

Chameli Devi Group of Institutions Department of Computer Science and Engineering Subject Notes CS405 Operating Systems UNIT-I Operating System An Operating System (OS) is an interface between computer user and computer hardware. An operating system is software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers. Examples-Some Popular Operating Systems include Linux Operating System, Windows Operating System, VMS, OS/400, AIX, z/OS, etc. Definition An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs. Fig 1.1 Operating System Operating System: Functions • Convenience: An OS makes a computer more convenient to use. • Efficiency: An OS allows the computer system resources to be used in an efficient manner. • Ability to Evolve: An OS should be constructed in such a way as to permit the effective development, testing and introduction of new system functions at the same time without interfering with service. Operating System: Evolutions The evolution of operating systems is directly dependent on the development of computer systems and how users use them. Here is a quick tour of computing systems through the past fifty years in the timeline.

GENERATION	YEAR	ELECTRONIC DEVICE USED	TYPES OF OS	DEVICE
First	1945–55	Vaccum Tubes	Plug Boards	
Second	1955–65	Transistors	Batch Systems	
Third	1965–80		Integrated Circuits(IC)	Multiprogramming
Fourth	Since 1980			Large Scale Integration PC

Early Evolution • 1945: ENIAC, Moore School of Engineering, University of Pennsylvania. • 1949: EDSAC and EDVAC • 1949: BINAC – a successor to the ENIAC • 1951: UNIVAC by Remington • 1952: IBM 701 • 1956: The interrupt • 1954–1957: FORTRAN was developed Operating Systems - Late 1950s By the late 1950s Operating systems were well improved and started supporting following usages: • It was able to perform Single stream batch processing. • It could use Common, standardized, input/output routines for device access. • Program transition capabilities to reduce the overhead of starting a new job were added. • Error recovery to clean up after a job terminated abnormally was added. • Job control languages that allowed users to specify the job definition and resource requirements were made possible. Operating Systems - In 1960s • 1961: The dawn of minicomputers • 1962: Compatible Time-Sharing System (CTSS) from MIT • 1963: Burroughs Master Control Program (MCP) for the B5000 system • 1964: IBM System/360 • 1960s: Disks became mainstream • 1966: Minicomputers got cheaper, more powerful, and really useful. • 1967–1968: Mouse was invented. • 1964 and onward: Multics • 1969: The UNIX Time-Sharing System from Bell Telephone Laboratories. Supported OS Features by 1970s • Multi User and Multi tasking was introduced. • Dynamic address translation hardware and Virtual machines came into picture. • Modular architectures came into existence. • Personal, interactive systems came into existence. Accomplishments after 1970 • 1971: Intel announces the microprocessor • 1972: IBM comes out with VM: the Virtual Machine Operating System • 1973: UNIX 4th Edition is published • 1973: Ethernet • 1974 The Personal Computer Age begins • 1974: Gates and Allen wrote BASIC for the Altair • 1976: Apple II • August 12, 1981: IBM introduces the IBM PC • 1983 Microsoft begins work on MS-Windows • 1984 Apple Macintosh comes out • 1990 Microsoft Windows 3.0 comes out • 1991 GNU/Linux • 1992 The first Windows virus comes out • 1993 Windows NT • 2007: iOS • 2008: Android OS And as the research and development work continues, we are seeing new operating systems being developed and existing ones getting improved and modified to enhance the overall user experience, making operating systems fast and efficient like never before. Also, with the onset of new devices like wearable, which includes, Smart Watches, Smart Glasses, VR gears etc, the demand for unconventional operating systems is also rising. Desirable Characteristics & Features of an Operating System Following are some of important features of an operating System. • Memory Management • Processor Management • Device Management • File Management • Security • Control over system performance • Job accounting • Error detecting aids • Coordination between other software and users

Memory Management Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address. Main memory provides a fast storage that can be accessed directly by the CPU. For a program to be executed, it must be in the main memory. An Operating System does the following activities for memory management – • Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part is not in use. • In multiprogramming, the OS decides which process will get memory when and how much. • Allocates the memory when a process requests it to do so. • De-allocates the memory when a process no longer needs it or has been terminated.

Processor Management In multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called process scheduling. An Operating System does the following activities for processor management – • Keeps tracks of processor and status of process. The program responsible for this task is known as traffic controller. • Allocates the processor (CPU) to a process. • De-allocates

processor when a process is no longer required. Device Management An Operating System manages device communication via their respective drivers. It does the following activities for device management – • Keeps tracks of all devices. Program responsible for this task is known as the I/O controller. • Decides which process gets the device when and for how much time. • Allocates the device in the efficient way. • De-allocates devices. File Management A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. An Operating System does the following activities for file management – • Keeps track of information, location, uses, status etc. The collective facilities are often known as file system. • Decides who gets the resources. • Allocates the resources. • De-allocates the resources. Other Important Characteristics Following are some of the important activities that an Operating System performs – • Security– By means of password and similar other techniques, it prevents unauthorized access to programs and data. • Control over system performance– Recording delays between request for a service and response from the system. • Job accounting– Keeping track of time and resources used by various jobs and users. • Error detecting aids– Production of dumps, traces, error messages, and other debugging and error detecting aids. • Coordination between other software's and users– Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems. Types of Operating System Operating systems are there from the very first computer generation and they keep evolving with time. In this chapter, we will discuss some of the important types of operating systems which are most commonly used.

**Batch Operating System**

The users of a batch operating system do not interact with the computer directly. Each user prepares his job on an off-line device like punch cards and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group. The programmers leave their programs with the operator and the operator then sorts the programs with similar requirements into batches. The problems with Batch Systems are as follows – • Lack of interaction between the user and the job. • CPU is often idle, because the speed of the mechanical I/O devices is slower than the CPU. • Difficult to provide the desired priority.

**Time-Sharing Operating Systems**

Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing. The main difference between Multi-programmed Batch Systems and Time-Sharing Systems is that in case of Multi-programmed batch systems, the objective is to maximize processor use, whereas in Time-Sharing Systems, the objective is to minimize response time. Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, the processor executes each user program in a short burst or quantum of computation. That is, if nusers are present, then each user can get a time quantum. When the user submits the command, the response time is in few seconds at most. The operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems have been modified to time-sharing systems.

**Advantages of Time-Sharing Operating Systems**

- Provides the advantage of quick response.
- Avoids duplication of software.
- Reduces CPU idle time.

**Disadvantages of Time-Sharing Operating Systems**

- Problem of reliability.
- Question of security and integrity of user programs and data.
- Problem of data communication.

**Distributed Operating System**

Distributed systems use multiple central processors to serve multiple real-time applications and multiple users. Data processing jobs are distributed among the processors accordingly. The processors communicate with one another through various communication lines (such as high-speed buses or telephone lines). These are referred as loosely coupled systems or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, computers, and so on.

**The advantages of Distributed Systems**

- With resource sharing facility, a user at one site may be able to use the resources available at another.
- Speedup the exchange of data with one another via electronic mail.
- If one site fails in a distributed system, the remaining sites can potentially continue operating.
- Better service to the customers.
- Reduction of the load on the host computer.
- Reduction of delays in data processing.

**Network Operating System**

A Network Operating System runs on a server and provides the server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks. Examples of network operating systems include Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD. The advantages of network operating systems are as follows –

- Centralized servers are highly stable.
- Security is server managed.

Upgrades to new technologies and hardware can be easily integrated into the system.

- Remote access to servers is possible from different locations and types of systems. The disadvantages of Network Operating Systems
- High cost of buying and running a server.
- Dependency on a central location for most operations.
- Regular maintenance and updates are required.

**Real Time Operating System** A real-time system is defined as a data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment. The time taken by the system to respond to an input and display of required updated information is termed as the response time. So, in this method, the response time is very less as compared to online processing. Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a dedicated application. A real-time operating system must have well-defined, fixed time constraints, otherwise the system will fail. For example, scientific experiments, medical image systems, industrial control systems, weapon systems, robots, air traffic control systems, etc. There are two types of real-time operating systems. **Hard Real-Time Systems** Hard real-time systems guarantee that critical tasks complete on time. In hard real-time systems, secondary storage is limited or missing and the data is stored in ROM. In these systems, virtual memory is almost never found. **Soft Real-Time Systems** Soft real-time systems are less restrictive. A critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example, multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers, etc.

**Operating Systems Services:** Types of Services

Operating system services are responsible for the management of platform resources, including the processor, memory, files, and input and output. They generally shield applications from the implementation details of the machine. Types of Operating system services include:

1. Kernel operations provide low-level services necessary to:
  - create and manage processes and threads of execution
  - execute programs
  - define and communicate asynchronous events
  - define and process system clock operations
  - implement security features
  - manage files and directories, and
  - Control input/output processing to and from peripheral devices.
2. Command interpreter and utility services include mechanisms for services at the operator level, such as:
  - comparing, printing, and displaying file contents
  - editing files
  - searching patterns
  - evaluating expressions
  - logging messages
  - moving files between directories
  - sorting data
  - executing command scripts
  - local print spooling
  - scheduling signal execution processes, and
  - Accessing environment information.
3. Batch processing services support the capability to queue work (jobs) and manage the sequencing of processing based on job control commands and lists of data. These services also include support for the management of the output of batch processing, which frequently includes updated files or databases and information products such as printed reports or electronic documents. Batch processing is performed asynchronously from the user requesting the job.
4. File and directory synchronization services allow local and remote copies of files and directories to be made identical. Synchronization services are usually used to update files after periods of off line working on a portable system.

**Operating System – Important Services**

- An Operating System provides services to both the users and to the programs.
- It provides programs an environment to execute.
- It provides users the services to execute the programs in a convenient manner.

Following are a few common services provided by an operating system –

- Program execution
- I/O operations
- File System manipulation
- Communication
- Error Detection
- Resource Allocation
- Protection

Program execution

Operating systems handle many kinds of activities from user programs to system programs like printer spooler, name servers, file server, etc. Each of these activities is encapsulated as a process. A process includes the complete execution context (code to execute, data to manipulate, registers, OS resources in use). Following are the major activities of an operating system with respect to program management –

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

**I/O Operation** An I/O subsystem comprises of I/O devices and their corresponding driver software. Drivers hide the peculiarities of specific hardware devices from the users. An Operating System manages the communication between user and device drivers.

- I/O operation means read or write operation with any file or any specific I/O device.
- Operating system provides the access to the required I/O device when required.

**File system manipulation** A file represents a collection of related information. Computers can store files on the disk (secondary storage), for long-term storage purpose. Examples of storage media include magnetic tape, magnetic disk and optical disk drives like CD, DVD. Each of these media has its own properties like speed, capacity, and data transfer rate and data access methods. A file system is normally organized into directories for

easy navigation and usage. These directories may contain files and other directions. Following are the major activities of an operating system with respect to file management – • Program needs to read a file or write a file. • The operating system gives the permission to the program for operation on file. • Permission varies from read-only, read-write, denied and so on. • Operating System provides an interface to the user to create/delete files. • Operating System provides an interface to the user to create/delete directories. • Operating System provides an interface to create the backup of file system.

**Communication** In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, the operating system manages communications between all the processes. Multiple processes communicate with one another through communication lines in the network. The OS handles routing and connection strategies, and the problems of contention and security. Following are the major activities of an operating system with respect to communication – • Two processes often require data to be transferred between them • Both the processes can be on one computer or on different computers, but are connected through a computer network. • Communication may be implemented by two methods, either by Shared Memory or by Message Passing. Error handling Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware. Following are the major activities of an operating system with respect to error handling – • The OS constantly checks for possible errors. • The OS takes an appropriate action to ensure correct and consistent computing.

