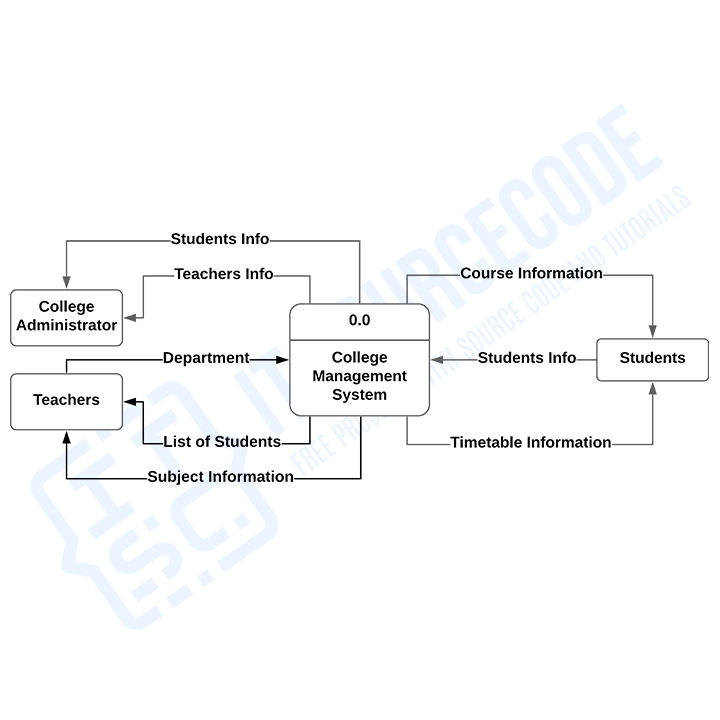
* Y shows the existence of a condition.
* N represents the condition, which is not satisfied.
* A blank - against action states it is to be ignored.
* X (or a check mark will do) against action states it is to be carried out.

## **Structured English**

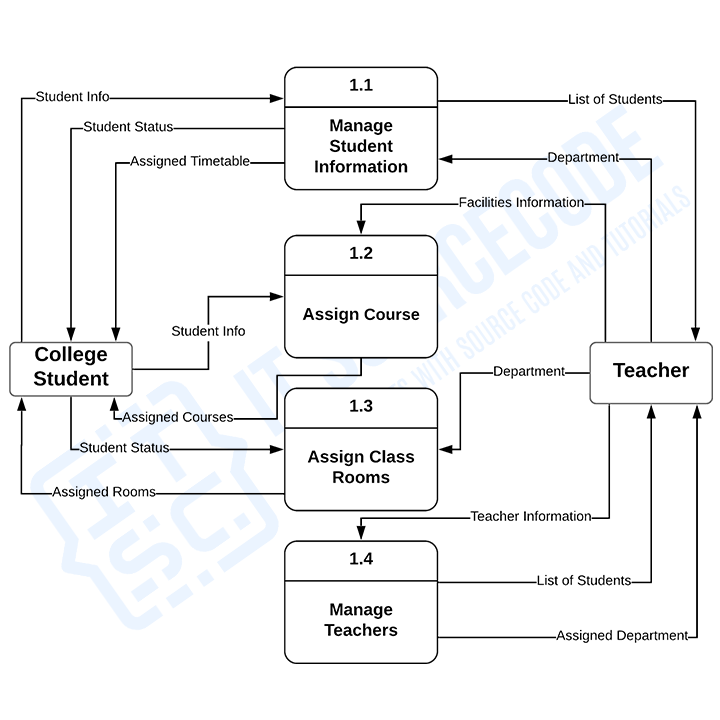
Structure English is derived from structured programming language which gives more understandable and precise description of process. It is based on procedural logic that uses construction and imperative sentences designed to perform operation for action.

* It is best used when sequences and loops in a program must be considered and the problem needs sequences of actions with decisions.
* It does not have strict syntax rule. It expresses all logic in terms of sequential decision structures and iterations.

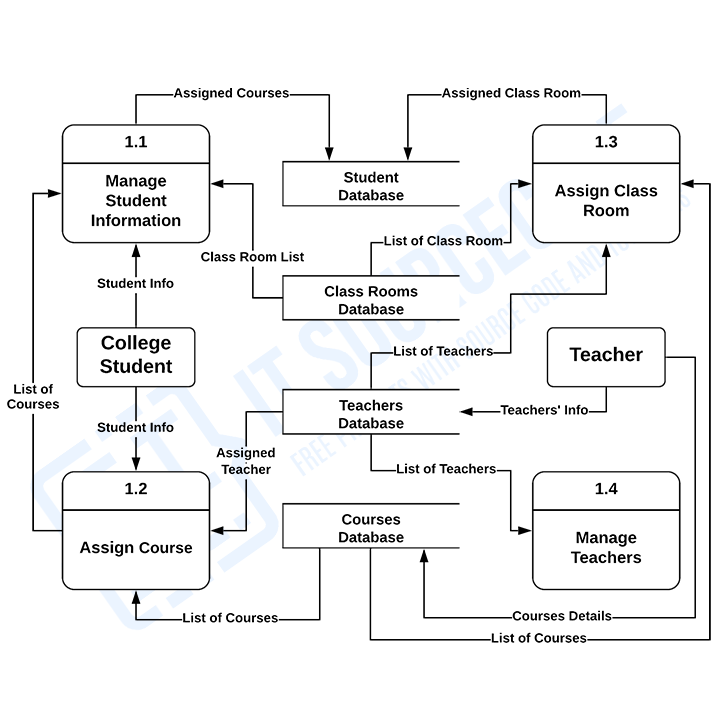
## **0 Level DFD –**

[](https://itsourcecode.com/wp-content/uploads/2022/09/image.png)

## **1 Level DFD –**

[](https://itsourcecode.com/wp-content/uploads/2022/09/image-1.png)

**2 Level DFD**

[](https://itsourcecode.com/wp-content/uploads/2022/09/image-2.png)

## **ER Diagram**

## **Database Design for the University Management System**