**Resource Management** In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job. Following are the major activities of an operating system with respect to resource management – • The OS manages all kinds of resources using schedulers. • CPU scheduling algorithms are used for better utilization of CPU.

**Protection** Considering a computer system having multiple users and concurrent execution of multiple processes, the various processes must be protected from each other's activities. Protection refers to a mechanism or a way to control the access of programs, processes, or users to the resources defined by a computer system. Following are the major activities of an operating system with respect to protection – • The OS ensures that all access to system resources is controlled. • The OS ensures that external I/O devices are protected from invalid access attempts. • The OS provides authentication features for each user by means of passwords.

**Different ways of providing these Services** – • **Utility Programs** A program performs very specific tasks. These programs usually related to managing system resources. Operating systems contain a number of utilities for managing disk drives, printers, and other devices. Utilities differ from applications mostly in terms of size, complexity and function. For example, word processors, spreadsheet programs, and database applications are considered applications because they are large programs that perform a variety of functions not directly related to managing computer resources. Utilities are sometimes installed as memory-resident programs. Examples of utility programs are antivirus software, backup software and disk tools.

• **System Calls** A system call is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. A system call is a way for programs to interact with the operating system. A computer program makes a system call when it makes a request to the operating system's kernel. System call provides the services of the operating system to the user programs via Application Program Interface (API). It provides an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system. All programs needing resources must use system calls.

**Services Provided by System Calls:**

- Process creation and management
- Main memory management
- File Access, Directory and File system management
- Device handling(I/O)
- Protection
- Networking, etc.

**Types of System Calls:** There are 5 different categories of system calls –

- **Process control:** end, abort, create, terminate, allocate and free memory.
- **File management:** create, open, close, delete, read file etc.
- **Device management**
- **Information maintenance**
- **Communication**

**Operating System - Properties**

**Batch processing** Batch processing is a technique in which an Operating System collects the programs and data together in a batch before processing starts. An operating system does the following activities related to batch processing – • The OS defines a job which has predefined sequence of commands, programs and data as a single unit. • The OS keeps a number of jobs in memory and executes them without any manual intervention. • Jobs are processed in the order of submission, i.e., first come first served fashion. • When a job completes its execution, its memory is released and the output for the job gets copied into an output spool for later printing or processing.

**Fig 1.2 Batch processing Advantages**

- Batch processing takes much of the work of the operator to the computer.
- Increased performance as a new job gets started as soon as the previous job is finished, without any manual intervention.

**Disadvantages**

- Difficult to debug program.
- A job could enter an infinite loop.
- Due to lack of protection scheme, one batch job can affect pending jobs.

**Multitasking Multitasking**

is when multiple jobs are executed by the CPU simultaneously by switching between them. Switches occur so frequently that the users may interact with each program while it is running. An OS does the following activities related to multitasking – • The user gives instructions to the operating system or to a program directly, and receives an immediate response. • The OS handles multitasking in the way that it can handle multiple operations/executes multiple programs at a time. • Multitasking Operating Systems are also known as Time-sharing systems. • These Operating Systems were developed to provide interactive use of a computer system at a reasonable cost. • A time-shared operating system uses the concept of CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared CPU. • Each user has at least one separate program in memory. Fig 1.3 Multitasking • A program that is loaded into memory and is executing is commonly referred to as a process. • When a process executes, it typically executes for only a very short time before it either finishes or needs to perform I/O. • Since interactive I/O typically runs at slower speeds, it may take a long time to complete. During this time, a CPU can be utilized by another process. • The operating system allows the users to share the computer simultaneously. Since each action or command in a time-shared system tends to be short, only a little CPU time is needed for each user. • As the system switches CPU, rapidly from one user/program to the next, each user is given the impression that he/she has his/her own CPU, whereas actually one CPU is being shared among many users. Multiprogramming Sharing the processor, when two or more programs reside in memory at the same time, is referred as multiprogramming. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute. The following figure shows the memory layout for a multiprogramming system. Fig 1.4 Multiprogramming An OS does the following activities related to multiprogramming. • The operating system keeps several jobs in memory at a time. • This set of jobs is a subset of the jobs kept in the job pool. • The operating system picks and begins to execute one of the jobs in the memory. • Multiprogramming operating systems monitor the state of all active programs and system resources using memory management programs to ensure that the CPU is never idle, unless there are no jobs to process. Advantages • High and efficient CPU utilization. • User feels that many programs are allotted CPU almost simultaneously. Disadvantages • CPU scheduling is required. • To accommodate many jobs in memory, memory management is required. Interactivity Interactivity refers to the ability of users to interact with a computer system. An Operating system does the following activities related to interactivity – • Provides the user an interface to interact with the system. • Manages input devices to take inputs from the user. For example, keyboard. • Manages output devices to show outputs to the user. For example, Monitor. The response time of the OS needs to be short, since the user submits and waits for the result. Real Time System Real-time systems are usually dedicated, embedded systems. An operating system does the following activities related to real-time system activity. • In such systems, Operating Systems typically read from and react to sensor data. • The Operating system must guarantee response to events within fixed periods of time to ensure correct performance. Distributed Environment A distributed environment refers to multiple independent CPUs or processors in a computer system. An operating system does the following activities related to distributed environment – • The OS distributes computation logics among several physical processors. • The processors do not share memory or a clock. Instead, each processor has its own local memory. • The OS manages the communications between the processors. They communicate with each other through various communication lines. Spooling Spooling is an acronym for simultaneous peripheral operations on line. Spooling refers to putting data of various I/O jobs in a buffer. This buffer is a special area in memory or hard disk which is accessible to I/O devices. An operating system does the following activities related to distributed environment – • Handles I/O device data spooling as devices have different data access rates. • Maintains the spooling buffer which provides a waiting station where data can rest while the slower device catches up. • Maintains parallel computation because of spooling process as a computer can perform I/O in parallel fashion. It becomes possible to have the computer read data from a tape, write data to disk and to write out to a tape printer while it is doing its computing task. Fig 1.5 Spooling

Code inspections, Software Testing, Fundamentals, Software Test Process, Testing Levels, Test Criteria, Test Case Design, Test Oracles, Test Techniques, Testing Frameworks, Test Plan, Test Metrics, Testing Tools. Course Objective- To Describe the software Design phases and Software Testing Techniques Course Outcome-Understand Various Software Design Concept and Software Testing Techniques in Commercial Environment Introduction Software Design Process Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation. For assessing user requirements, an SRS (Software Requirement Specification) document is created whereas for coding and implementation, there is a need of more specific and detailed requirements in software terms. The output of this process can directly be used into implementation in programming languages. The architectural design defines the relationship between major structural elements of the software, the design pattern that can be used to achieve the requirements that have been defined by the system. Figure 2.1: Software Design Process The Design Process Software design is an iterative process through which requirements are translated into a “blueprint” for constructing the software. Initially, the blueprint depicts a holistic view of software. That is, the design is represented at a high level of abstraction at level that can be directly traced to the specific system objective and more detailed data, functional, and behavioral requirements. As design iterations occur, subsequent refinement leads to design representations at much lower level of abstraction. Design and Software Quality: - There are three characteristics for good design:

- The design must implement all the explicit requirements contained in the analysis model, and it must accommodate all the implicit requirements desired by the customer.
- The design must be being addable, understandable guide for those who generate code and for those who to stand subsequently support the software.
- The design should provide a complete picture of the software, addressing the data, functional, and behavioral domains from an implementation perspective.

Software Design Principles: - Software design is both a process and a model. The design process is a sequence of steps that enable the designer to describe all aspects of the software to be built. It is important to note, however, that the design process is not simply a cookbook. Creative skill, experience, a sense of what makes “good” software and an overall commitment to quality are critical success factors for a competent design. Principles for software design, which have been adapted and extended in the following list:

- The design process should not suffer from “tunnel vision.” A good designer should consider alternative approaches, judging each based on the requirements of the problem, the resources available to do the job.
- The design should be traceable to the analysis model. Because a single element of the design model often traces to multiple requirements, it is necessary to have a means for tracking how requirements have been satisfied by the design model.
- The design should not reinvent the wheel. Systems are constructed using a set of design patterns, many of which have likely been encountered before. These patterns should always be chosen as an alternative to reinvention. Time is short and resources are limited! Design time should be invested in representing truly new ideas and integrating those patterns that already exist.
- The design should “minimize the intellectual distance” between the software and the problem as it exists in the real world.
- That is, the structure of the software design should (whenever possible) mimic the structure of the problem domain.
- The design should exhibit uniformity and integration. A design is uniform if it appears that one person developed the entire thing. Rules of style and format should be defined for a design team before design work begins. A design is integrated if care is taken in defining interfaces between design components.

**SOFTWARE MODELING AND UML:** Software modeling: Software models are ways of expressing a software design. Usually some sort of abstract language or pictures are used to express the software design. For object-oriented software, an object modeling language such as UML is used to develop and express the software design. Unified Modeling Language (UML): Over the past decade, Grady Booch, James Rumbaugh, and Ivar Jacobson have collaborated to combine the best features of their individual object-oriented analysis and design methods into a unified method. The result, called the Unified Modeling Language (UML), has become widely used throughout the industry. UML allows a software engineer to express an analysis model using a modeling notation that is governed by a set of syntactic, semantic, and pragmatic rules. In UML, a system is represented using five different “views” that describe the system from distinctly different perspectives. Each view is defined by a set of diagrams. The following views are present in UML:

- User model view. This view represents the system (product) from the user’s (called actors in UML) perspective. The use-case is the modeling approach of choice for the user model view. This important analysis representation describes a usage scenario from the end-user’s perspective.
- Structural model view. Data and functionality are viewed from inside the system. That is, static structure (classes, objects, and relationships) is modeled.
- Behavioral model view. This part of the analysis model represents the dynamic or behavioral aspects of the system. It also depicts the interactions or collaborations between various structural elements described in

the user model and structural model views. Implementation model view. The structural and behavioral aspects of the system are represented as they are to be built. Environment model view. The structural and behavioral aspects of the environment in which the system is to be implemented are represented UML Diagram Types: There are several types of UML diagrams: User model view represent through: Use-case Diagram: Shows actors, use-cases, and the relationships between them. Structural model view represents through Class Diagram: Shows relationships between classes and pertinent information about classes themselves. Object Diagram: Shows a configuration of objects at an instant in time. Behavioral model view represents through Interaction Diagrams: Show an interaction between groups of collaborating objects.Two types: Collaboration diagram and sequence diagram Package Diagrams: Shows system structure at the library/package level. State Diagram: Describes behavior of instances of a class in terms of states, stimuli, and transitions. Activity Diagram: Very similar to a flowchart— shows actions and decision points, but with the abilityto accommodate concurrency. Environment model view represent through Deployment Diagram: Shows configuration of hardware and software in a distributed system. Implementation model view represent through Component Diagram: It shows code modules of a system. This code module includes application program, ActiveX control, Java beans and back end databases. It representing interfaces and dependencies among software architect. UML is a modeling language used to model software and non-software systems. Although UML is used for non-software systems, the emphasis is on modeling OO software applications. Most of the UML diagrams discussed so far are used to model different aspects such as static, dynamic, etc. Now whatever be the aspect, the artifacts are nothing but objects. If we look into class diagram, object diagram, collaboration diagram, interaction diagrams all would basically be designed based on the objects. Hence, the relation between OO design and UML is very important to understand. The OO design is transformed into UML diagrams according to the requirement. Before understanding the UML in detail, the OO concept should be learned properly. Once the OO analysis and design is done, the next step is very easy. The input from OO analysis and design is the input to UML diagrams. 1. Design of a system consists of classes, interfaces, and collaboration. UML provides class diagram, object diagram to support this. 2. Implementation defines the components assembled together to make a complete physical system. UML component diagram is used to support the implementation perspective. 3. Process defines the flow of the system. Hence, the same elements as used in Design are also used to support this perspective. 4. Deployment represents the physical nodes of the system that forms the hardware. UML deployment diagram is used to support this perspective.

**ARCHITECTURAL DESIGN:** Establishing the overall structure of a software system. Objectives: To introduce architectural design and to discuss its importance To explain why multiple models are required to document software architecture to describe types ofarchitectural model that may be used. A high-level model of a thing: - Describes critical aspects of the thing Understandable to many stakeholders Allows evaluation of the thing's properties before it is built Provides well understood tools and techniques for constructing the thing from its blueprint.

**ARCHITECTURAL VIEWS AND STYLES:** Architectural Styles: The builder has used an architectural style as a descriptive mechanism to differentiate the house from other styles (e.g., A-frame, raised ranch, Cape Cod). But more important, the architectural style is also a pattern for construction. Further details of the house must be defined, its final dimensions must be specified, customized features may be added, building materials are to be determined, but the pattern—a “center hall colonial”— guides the builder in his work. The commonly used architectural styles are:

**Data-centered Architectures:** A data store (e.g., a file or database) resides at the center of this architecture and is accessed frequently by other components that update, add, delete, or otherwise modify data within the store. A typical Data-centered style. Clients of the are accesses a central repository. In some cases the data repository is passive. That is, client software accesses the data independent of any changes to the data or the actions of other client software. A variation on this approach transforms the repository into a “blackboard”

that sends notifications to client software when data of interest to the client change. **Data Store (Repository) Client Software Filter Figure 2.2: Data Centered Architecture**  
**Data-flow Architectures:** This architecture is applied when input data are to be transformed through a series of computational or manipulative components into output data. A pipe and filter pattern has a set of components, called filters, connected by pipes that transmit data from one component to the next. Each filter works independently of those components upstream and downstream, is designed to expect data input of a certain form, and produces data output (to the next filter) of a specified form. However, the filter does not require knowledge of the working of its neighboring filters. If the data flow degenerates into a single line of transforms, it is termed batch sequential. This pattern accepts a

batch of data and then applies a series of sequential components (filters) to transform it. Piper Figure 2.3: Pipes and Filters Figure 2.4: Batch Sequential Call and return Architectures: The program structure can be easily modified or scaled. The program structure is organized into modules within the program. In this architecture how modules call each other. The program structure decomposes the function into control hierarchy where a main program invokes number of program components. User Interface Layer Application Layer Utility Layer Core Layer Components Figure 2.5: Call and Return Architecture Object-oriented Architecture: The components of a system encapsulate data and the operations. That must be applied to manipulate the data. Communication and coordination between components is accomplished via message passing Layered Architectures: The basic structure of a layered architecture is illustrated in Figure. A number of different layers are defined, each accomplishing operation that progressively become closer to the machine instruction set. At the outer layer, components service user interface operations. At the inner layer, components perform operating system interfacing. Intermediate layers provide utility services and application software functions. Figure 2.6: Layered Architecture Logical view Development view Scenarios Process view Physical view Architectural Views: 4+1 is a architectural view model used for "describing the architecture of software-intensive systems, based on the use of multiple, concurrent views". The views are used to describe the system from the viewpoint of different stakeholders, such as end-users, developers and project managers.

- Development view: The development view illustrates a system from a programmer's perspective and is concerned with software management. This view is also known as the implementation view. It uses the UML Component diagram to describe system components. UML Diagrams used to represent the development view include the Package diagram.
- Logical view: The logical view is concerned with the functionality that the system provides to end users. UML diagrams used to represent the logical view include, class diagrams, and state diagrams.
- Physical view: The physical view depicts the system from a system engineer's point of view. It is concerned with the topology of software components on the physical layer as well as the physical connections between these components. This view is also known as the deployment view. UML diagrams used to represent the physical view include the deployment diagram.
- Process view: The process view deals with the dynamic aspects of the system, explains the system processes and how they communicate, and focuses on the runtime behavior of the system. The process view addresses concurrency, distribution, integrators, performance, and scalability, etc. UML diagrams to represent process view include the activity diagram.

Figure 2.7: Architectural View USER INTERFACE DESIGN: User interface is the front-end application view to which user interacts in order to use the software. User can manipulate and control the software as well as hardware by means of user interface. Today, user interface is found at almost every place where digital technology exists, right from computers, mobile phones, cars, music players, airplanes, ships etc. User interface is part of software and is designed such a way that it is expected to provide the user insight of the software. User interface provides fundamental platform for human-computer interaction. User interface can be graphical, text-based, audio-video based, depending upon the underlying hardware and software combination. UI can be hardware or software or a combination of both. The software becomes more popular if its user interface is:

- Attractive
- Simple to use
- Responsive in short time
- Clear to understand
- Consistent on all interfacing screens

User Interface Design Principles: The principles of user interface design are intended to improve the quality of user interface design.

- The structure principle: Design should organize the user interface purposefully, in meaningful and useful ways based on clear, consistent models that are apparent and recognizable to users, putting related things together and separating unrelated things, differentiating dissimilar things and making similar things resemble one another. The structure principle is concerned with overall user interface architecture.
- The simplicity principle: The design should make simple, common tasks easy, communicating clearly and simply in the user's own language, and providing good shortcuts that are meaningfully related to longer procedures.
- The visibility principle: The design should make all needed options and materials for a given task visible without distracting the user with extraneous or redundant information. Good designs don't overwhelm users with alternatives or confuse with unneeded information.
- The feedback principle: The design should keep users informed of actions or interpretations, changes of state or condition, and errors or exceptions that are relevant and of interest to the user through clear, concise, and unambiguous language familiar to users.
- The tolerance principle: The design should be flexible and tolerant, reducing the cost of mistakes and misuse by allowing undoing and redoing, while also preventing errors wherever possible by tolerating varied inputs and sequences and by interpreting all reasonable actions.
- The reuse principle: The design should reuse internal and external components and behaviors, maintaining consistency with purpose rather than merely arbitrary consistency, thus reducing the need for users to rethink and remember.

User Interface Analysis and Design: User

interface analysis and design can be done with the help of following steps: Create different models for the system functions. In order to perform these functions, identify the human-computer interface tasks. Prepare all interface designs by solving various design issues. Apply modern tools and techniques to prototype the design. Implement design model. Evaluate the design from end user to bring quality in it. These steps can be broadly categorized in two classes. 1. Interface analysis and design models 2. The process Figure 2.8: User Interface Design

**FUNCTION-ORIENTED DESIGN:** In function-oriented design, the system is comprised of many smaller sub-systems known as functions. These functions are capable of performing significant task in the system. The system is considered as top view of all functions. Function oriented design inherits some properties of structured design where divide and conquer methodology is used. This design mechanism divides the whole system into smaller functions, which provides means of abstraction by concealing the information and their operation. These functional modules can share information among themselves by means of information passing and using information available globally.

**Functional Design Process:**

- Data-flow design: Model the data processing in the system using data-flow diagrams.
- Structural decomposition: Model how functions are decomposed to sub-functions using graphicalstructure charts.

**SA/ SD Component Based Design Structured Analysis and Structured Design:** Structured analysis is a set of techniques and graphical tools that allow the analyst to develop a new kind of system specification that are easily understandable to the user. Goals of SASD

- Improve Quality and reduce the risk of system failure
- Establish concrete requirements specifications and complete requirements documentation
- Focus on Reliability, Flexibility, and Maintainability of system

**Component Based Design:** Component-based architecture focuses on the decomposition of the design into individual functional or logicalcomponents that represent well-defined communication interfaces containing methods, events, and properties. It provides a higher level of abstraction and divides the problem into sub-problems, each associatedwith component partitions. The primary objective of component-based architecture is to ensure component reusability. A component encapsulates functionality and behaviors of a software element into a reusable and self-deployable binary unit. Component-oriented software design has many advantages over the traditional object-oriented approaches such as – Reduced time in market and the development cost by reusing existing components. Interface analysis and design models User model Design model Mental model Implementation model Increased reliability with the reuse of the existing components. Principles of Component-Based Design A component-level design can be represented by using some intermediary representation (e.g. graphical, tabular, or text-based) that can be translated into source code. The design of data structures, interfaces, and algorithms should conform to well-established guidelines to help us avoid the introduction of errors. It has following salient features – The software system is decomposed into reusable, cohesive, and encapsulated component units. Each component has its own interface that specifies required ports and provided ports; each component hides its detailed implementation. A component should be extended without the need to make internal code or design modifications to the existing parts of the component. Depend on abstractions component do not depend on other concrete components, which increase difficulty in expendability. Connectors connected components, specifying and ruling the interaction among components. The interaction type is specified by the interfaces of the components. Components interaction can take the form of method invocations, asynchronous invocations, broadcasting, message driven interactions, data stream communications, and other protocol specific interactions. For a server class, specialized interfaces should be created to serve major categories of clients. Only those operations that are relevant to a particular category of clients should be specified in the interface.

**DESIGN METRICS** In software development, a metric is the measurement of a particular characteristic of a program's performance or efficiency. Design metric measure the efficiency of design aspect of the software. Design model considering three aspects:

- Architectural design
- Object oriented design
- User interface design

**Architectural Design Metrics:** Architectural design metrics focus on characteristics of the program architecture with an emphasis on the architectural structure and the effectiveness of modules. These metrics are black box in the sense that they donot require any knowledge of the inner workings of a particular software component.

**Object Oriented Design Metrics:** There are nine measurable characteristics of object-oriented design and those are:

- **Size:** It can be measured using following factors:
- **Population:** means total number of classes and operations.
- **Volume:** means total number of classes or operation that is collected dynamically.
- **Length:** means total number of interconnected design elements.
- **Functionality:** is a measure of output delivered to the customer.
- **Complexity:** It is measured representing the characteristics that how the classes are interrelated with each other.
- **Coupling:** It is a measure stating the collaboration between classes or number of messages that can be passed between the objects.
- **Completeness:** It is a measure representing all the requirements of the design component.
- **Cohesion:** It is a degree by which we can

identified the set of properties that are working together to solve particular problem. Sufficiency: It is a measure representing the necessary requirements of the design component. Primitiveness: The degree by which the operations are simple, i.e. number of operations independent from other. Similarity: the degree by which we measure that two or more classes are similar with respect to their functionality and behavior. Volatility: Is the measure that represents the probability of changes that will occur. User Interface Design Metrics: Although there is significant literature on the design of human/computer interfaces, relatively little information has been published on metrics that would provide insight into the quality and usability of the interface. Layout appropriateness (LA) is a worthwhile design metric for human/computer interfaces. A typical GUI uses layout entities—graphic icons, text, menus, windows, and the like—to assist the user in completing tasks. To accomplish a given task using a GUI, the user must move from one layout entity to the next. Cohesion metrics can be defined as the relative connection of on screen content to other screen contents. UI cohesion for screen is high.

**SOFTWARE STATIC AND DYNAMIC ANALYSIS:** Static Analysis: Static analysis involves no dynamic execution of the software under test and can detect possible defects in an early stage, before running the program. Static analysis is done after coding and before executing unit tests. Static analysis can be done by a machine to automatically “walk through” the source code and detect non-complying rules. The classic example is a compiler which finds lexical, syntactic and even some semantic mistakes. Static analysis can also be performed by a person who would review the code to ensure proper coding standards and conventions are used to construct the program. Dynamic Analysis: Dynamic analysis is based on the system execution, often using tools. Dynamic program analysis is the analysis of computer software that is performed with executing programs built from that software on a real or virtual processor (analysis performed without executing programs is known as static code analysis). Dynamic program analysis tools may require loading of special libraries or even recompilation of program code. The most common dynamic analysis practice is executing Unit Tests against the code to find any errors in codes.

**CODE INSPECTIONS:** Code Inspection is the most formal type of review, which is a kind of static testing to avoid the defect multiplication at a later stage. The main purpose of code inspection is to find defects and it can also spot any process improvement if any. An inspection report lists the findings, which include metrics that can be used to aid improvements to the process as well as correcting defects in the document under review. Preparation before the meeting is essential, which includes reading of any source documents to ensure consistency. Inspections are often led by a trained moderator, who is not the author of the code. The inspection process is the most formal type of review based on rules and checklists and makes use of entry and exit criteria. It usually involves peer examination of the code and each one has a defined set of roles. After the meeting, a formal follow-up process is used to ensure that corrective action is completed.

**SOFTWARE TESTING FUNDAMENTALS:** Software testing is an activity performed to uncover errors. It is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The purpose of software testing is to ensure whether the software functions appear to be working according to specification and performance requirements.

**Testing objective:** Testing is a process of executing a program with the intent of finding an error. A good test case is one that has high probability of finding an undiscovered error. A successful test is one that uncovers an as yet undiscovered error.

**Testing principles:** The following basic principles and fundamentals are general guidelines applicable for all types of real-time testing:

- Testing proves the presence of defects. It is generally considered better when a test reveals defects than when it is error-free.
- Testing the product should be accomplished considering the risk factor and priorities.
- Early testing helps identify issues prior to the development stage, which eases error correction and helps reduce cost.
- Normally a defect is clustered around a set of modules or functionalities. Once they are identified, testing can be focused on the defective areas, and yet continue to find defects in other modules simultaneously.
- Testing will not be as effective and efficient if the same kinds of tests are performed over a long duration.
- Testing has to be performed in different ways, and cannot be tested in a similar way for all modules.
- All testers have their own individuality, likewise the system under test.
- Just identifying and fixing issues does not really help in setting user expectations.
- Even if testing is performed to showcase the software's reliability, it is better to assume that none of the software products are bug-free.

**SOFTWARE TEST PROCESS:** Testing is a process rather than a single activity. This process starts from test planning then designing test cases, preparing for execution and evaluating status till the test closure. So, we can divide the activities within the fundamental test process into the following basic steps:

- Planning and Control
- Analysis and Design
- Implementation and Execution
- Evaluating exit criteria and Reporting
- Test Closure activities

**TESTING LEVELS:** - The testing can be typically carried out in levels. In software development process at each phase some faults may get introduced. These faults are eliminated in the next software development phase but at the

same timesome new faults may get introduced. Each level of testing performs some typical activity. Levels of testing include different methodologies that can be used while conducting software testing. The main levels of software testing are:

- Unit Testing**
- Integration Testing**
- System Testing**
- Acceptance Testing**
- Unit Testing:** In this type of testing errors are detected from each software component individually.
- Integration Testing:** In this type of testing technique interacting component are verified and the interface errors are detected.
- System Testing:** In this type of testing all the system elements forming the system is tested as a whole.
- Acceptance Testing:** Acceptance testing is a kind of testing conducted to ensure that the software works correctly in user's working environment.

**Figure 2.10: Levels of Testing TEST CRITERIA AND TEST CASE DESIGN:** Test cases are used to determine the presence of fault in the program. Executing test cases require money because

- 1) machine time is required to execute test cases
- 2) Human efforts are involved in executing test case.

Hence in the project testing minimum number of test cases should be there as far as possible. The testing activity should involve two goals-

- 1) Maximize the number of errors detected.
- 2) Minimize the number of test cases.

The selection of test case should be such that faulty module or program segment must be exercised by at least one test case. Test selection criterion can be defined as the set of conditions that must be satisfied by the set of test cases. Testing criterion is based on two fundamental properties – reliability and validity. A test criterion is reliable if all the test cases detect same set of errors. A test criterion is valid if, for any error in the program there is some test case which causes error in the program. Generating test cases to satisfy criteria is complex task.

**TEST ORACLES:** Test Oracles is a mechanism for determining whether a test has passed or failed. The use of oracles involves comparing the output(s) of the system under test, for a given test-case input, to the output(s) that the oracle determines that product should have. Suppose we have written 2 test cases one test case is for the program which we want to test and other for the test oracle. If the output of both is the same then that means program behaves correctly otherwise there is some fault in the program.

**Figure 2.11: Test Oracle TEST TECHNIQUES:** There are various testing techniques available in which internal structure/design/implementation of the item being tested. There are different methods that can be used for software testing.

- Black box testing**
- White box testing**
- Integration testing**
- Unit testing**

**BLACK BOX TESTING:** Black box testing is also called as behavioral testing. Black-box testing is a method of software testing that examines the functionality of an application based on the specifications. It is also known as Specifications based testing. Independent Testing Team usually performs this type of testing during the software testing life cycle. This method of test can be applied to each and every level of software testing such as system and acceptance testing. Black box testing uncovers following types of errors:

- Incorrect or missing functions
- Interface errors
- Errors in data structures
- Performance errors
- Initialization or termination errors

There are different techniques involved in Black Box testing. Equivalence partitioning, Boundary Value Analysis, Cause-Effect Graphing, Error-Guessing.

**WHITE BOX TESTING:** White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called glass testing or open-box testing. In order to perform white-box testing on an application, a tester needs to know the internal workings of the code. The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately. White box testing techniques includes:

- Statement Coverage - This technique is aimed at exercising all programming statements with minimal tests.
- Branch Coverage - This technique is running a series of tests to ensure that all branches are tested atleast once.

**UNIT TESTING:** Unit testing, a testing technique using which individual modules are tested to determine if there are any issues by the developer himself. It is concerned with functional correctness of the standalone modules. The main aim is to isolate each unit of the system to identify, analyze and fix the defects.

**Black Box Testing -** Using which the user interface, input and output are tested.

**White Box Testing -** used to test each one of those functions behavior is tested.

**Gray Box Testing -** Used to execute tests, risks and assessment methods.

**TESTING FRAMEWORKS:** Testing frameworks are an essential part of any successful automated testing process. They can reduce maintenance costs and testing efforts and will provide a higher return on investment (ROI) for QA teams looking to optimize their agile processes. A testing framework is a set of guidelines or rules used for creating and designing test cases. A framework is comprised of a combination of practices and tools that are designed to help QA professionals test more efficiently.

**Table 2.1: Test Case Test Planning Activities**

To determine the scope and the risks that need to be tested and that are NOT to be tested.

**Documenting Test Strategy.** Making sure that the testing activities have been included.

**S. No. Parameter Description**

- 1. Test plan identifier Unique identifying reference.
- 2. Introduction A brief introduction about the project and to the document.
- 3. Test items A test item is a software item that is the application under test.
- 4. Features to be tested A feature that needs to be tested on the test ware.
- 5. Features not to be tested Identify the features and the reasons for not including as part of testing.
- 6. Approach Details about the overall

approach to testing. 7. Item pass/fail criteria Documented whether a software item has passed or failed its test. 8. Test deliverables The deliverables that are delivered as part of the testing process, such as test plans, test specifications and test summary reports. 9. Testing tasks All tasks for planning and executing the testing. 10. Environmental needs Defining the environmental requirements such as hardware, software, OS, network configurations, tools required. 11. Responsibilities Lists the roles and responsibilities of the team members. 12. Staffing and training needs Captures the actual staffing requirements and any specific skills and training requirements. 13. Schedule States the important project delivery dates and key milestones. 14. Risks and Mitigation High-level project risks and assumptions and a mitigating plan for each identified risk. 15. Approvals Captures all approvers of the document, their titles and the sign off date. Deciding Entry and Exit criteria. Evaluating the test estimate. Planning when and how to test and deciding how the test results will be evaluated, and defining test exit criterion. The Test artifacts delivered as part of test execution. Defining the management information, including the metrics required and defect resolution and risk issues. Ensuring that the test documentation generates repeatable test assets.

**TEST METRICS:** In software testing, Metric is a quantitative measure of the degree to which a system, system component, or process possesses a given attribute. Measurement is nothing but quantitative indication of size / dimension / capacity of an attribute of a product / process.

**Measure - No. of Errors**

**Metrics - No. of Errors** found per person

The most commonly used metric is cyclomatic complexity and Hallstead complexity.

**Cyclomatic complexity:** Cyclomatic complexity is a software metric used to measure the complexity of a program. This metric measures independent paths through program source code. Independent path is defined as a path that has at least one edge which has not been traversed before in any other paths. Cyclomatic complexity can be calculated with respect to functions, modules, methods or classes within a program.

**TESTING TOOLS:** Tools from a software testing context can be defined as a product that supports one or more test activities right from planning, requirements, creating a build, test execution, defect logging and test analysis.

**Classification of Tools**

Tools can be classified based on several parameters. They include:

- The purpose of the tool
- The Activities that are supported within the tool
- The Type/level of testing it supports
- S. No.
- Tool Type Used for
- Used by

1. Test Management Tool
2. Test Managing, scheduling, defect logging, tracking and analysis.
3. Testers
4. Configuration management tool
5. For Implementation, execution, tracking changes
6. All Team members
7. Static Analysis Tools
8. Static Testing Developers
9. Test Tool data Preparation
10. Analysis and generation Design
11. Test data Testers
12. 5. Test Execution Tools
13. Implementation, Execution Testers
14. 6. Test Comparators
15. Comparing expected and actual results
16. All Team members

**Table 2.2: Test Tools**

1. 7. Cover age measurement tools
2. Provides structural coverage
3. Developers
4. 8. Performance Testing tools
5. Monitoring the performance, response time
6. Testers
7. 9. Project planning and Tracking Tools
8. For Planning
9. Project Managers
10. 10. Incident Management Tools
11. For managing the tests
12. Testers

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Name: Data Science Subject Code: AD 404 Subject Notes Syllabus: Introduction to Data Science-Types of Data: structured and unstructured data, Data Science Road Map: Frame the Problem; Understand the Data, Data Wrangling, Exploratory Analysis, Extract Features, Model and Deploy Code. Graphical Summaries of Data: Pie Chart, Bar Graph, Pareto Chart, Histogram. Measures of central tendency of Quantitative Data: Mean, Median, Mode. Measures of Variability of Quantitative Data: Range, Standard Deviation and Variance. Probability: Introduction to Probability, Conditional Probability.

Course

**Objectives:** • The purpose of this subject is to cover the underlying concepts and techniques used in Data Science. Some of these techniques can be used in Data Analysis & in prediction. • To understand modern way to get insights from the data.

Course

**Outcome (CO1):** To expose students to various perspectives and concepts in Data Science. Unit-I Data Science Data Science is a combination of mathematics, statistics, machine learning, and computer science. Data Science is collecting, analyzing and interpreting data to gather insights into the data that can help decisionmakers make informed decisions. Data Science is used in almost every industry today that can predict customer behavior and trends and identify new opportunities. Businesses can use it to make informed decisions about product development and marketing. It is used as a tool to detect fraud and optimize processes. Governments also use Data Science to improve efficiency in the delivery of public services. In simple terms, Data Science helps to analyze data and extract meaningful insights from it by combining

statistics & mathematics, programming skills, and subject expertise. Data In general, data is a distinct piece of information that is gathered and translated for some purpose. Data can be available in different forms, such as bits and bytes stored in electronic memory, numbers or text on pieces of paper, or facts stored in a person's mind. Structured Data The data which is to the point, factual, and highly organized is referred to as structured data. It is quantitative in nature, i.e., it is related to quantities that means it contains measurable numerical values like numbers, dates, and times. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Figure-1.1 :Structured Data It is easy to search and analyze structured data. Structured data exists in a predefined format. Relational database consisting of tables with rows and columns is one of the best examples of structured data. Structured data generally exist in tables like excel files and Google Docs spreadsheets. The programming language SQL (structured query language) is used for managing the structured data. SQL is developed by IBM in the 1970s and majorly used to handle relational databases and warehouses. Structured data is highly organized and understandable for machine language. Common applications of relational databases with structured data include sales transactions, Airline reservation systems, inventory control, and others.

Unstructured Data All the unstructured files, log files, audio files, and image files are included in the unstructured data. Some organizations have much data available, but they did not know how to derive data value since the data is raw.

Figure-1.2 :Unstructured Data Unstructured data is the data that lacks any predefined model or format. It requires a lot of storage space, and it is hard to maintain security in it. It cannot be presented in a data model or schema. That's why managing, analyzing, or searching for unstructured data is hard. It resides in various different formats like text, images, audio and video files, etc. It is qualitative in nature and sometimes stored in a non-relational database or NOSQL. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science It is not stored in relational databases, so it is hard for computers and humans to interpret it. The limitations of unstructured data include the requirement of data science experts and specialized tools to manipulate the data. The amount of unstructured data is much more than the structured or semi-structured data. Examples of human-generated unstructured data are Text files, Email, social media, media, mobile data, business applications, and others. The machine-generated unstructured data includes satellite images, scientific data, sensor data, digital surveillance, and many more. Data Science Road Map The process of solving a data science problem is summarized in the following figure, which I called the Data Science Road Map. Figure-1.3 : Data Science Road Map

1. Frame the data science problem This is the first and important step when undertaking any data science task. Many Great data scientist such as Dean Abbott, founder of Abbott Analytics, have mastered this art of asking the right questions before solving the any problem. There is no amount of technical expertise or statistical rigor that can compensate for the fact that you addressed a pointless problem. We need to identify the type of client we are dealing with, whether human or machine. Most initiatives begin with a pretty open-ended query if your clients are humans. There may be a known problem, but it's unclear what a solution would entail. If your clients are machines, the business challenge is usually obvious, but there can be some doubt about the software limitations (languages to employ, runtime, how accurate predictions must be, and so on). It's crucial to define exactly what Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science constitutes a solution to this problem before getting down to business. It's crucial to define exactly what constitutes a solution to this problem before getting down to business.

2. Understand the data. Understanding the data can be categorized into 3 sub-groups. Once undergone this steps you will have full understanding of your data and it's useful characteristics and features. These sub-groups include;

- Basic Questions
- Data wrangling
- Exploratory analysis

Basic Questions Before taking your data under any statistical analysis, you will need to have a good bundle of questions to ask about your data. This helps by making sure that you are working on the right problem and with the right dataset. Here is a list of some of the generic questions you can ask.

- Is this data representative enough? For example, maybe data was only collected for a subset of users.
- Are there likely to be gross outliers or extraordinary sources of noise?
- Are there any fields that are unique identifiers?
- Is this the entire dataset?

Data Wrangling Data wrangling is the process of transforming raw data into something that can be used for more conventional analytics. Most times as a data scientist you will not alwalys be tasked with data that is in the best way to work on your analysis. This whole process means that you will be constructing a software pipeline that extracts data from wherever it is stored, does any necessary cleaning or filtering, and converts it to a standard format. Exploratory analysis After transforming your data into a more suitable format, the next step is exploratory data analysis. This will be achieved by using a little bit of your statistics knowledge. Typically exploratory analysis is using visual methods to present some of the basic or complex characteristics of your data. Some of the statistics concepts that you can employ for your exploratory

analysis are correlation which you can visualize in form of heatmaps, mean, mode, quantile, skewness and kurtosis. 3. Extract features from the data. A feature is essentially a number or a category extracted from your data that describes an entity. You could, for example, extract the average word length or the number of characters in a text document. The most critical aspect of getting your analysis to function is extracting good features. A particularly good feature will Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science usually connect to a real-world phenomena. The majority of the information we extract will be used to make a prediction. You may, however, need to extract the object you're predicting, also known as the target variable. 4. Creating a model When deciding on the type of model to implement. We first off understand the type of machine learning algorithm that will blend well with our problem. Examples of these algorithms, a regression model that predicts a stock's price on the next day, or a clustering algorithm that breaks customers into different segments. When working on your model just make sure it is tuned to solve the particular problem you are trying to solve. 5. Present your results/findings If your client is human, you will need to present your results in form of reports describing what your findings were. Most times, you will need to present your reports in form of graphs or any other visual method. This is because most of the people you will be working with will not be as technical as you are. 6. Deployment of your code Deployment of your code stands to be the last step in the data science road map. This stage, you will need to present your model to a form that the user can be able to understand. How do you achieve this? This is very simple since most of the hard-work you have already taken care of. All you need to do is design an interface that will make your code understandable to your user. You can make your code understandable to the user through web app or mobile app. Graphical Summaries of Data Many powerful approaches to data analysis communicate their findings via graphs. These are an important counterpart to data analysis approaches that communicate their findings via numbers or tables. A graph can be either informative or misleading, just like any other type of statistical result. To understand whether a graph is informative, we should consider the following:

- Every graph should provide insight into the specific research question that is the overall goal of the data analysis.
- The graph is constructed using a sample of data, but the purpose of the graph is to learn about the population that the sample represents.
- What statistical principle or concept is the graph based on?
- What are the theoretical properties of any numerical summaries that are shown in the graph?

Pie Chart A pie chart is a graphical representation technique that displays data in a circular-shaped graph. It is a composite static chart that works best with few variables. Pie charts are often used to represent sample data—with data points belonging to a combination of different categories. Each of these categories is Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science represented as a “slice of the pie.” The size of each slice is directly proportional to the number of data points that belong to a particular category. Figure-1.4 : Pie Chart Bar Graph A bar graph is a graphical representation of information. It uses bars that extend to different heights to depict value. Bar graphs can be created with vertical bars, horizontal bars, grouped bars (multiple bars that compare values in a category), or stacked bars (bars containing multiple types of information). Bar graphs are commonly used in business and financial analysis to display often complicated data. They can convey information quickly and effectively. In the financial industry, a volume chart is a commonly used vertical bar graph. Figure-1.5 : Bar Graph Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Pareto Chart A Pareto chart is a bar graph. The lengths of the bars represent frequency or cost (time or money), and are arranged with longest bars on the left and the shortest to the right. In this way the chart visually depicts which situations are more significant. Figure 1.6: Pareto Chart, Customer Complaints

Figure 1.7: Pareto Chart, Document Complaints Histogram In statistics, a histogram is a graphical representation of the distribution of data. The histogram is represented by a set of rectangles, adjacent to each other, where each bar represents a kind of data. Statistics is a stream of mathematics that is applied in various fields. Figure 1.8: Histogram Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Measures of central tendency of Quantitative Data- Mean, Median, and Mode mean, median, and mode, in mathematics, the three principal ways of designating the average value of a list of numbers. The arithmetic mean is found by adding the numbers and dividing the sum by the number of numbers in the list. This is what is most often meant by an average. The median is the middle value in a list ordered from smallest to largest. The mode is the most frequently occurring value on the list. Measures of Variability of Quantitative Data- Range, Standard Deviation and Variance

- Range: the difference between the highest and lowest values
- Interquartile range: the range of the middle half of a distribution
- Standard deviation: average distance from the mean
- Variance: average of squared distances from the mean

Probability: Probability can be defined as a chance that an uncertain event will occur. It is the numerical measure of the likelihood that an event will occur. The value of probability

always remains between 0 and 1 that represent ideal uncertainties.  $0 \leq P(A) \leq 1$ , where  $P(A)$  is the probability of an event A.  $P(A) = 0$ , indicates total uncertainty in an event A.  $P(A) = 1$ , indicates total certainty in an event A. Probability: We can find the probability of an uncertain event by using the below formula.  $\square P(\neg A) = \text{probability of a not happening event}$ .  $\square P(\neg A) + P(A) = 1$ . Some important terms Event: Each possible outcome of a variable is called an event. Sample space: The collection of all possible events is called sample space. Random variables: Random variables are used to represent the events and objects in the real world. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Prior Probability: The prior probability of an event is probability computed before observing new information. Posterior Probability: The probability that is calculated after all evidence or information has taken into account. It is a combination of prior probability and new information. Conditional Probability: Conditional probability is a probability of occurring an event when another event has already happened. Let's suppose, we want to calculate the event A when event B has already occurred, "the probability of A under the conditions of B", it can be written as: Where  $P(A \wedge B) = \text{Joint probability of A and B}$   $P(B) = \text{Marginal probability of B}$ . Marginal Probability: the probability of an event occurring ( $p(B)$ ), it may be thought of as an unconditional probability. It is not conditioned on another event. Joint Probability:  $p(A \text{ and } B)$ . The probability of event A and event B occurring. It is the probability of the intersection of two or more events. The probability of the intersection of A and B may be written  $p(A \cap B)$ . If the probability of A is given and we need to find the probability of B, then it will be given as: It can be explained by using the below Venn diagram, where B is occurred event, so sample space will be reduced to set B, and now we can only calculate event A when event B is already occurred by dividing the probability of  $P(A \wedge B)$  by  $P(B)$ . Figure 1.9: Joint Probability

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Name: Data Science Subject Code: AD 404 Subject Notes Syllabus: Unstructured Data Analytics- Importance of Unstructured Data, Unstructured Data Analytics: Descriptive, diagnostic, predictive and prescriptive data Analytics based on Case study. Data Visualization: boxplots, histograms, scatterplots, features map visualization, t-SNE . Overview of Advance Excel- Introduction, Data validation, Introduction to charts, pivot table, Scenario manager, Protecting data, Excel minor, Introduction to macros.

Course

Objectives: • The purpose of this subject is to cover the underlying concepts and techniques used in Data Science. Some of these techniques can be used in Data Analysis & in prediction. • To understand modern way to get insights from the data.

Course

Outcome (CO2): To Understand the Concept of Advance Excel. Unit-2 Unstructured Data Unstructured data are datasets that have not been structured in a predefined manner. Unstructured data is typically textual, like open-ended survey responses and social media conversations, but can also be non-textual, like images, video, and audio. Unstructured information is growing quickly due to increased use of digital applications and services. Some estimates say that 80-90% of company data is unstructured, and it continues to grow at an alarming rate per year. While structured data is important, unstructured data is even more valuable to businesses if analyzed correctly Examples of Unstructured Data • Business documents • Spreadsheets • Emails • Social media posts • Open-ended surveys • Website landing pages • Images • Audio recordings • Text messages • Chat • Video files Semi-Structured Data Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Somewhere in-between structured and unstructured data lies semi-structured data. This type of data is also typically heavy but loosely categorised with some type of tagging structure. This data can be divided into groups, separating various elements and enabling search, but the data within these groups lacks structure. The line between unstructured and semi-structured data can be a blurry one, however, as some argue that all data has some degree of structure. Importance of Unstructured Data ♣ Customer Experience and Sentiment Information found in unstructured data sources can help businesses improve the customer experience by monitoring and extract important insights from phone calls, customer service chats, comments on social media or emails. Within text and audio-rich data lies a wealth of knowledge on customer sentiment of your brand and buyer pain points. ♣ Marketing Intelligence Unstructured data can also be a gold mine of marketing intelligence. There is lots of information waiting to be discovered

that tell businesses about patterns in customer behaviour and what types of product offerings or services are the most attractive to their audience. This has a direct effect on product development and other marketing initiatives and campaigns.

- ♣ R&D and Innovation Opportunities News reports, blogs, social media posts and comments and online reviews of competitors are all forms of unstructured information that provides a treasure trove of information about industry and market trends. Uncovering this information before your competitors puts you in a prime position to anticipate changes and adjust.
- ♣ Regulatory Compliance Digging deeper into unstructured data can also help highly-regulated organisations discover possible compliance issues before they have a negative impact on business.

**Unstructured Data Analytics** Unstructured data refers to any information that doesn't have a defined data model or format, and is not organized in a specific way. This type of data typically includes text-based data such as emails, social media posts, and documents, as well as multimedia data like images, videos, and audio files. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Unstructured data analytics refers to the process of extracting insights and useful information from unstructured data sources. This involves using various techniques, such as natural language processing, machine learning, and text mining, to identify patterns, relationships, and trends within the data. Unstructured data analytics can be applied across a wide range of industries and use cases. For example, businesses can use it to analyze customer feedback and sentiment on social media, to extract insights from email and chat logs, or to analyze video or image data for security purposes. Governments and other organizations can also use unstructured data analytics to monitor public sentiment and to identify potential risks or threats. Most businesses typically perform the following types of analytics to solve a business problem:

- Descriptive analytics: Describes what happened in the past using summary and statistics
- Diagnostic analytics: Uncovers why it happened using data discovery, drill-down, data mining
- Predictive analytics: Predicts what might happen in the future using machine learning models and forecasting
- Prescriptive analytics: Suggests a corrective course of action using predictive analytics, machine learning, and artificial intelligence (AI) for decision-making

**Data Visualization** In simple terms, Data Visualization (DataViz) is the process of generating graphical representations of data for various purposes. These graphical representations are commonly known as plots or charts in data science terminology.

**Importance of Data Visualization in Data Science** There are many reasons for data visualization in data science. Data visualization benefits include communicating your results or findings, monitoring the model's performance at the evaluation stage, hyperparameter tuning, identifying trends, patterns and correlation between dataset features, data cleaning such as outlier detection, and validating model assumptions.

**Different Types of Data Visualization**

1. Box and whisker plot This plot is used to plot the variation of the values of a numerical feature. You can get the values' minimum, maximum, median, lower and upper quartiles.
- Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Figure: 2.1 Box and whisker plot
2. Histogram A histogram represents the distribution of numerical data. Looking at a histogram, we can decide whether the values are normally distributed (a bell-shaped curve), skewed to the right or skewed left. A histogram of residuals is useful to validate important assumptions in regression analysis.
- Figure: 2.2 Histogram
3. Scatter plot Scatter plots are created to see whether there is a relationship (linear or non-linear and positive or negative) between two numerical variables. They are commonly used in regression analysis.
- Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Figure: 2.3 Scatter plot
4. Features Map Feature mapping, also known as feature engineering, is the process of transforming raw input data into a set of meaningful features that can be used by a machine learning algorithm. Feature mapping is an important step in machine learning, as the quality of the features can have a significant impact on the performance of the algorithm.
- Figure: 2.4 Features Map
- Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science
5. t-SNE T-distributed Stochastic Neighbourhood Embedding (tSNE) is an unsupervised Machine Learning algorithm developed in 2008 by Laurens van der Maaten and Geoffery Hinton. It has become widely used in bioinformatics and more generally in data science to visualise the structure of high dimensional data in 2 or 3 dimensions. It is the best known of a group of algorithms called Manifold Learning which are used for non-linear dimensionality reduction. It is sometimes contrasted with Principal Component Analysis (PCA).
- t-SNE vs PCA The first thing to note is that PCA was developed in 1933 while t-SNE was developed in 2008. A lot has changed in the world of data science since 1933 mainly in the realm of compute and size of data.
- Second, PCA is a linear dimension reduction technique that seeks to maximize variance and preserves large pairwise distances. In other words, things that are different end up far apart. This can lead to poor visualization especially when dealing with non-linear manifold structures.
- Think of a manifold structure as any geometric shape like: cylinder, ball, curve, etc. t-SNE differs from PCA by preserving only small distances between points that are close in the original high-dimensional space.

pairwise distances or local similarities whereas PCA is concerned with preserving large pairwise distances to maximize variance. Laurens illustrates the PCA and t-SNE approach pretty well using the Swiss Roll dataset in Figure . You can see that due to the non-linearity of this toy dataset (manifold) and preserving large distances that PCA would incorrectly preserve the structure of the data. Figure2.5 — Swiss Roll Dataset. Preserve small distance with t-SNE (solid line) vs maximizing variance PCA [1] Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Overview of Advance Excel Advanced Excel refers to features and functions of Microsoft Excel tool which helps the user to perform complex and large calculations, data processing on the huge amount of data, performing data analysis, better representation of data, etc. Advanced Excel functions allow business organisations to increase their productivity and performance by easily sorting and filtering relevant information and using it for better decision making. Some of the most important advanced Excel functions include date and time function, lookup and reference function, statistical and logical function, data validation and text function, pivot charts and tables, multidimensional analysis, advanced dashboard, etc.

1. Data Validation in Excel Data validation in Excel is a technique that restricts user input in a worksheet. It is often used to limit user entry. Figure: 2.5 Data Validation in Excel Settings Tab The settings tab is where you enter the validation criteria. There are eight options available to validate for user input:

- Any Value - It removes any existing data validation.
- Whole Number - It allows only whole numbers. For example, you can specify that the user must enter the number between 0 to 30.
- Decimal - The user must enter a number with decimal values.
- List - The user will have to create a drop-down list to choose from.
- Date - The user will have to enter the date format.
- Time - The user should enter a time.
- Text Length - It validates input based on the length of the data.
- Custom - It validates the user input using a custom formula.

2. Introduction to charts In Microsoft Excel, charts are used to make a graphical representation of any set of data. A chart is a visual representation of data, in which the data is represented by symbols such as bars in a bar chart or lines in a line chart. Charts Group You can find the Charts group under the INSERT tab on the Ribbon. Figure: 2.6 Charts Group The Charts group on the Ribbon looks as follows – Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science The Charts group is formatted in such a way that –

- Types of charts are displayed.
- The subgroups are clubbed together.
- It helps you find a chart suitable to your data with the button Recommended Charts.

Chart Tools When you click on a chart, a new tab Chart Tools is displayed on the ribbon. There are two tabs under CHART TOOLS –

- DESIGN
- FORMAT

3. Pivot Table A PivotTable is an interactive way to quickly summarize large amounts of data. You can use a PivotTable to analyze numerical data in detail, and answer unanticipated questions about your data. A PivotTable is especially designed for:

- Querying large amounts of data in many user-friendly ways.
- Subtotaling and aggregating numeric data, summarizing data by categories and subcategories, and creating custom calculations and formulas.
- Expanding and collapsing levels of data to focus your results, and drilling down to details from the summary data for areas of interest to you.
- Moving rows to columns or columns to rows (or "pivoting") to see different summaries of the source data.
- Filtering, sorting, grouping, and conditionally formatting the most useful and interesting subset of data enabling you to focus on just the information you want.
- Presenting concise, attractive, and annotated online or printed reports.

For example, here's a simple list of household expenses on the left, and a PivotTable based on the list to the right: Sales data Corresponding PivotTable Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Figure: 2.7 PivotTable 4. Scenario Manager in Excel Excel's scenario manager is a collection of digital tools that allow a user to create, analyze and compare data results in different business situations. You can store multiple versions of data within the same cell, then change them depending on a scenario's goal. If you have several potential data sets, this feature can help determine the final value for a metric. For example, a company may create a budget based on a high- or low-revenue scenario, then compare results to understand how projected revenue may influence budgeting plans. Some other features of the scenario manager include saving various groups of values, merging different scenarios together and generating summaries of each situation. Using this tool can help a business understand and predict important information about a decision-making process. It can also help stakeholders collaborate effectively on a databased project.

How to use scenario manager in Excel

1. Locate the scenario manager
2. Enter data into the spreadsheet
3. Create a scenario
4. Add another scenario
5. Merge scenarios
6. Create a scenario summary report
5. Protecting data To prevent others from accessing data in your Excel files, protect your Excel file with a password.
- Select File > Info.
- Select the Protect Workbook box and choose Encrypt with Password.
- Enter a password in the Password box, and then select OK.
- Confirm the password in the

Reenter Password box, and then select OK. ♣ Microsoft cannot retrieve forgotten passwords, so be sure that your password is especially memorable. ♣ There are no restrictions on the passwords you use with regards to length, characters or numbers, but passwords are case-sensitive. ♣ It's not always secure to distribute password-protected files that contain sensitive information such as credit card numbers. ♣ Be cautious when sharing files or passwords with other users. You still run the risk of passwords them falling into the hands of unintended users. Remember that locking a file with a password does not necessarily protect your file from malicious intent.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science

6. Excel Minor Major unit the numeric interval at which lines will be placed across the chart and at which numbers will be displayed along the Y axis. Minor unit the second numeric interval and determines where little tick-marks will be placed along the Y axis. This should be a value smaller than the Major unit.

Figure: 2.8 Excel minor 7. Macros in Excel Macros are a set of programming instructions written in VBA, which automate a repeated and standardized process in Excel and other Microsoft Office Applications, like Access, PowerPoint, Word & Outlook. An Excel macro is an action or a set of actions that can be recorded, named, saved and executed as many times as required and whenever desired. By using macros, we are able to automate repetitive tasks associated with data manipulation and data reporting that must be accomplished repeatedly. Record a Macro Here are the steps to record this macro:

1. Click the Developer tab.
2. In the Code group, click on the Macro button. This will open the 'Record Macro' dialog box.
3. In the Record Macro dialog box, enter a name for your macro. I am using the name EnterText. There are some naming conditions that you need to follow when naming a macro. For example, you can not use spaces in between. I usually prefer to keep my macro names as a single word, with different parts with a capitalized first alphabet. You can also use underscore to separate two words – such as Enter\_Text.
4. (Optional Step) You can assign a keyboard shortcut if you want. In this case, we will use the shortcut Control + Shift + N. Remember that the shortcut you assign here would override any existing shorcuts in your workbook. For example, if you assign the shortcut Control + S, you will not be able to use this for saving the workbook (instead, everytime you use it, it will execute the macro).
5. In the 'Store macro in' option, make sure 'This Workbook' is selected. This step ensures that the macro is a part of the workbook. It will be there when you save it and reopen again, or even if you share it with someone.
6. (Optional Step) Enter a description. I usually don't do this, but if you're extremely organized, you may want to add what the macro is about.
7. Click OK. As soon as you click OK, it starts to record your actions in Excel. You can see the 'Stop recording' button in the Developer tab, which indicates that the macro recording is in progress.
8. Select cell A2.
9. Enter the text Excel (or you can use your name).
10. Hit the Enter key. This will select cell A3.
11. Click on the Stop Recording button the Developer tab.

Chameli Devi Group of Institutions, Indore Department of Artificial Intelligence & Data Science Subject Notes AD403: Software Engineering with Agile Methodology B. Tech AD-4 th Semester Unit III Need and Types of Maintenance, Software Configuration Management (SCM), Software Change Management, Version Control, Change control and Reporting, Program Comprehension Techniques, Re-engineering, Reverse Engineering, Tool Support. Project Management Concepts, Feasibility Analysis, Project and Process Planning, Resources Allocations, Software efforts, Schedule, and Cost estimations, Project Scheduling and Tracking, Risk Assessment and Mitigation, Software Quality Assurance (SQA). Project Plan, Project Metrics. Course Objective: To Introduce Measures and Metrics for Software quality, reliability and software estimation Techniques Course Outcome: Understand Various Measures of software and generate project schedule Software Maintenance Software maintenance is a part of the Software Development Life Cycle. Its primary goal is to modify and update software application after delivery to correct errors and to improve performance. Software Maintenance is an inclusive activity that includes error corrections, enhancement of capabilities, deletion of obsolete capabilities, and optimization. Need for Maintenance Software Maintenance is needed for:  
1. Correct errors  
2. Change in user requirement with time  
3. Changing hardware/software requirements  
4. To improve system efficiency  
5. optimize the code to run faster  
6. To modify the components  
7. To reduce any unwanted side effects.

Types of Maintenance

1. Corrective Maintenance
- Corrective maintenance aims to correct any remaining errors regardless of where they may cause specifications, design, coding, testing, and documentation, etc.
2. Adaptive Maintenance It contains modifying the software to match changes in the ever-changing environment.
3. Preventive Maintenance It is the process by which we prevent our system from being obsolete. It involves the concept of reengineering & reverse engineering in which an old system with old technology is re

engineered using new technology. This maintenance prevents the system from dying out.

4. Perfective Maintenance It defines improving processing efficiency or performance or restricting the software to enhance changeability. This may contain enhancement of existing system functionality, improvement in computational efficiency, etc.

Figure 3.1: Maintenance Process

**SOFTWARE CONFIGURATION MANAGEMENT (SCM):** Software configuration management is a set of activities carried out for identifying, organizing and controlling Software Configuration Items (SCIs): Information that is created as part of the software engineering process.

**Baselines:** A Baseline is a software configuration management concept that helps us to control change. Signals a point of departure from one activity to the start of another activity. Helps control change without impeding justifiable change.

**Elements of SCM** There are four elements of SCM:

1. Software Configuration Identification
2. Software Configuration Control
3. Software Configuration Auditing
4. Software Configuration Status Reporting

**SOFTWARE CHANGE MANAGEMENT** Change control is function of configuration management, which ensures that all changes made to software system are consistent and made as per organizational rules and regulations. A change in the configuration of product goes through following steps –

1. Identification: A change request arrives from either internal or external source. When change request is identified formally, it is properly documented.
2. Validation: Validity of the change request is checked and its handling procedure is confirmed.
3. Analysis: The impact of change request is analyzed in terms of schedule, cost and required efforts. Overall impact of the prospective change on system is analyzed.
4. Control: It is decided that the changes are worth incorporation or not. If it is not, change request is refused formally.
5. Execution: If the previous phase determines to execute the change request, this phase takes appropriate actions to execute the changes, through a thorough revision if necessary.
6. Close request: The change is verified for correct implementation and merging with the rest of the system. This newly incorporated change in the software is documented properly and the request is formally closed.

**VERSION CONTROL:** Version Control is a system or tool that captures the changes to a source code element: file, folder, image or binary. This is beneficial for many reasons, but the most fundamental reason is it allows you to track changes on a per file basis.

**Version Control Benefits:**

1. Secure Access to your Source Code
2. File History
3. Facilitate Team Communication
4. Baseline Trace Ability
5. Automated Merge Capabilities
6. Ensures no one Over-Writes Someone Else's Code
7. Allows for Better Control for Parallel Development

**CHANGE CONTROL AND REPORTING:** Change control is a systematic approach to managing all changes made to a product or system. The purpose is to ensure that no unnecessary changes are made, that all changes are documented, that services are not unnecessarily disrupted and that resources are used efficiently. Change control is an essential step in software life cycle.

**Process of Change Management**

1. Creating a request for change
2. Reviewing and assessing a request for change
3. Planning the change
4. Testing the change
5. Creating a change proposal
6. Implementing changes
7. Reviewing change performance
8. Closing the process

**PROGRAM COMPREHENSION TECHNIQUES** Program comprehension is a domain of computer science concerned with the ways software engineers maintain existing source code. Program Comprehension is useful for reuse, maintenance, reverse engineering and many other activities in the context of Software Engineering.

Figure 3.2: PROGRAM COMPREHENSION RE-ENGINEERING:

Software re-engineering means re-structuring or re-writing part or all of the software engineering system. It is needed for the application which requires frequent maintenance. Software reengineering is a process of software development which is done to improve the maintainability of a software system.

Re-engineering a software system has two key advantages:

1. Reduced risk: As the software already exists, the risk is less as compared to developing new software.
2. Reduced cost: The cost of re-engineering is significantly less than the costs of developing new software.

**Re-engineering process activities:**

1. Source code translation: In this phase code is converted into new language.
2. Reverse Engineering: Under this activity the program is analyzed and understood thoroughly.
3. Program structure improvement: Restructure automatically for understandability.
4. Program modularization: The program structure is reorganized.
5. Data re-engineering: Finally clean-up and restructure system data

Figure 3.3: Re-engineering Activities

**REVERSE ENGINEERING:** Reverse engineering is the process of design recovery. In reverse engineering the data, architectural and procedural information is extracted from a source code. The reverse engineering is required because using this technique the dirty, ambiguous code can be converted to clear and unambiguous specification. This specification helps in understanding the source code.

There are 3 important issues in reverse engineering:

1. Abstraction Level: This level helps in obtaining the design information from the source code. It is expected that abstraction level should be high in reverse engineering.
2. Completeness Level: The completeness means detailing of abstract level. The completeness decreases as abstraction level increases.
3. Directionality Level: Directionality means extracting the information from source code and gives it to

software engineer. The directionality can be one way or two-way. The one-way directionality means extracting all the information from source code and gives it to software engineer. The two-way directionality means the information taken from source code is fed to reengineering tool that attempts to restructure or regenerate old programs.

**Figure 3.4: Reverse Engineering TOOL SUPPORT:** CASE Tool Support: CASE tools are set of software application programs, which are used to automate SDLC activities. CASE tools are used by software project managers, analysts and engineers to develop software system. There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools are to name a few. Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development. Components of CASE Tools CASE tools can be broadly divided into the following parts based on their use at a particular SDLC stage: 1. Central Repository - CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management are stored. Central repository also serves as data dictionary. 2. Upper Case Tools - Upper CASE tools are used in planning, analysis and design stages of SDLC. 3. Lower Case Tools - Lower CASE tools are used in implementation, testing and maintenance. 4. Integrated Case Tools - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.

**Figure 3.5: Case Component PROJECT MANAGEMENT CONCEPTS:** Software project management is an activity of organizing, planning and scheduling the software projects. The goal of software project management is to deliver the software product in given time and within the budget. It is also necessary that the software project should be developed in accordance with the requirements of the organization. The project management is the application of knowledge, skill, tools and techniques to project activities to meet the project requirements. Objectives of project management: 1. The objective of the project planning and management is to provide a framework for the project. 2. Using the project framework, the project manager decides the estimates for the schedule, cost and resources. 3. Another objective of the project planning and management is that- it should be possible to get the best case and worst-case outcomes of the project. 4. There should be sufficient information discovery through the project so that reasonable project estimate can be made.

**Feasibility Analysis:** When the client approaches the organization for getting the desired product developed, it comes up with a rough idea about what all functions of the software must perform and which all features are expected from the software. This feasibility study is focused towards goal of the organization. This study analyses whether the software product can be practically materialized in terms of implementation, contribution of project to organization, cost constraints, and as per values and objectives of the organization. It explores technical aspects of the project and product such as usability, maintainability, productivity, and integration ability. The output of this phase should be a feasibility study report that should contain adequate comments and recommendations for management about whether or not the project should be undertaken.

**PROJECT AND PROCESS PLANNING:** Project planning is part of project management, in which project manager should recognize the future problems in advance and should be ready with the tentative solution to those problems. A project plan must be prepared in advance from the available information. The project planning is an iterative process and it gets completed only on the completion of the project. This process is iterative because new information gets available at each phase of the project development. Hence the plan needs to be modified on regular basis for accommodation new requirements of the project. The main purpose of this phase is to plan time, cost, and resources adequately to estimate the work needed and to effectively manage risk. Initial planning generally consists of:

- Developing the scope statement
- Selecting the planning team
- Identifying deliverables
- Creating the work breakdown structure
- Identifying the activities needed to complete those deliverables
- Sequencing the activities in a logical way
- Estimating the resources needed
- Estimating the time needed
- Estimating the costs
- Developing the schedule
- Developing the budget
- Gaining formal approval to begin

**RESOURCE ALLOCATIONS:** Once the objectives of the project management are achieved, the project management is to estimate the resources for the project. Various recourses of the project are:

- Human or people
- Reusable software components
- Hardware or software component

The resources are available in limited quantity and stay in the organization as a pool of assets. The shortage of resources hampers development of the project and it can lag behind the schedule. Allocating extra resources increases development cost in the end. It is therefore necessary to estimate and allocate adequate resources for the project. Resource management includes:

1. Defining proper organization project by creating a project team and allocating responsibilities to each team member.
2. Determining resources required

at a particular stage and their availability.

3. Manage Resources by generating resource request when they are required and deallocating them when they are no more needed.

**SOFTWARE EFFORTS:** Project Estimation For an effective management, accurate estimation of various measures is a must. With the correct estimation, managers can manage and control the project more efficiently and effectively. Project estimation may involve the following:

- Software size estimation: Software size may be estimated either in terms of KLOC (Kilo Line of Code) or by calculating number of function points in the software. Lines of code depend upon coding practices. Function points vary according to the user or software requirement.
- Effort estimation: The manager estimates efforts in terms of personnel requirement and manhour required to produce the software. For effort estimation software size should be known.
- Time estimation: Once size and efforts are estimated, the time required to produce the software can be estimated. An effort required is segregated into sub categories as per the requirement specifications and interdependency of various components of software. Software tasks are divided into smaller tasks, activities or events by Work Breakthrough Structure (WBS). The tasks are scheduled on day-to-day basis or in calendar months. The sum of time required to complete all tasks in hours or days is the total time invested to complete the project.
- Cost estimation: This might be considered as the most difficult of all because it depends on more elements than any of the previous ones. For estimating project cost, it is required to consider –

- o Size of the software
- o Software quality of Hardware
- o Additional software or tools, licenses etc.
- o Skilled personnel with task-specific skills
- o Travel involved
- o Communication
- o Training and support

**PROJECT SCHEDULING:** Project Scheduling in a project refers to roadmap of all activities to be done with specified order and within time slot allotted to each activity. Project managers tend to define various tasks and project milestones and then arrange them keeping various factors in mind. They look for tasks like in critical path in the schedule, which are necessary to complete in specific manner (because of task interdependency) and strictly within the time allocated. During the project scheduling the total work is separated into various small activities.

Figure 3.6: Project Scheduling Process For scheduling a project, it is necessary to:

- Break down the project tasks into smaller, manageable form
- Find out various tasks and correlate them
- Estimate time frame required for each task
- Divide time into work-units
- Assign adequate number of work-units for each task
- Calculate total time required for the project from start to finish

**COST ESTIMATIONS:** Cost estimation can be defined as the approximate judgments of the costs for project. Cost estimation is usually measured in terms of effort. The effort is the amount of time for one person to work for a certain period of time.

COCOMO is one the most widely used software estimation models in the world. The Constructive Cost Model (COCOMO) is a procedural software cost estimation model. COCOMO is used to estimate size, effort and duration based on the cost of the software. COCOMO predicts the effort and schedule for a software product development based on inputs relating to the size of the software and a number of cost drivers that affect productivity.

COCOMO has three different models that reflect the complexities:

- Basic Model:** this model would be applied early in a project's development. It will provide a rough estimate early on that should be refined later on with one of the other models.
- Intermediate Model:** this model would be used after you have more detailed requirements for a project.
- Detailed Model:** when design of the project is complete you can apply this model to further refine your estimate.

Within each of these models there are also three different modes. The mode you choose will depend on your work environment, and the size and constraints of the project itself.

**Basic Model:** The basic COCOMO model estimates the software development effort using only Lines of Code (LOC). Various equations in this model are:

$$\text{Effort Applied (E)} = ab(\text{KLOC})^{bb} \quad [\text{man-months}]$$
$$\text{Development Time (D)} = cb(\text{Effort Applied})^{db} \quad [months]$$
$$\text{People required (P)} = \frac{\text{Effort Applied}}{\text{Development Time}} \quad [\text{count}]$$

Where, KLOC is the estimated number of delivered lines (expressed in thousands) of code for project. The coefficients ab, bb, cb and db are given in the following table:

Table 3.1: List of Constants Based on Mode for Basic COCOMO

Mode	ab	bb	cb	db
Organic	3.2	1.05	1.12	2.8
Semi-detached	3.0	1.20	1.20	3.0
Embedded	2.8	1.20	1.20	2.8

The formula for effort calculation is:

$$E = ai(\text{KLOC})(bi)(EAF)$$

Where E is the effort applied in person-months, KLOC is the estimated number of thousands of delivered lines of code for the project, and EAF is the factor calculated above. The coefficient ai and the exponent bi are given in the next table.

Software Project Ai bi

Project Type	Ai	bi
Organic	3.2	1.05
Semi-detached	3.0	1.20
Embedded	2.8	1.20

Table 3.2: List of Constants Based on Mode for Intermediate Model

Mode	ai	bi	EAF
Organic	3.2	1.05	1.12
Semi-detached	3.0	1.20	1.20
Embedded	2.8	1.20	2.8

The Development time D calculation uses E in the same way as in the Basic COCOMO.

**Detailed Model:** Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process. The detailed model uses different effort multipliers for each cost driver attribute. These Phase Sensitive effort multipliers are each to determine the amount of effort required to complete each phase.

In detailed COCOMO, the whole software is divided into different modules and then we apply COCOMO in different

modules to estimate effort and then sum the effort. The effort is calculated as a function of program size and a set of cost drivers are given according to each phase of the software life cycle.

**PROJECT SCHEDULING AND TRACKING:** Project schedule is the most important factor for software project manager. It is the duty of project manager to decide the project schedule and track the schedule. Tracking the schedule means determine the tasks and milestone in the project as it proceeds. Following are the various ways by which tracking of the project can be achieved:

- Conduct periodic meetings.
- Evaluate results of all the project reviews.
- Compare actual start date and scheduled start date of each of the project task.
- Determine if milestones of the project are achieved on scheduled date or not.
- Meet informally to the software practitioners.
- Assess the progress of the project quantitatively.

**RISK ASSESSMENT AND MITIGATION:** Risk management involves all activities pertaining to identification, analyzing and making provision for predictable and non-predictable risks in the project. Risk may include the following:

- Experienced staff leaving the project and new staff coming in.
- Change in organizational management.
- Requirement change or misinterpreting requirement.
- Under-estimation of required time and resources.
- Technological changes, environmental changes, business competition.

**Process of Risk Management:** Risk management performed in following stages:

- Risk identification: In this phase all possible risks are anticipated and a list of potential risks are prepared.
- Risk analysis: After risk identification, a list is prepared in which risks are prioritized.

**Risk planning:** The risk avoidance or risk minimization plan is prepared in this phase.

- Risk monitoring: Identified risks must be mitigated. Hence risk mitigation plan must be prepared once the risks are discovered

**Figure 3.7: Risk Management Process Risk Mitigation, Monitoring and Management (RMMM):** Risk analysis supports the project team in constructing a strategy to deal with risks. There are three important issues considered in developing an effective strategy:

- Risk avoidance or mitigation - It is the primary strategy which is fulfilled through a plan.
- Risk monitoring - The project manager monitors the factors and gives an indication whether the risk is becoming more or less.
- Risk management and planning - It assumes that the mitigation effort failed and the risk is a reality.

**RMMM Plan:**

- It is a part of the software development plan or a separate document.
- The RMMM plan documents all work executed as a part of risk analysis and used by the project manager as a part of the overall project plan.
- The risk mitigation and monitoring starts after the project is started and the documentation of RMMM is completed

**SOFTWARE QUALITY ASSURANCE (SQA):** Software Quality: In the context of software engineering, software quality measures how well software is designed (quality of design), and how well the software conforms to that design (quality of conformance)

**Quality Control:** Quality control (QC) is a procedure or set of procedures intended to ensure that a manufactured product or performed service adheres to a defined set of quality criteria or meets the requirements of the client or customer.

**Quality Assurance:** It is planned and systematic pattern of activities necessary to provide a high degree of confidence in the quality of a product. It provides quality assessment of the quality control activities and determines the validity of the data or procedures for determining quality.

A Quality Management Plan is prepared

- Application of Technical Methods (Employing proper methods and tools for developing software)
- Conduct of Formal Technical Review (FTR)
- Testing of Software
- Enforcement of Standards (Customer imposed standards or management imposed standards)
- Control of Change (Assess the need for change, document the change)
- Measurement (Software Metrics to measure the quality, quantifiable)
- Records Keeping and Recording (Documentation, reviewed, change control etc. i.e. benefits of docs)

**PROJECT PLAN:** A project plan is a formal document designed to guide the control and execution of a project. A project plan is the key to a successful project and is the most important document that needs to be created when starting any business project. A project plan is used for the following purposes:

- To document and communicate stakeholder products and project expectations
- To control schedule and delivery
- To calculate and manage associated risks

**PROJECT METRICS:** Metrics is a quantitative measure of the degree to which a system, component, or process possesses a given attribute. Project metrics are quantitative measures that enable software engineers to gain insight into the efficiency of the software process and the projects conducted using the process framework. Project metrics are used by a project manager and a software team to adapt project work flow and technical activities.

**Size Oriented Metrics:**

- Size oriented measure is derived by considering the size of software that has been produced.
- The organization builds a simple record of size measure for the software projects. It is built on past experiences of organizations.
- It is a direct measure of software
- A simple set of size measure that can be developed is as given below:
  - Size= KLOC
  - Effort = Person/month
  - Productivity=KLOC/ Person-month
  - Quality= number of faults/KLOC
  - Cost=\$/KLOC
  - Documentation= Pages of documentation/KLOC

**Function Oriented Metrics:**

- Use a measure of the functionality delivered by the application as a normalization value.
- Functionality cannot be measured directly; it must be derived indirectly using other direct measures.

A measure called the function point. Function points

are derived using an empirical relationship based on countable (direct) measures of software's information domain and assessments of software complexity.

Chameli Devi Group of Institutions Department of Artificial Intelligence and Data Science Subject Notes AD405 Operating Systems UNIT-II Processes Concept Process: A process is basically a program in execution. The execution of a process must progress in a sequential fashion. A process is defined as an entity which represents the basic unit of work to be implemented in the system. To put it in simple terms, we write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program. When a program is loaded into the memory and it becomes a process, it can be divided into four sections – Stack, Heap, Text and Data. S. No. Component & Description 1 Stack - The process Stack contains the temporary data such as method/function parameters, return address and local variables. 2 Heap - This is dynamically allocated memory to a process during its run time. 3 Text - This includes the current activity represented by the value of Program Counter and the contents of the processor's registers. 4 Data - This section contains the global and static variables. Process Life Cycle & Process State diagram When a process executes, it passes through different states. These stages may differ in different operating systems, and the names of these states are also not standardized. In general, a process can have one of the following five states at a time. S. No. State & Description 1 Start - This is the initial state when a process is first started/ created. 2 Ready - The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run. Process may come into this state after Start state or while running it by but interrupted by the scheduler to assign CPU to some other process. 3 Running - Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions. 4 Waiting - Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available. 5 Terminated or Exit - Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory. Figure: Process State Diagram Process Based Kernel A kernel process is a process that is created in the kernel protection domain and always executes in the kernel protection domain. Kernel processes can be used in subsystems, by complex device drivers, and by system calls. Example- The kernel grants device access through system calls. For example, a user process might call the open(), read(), or write() system calls to manipulate a file within the file system. A process could call the fork() system call to create a new child process Dual mode of process execution The dual-mode operations in the operating system protect the operating system from illegal users. We accomplish this defense by designating some of the system instructions as privileged instructions that can cause harm. To ensure proper operating system execution, we must differentiate between machine code execution and user-defined code. We have two modes of the operating system: user mode and kernel mode. 1. User Mode When the computer system runs user applications like file creation or any other application program in the User Mode, this mode does not have direct access to the computer's hardware. For performing hardware related tasks, like when the user application requests for a service from the operating system or some interrupt occurs, in these cases, the system must switch to the Kernel Mode. The mode bit of the user mode is 1. This means that if the mode bit of the system's processor is 1, then the system will be in the User Mode. 2. Kernel Mode All the bottom level tasks of the Operating system are performed in the Kernel Mode. As the Kernel space has direct access to the hardware of the system, so the kernel-mode handles all the processes which require hardware support. Apart from this, the main functionality of the Kernel Mode is to execute privileged instructions. Example: With the mode bit, we can distinguish between a task executed on behalf of the operating system and one executed on behalf of the user. CPU Scheduling Concepts Definition: The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy. Process Scheduling Queues The OS maintains all PCBs in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue. When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue. The Operating System maintains the following important process scheduling queues – • Job queue– this queue keeps all the processes in the system. • Ready queue– this queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue. • Device queues– the processes which are blocked due to unavailability of an I/O device

constitute this queue. Figure: Process Scheduling Queues Types of Schedulers Schedulers are special system software which handles process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types – Long-Term Scheduler, Short-Term Scheduler and Medium-Term Scheduler

**Long Term Scheduler:** It is also called a job scheduler. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling. The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system. On some systems, the long-term scheduler may not be available or minimal. Time-sharing operating systems have no long-term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

**Short Term Scheduler:** It is also called as CPU scheduler. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them. Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

**Medium Term Scheduler:** Medium-term scheduling is a part of swapping. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is incharge of handling the swapped out-processes. A running process may become suspended if it makes an I/O request. Suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called swapping, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix

**Context Switch** A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time. Using this technique, a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system features. When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block. After this, the state for the process to run next is loaded from its own PCB and used to set the PC, registers, etc. At that point, the second process can start executing.

**Scheduling Algorithms**

A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms. These algorithms are either non-preemptive or preemptive. Non-preemptive algorithms are designed so that once a process enters the running state; it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

**First Come First Serve (FCFS)**

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.

**Shortest Job Next (SJN)**

- This is also known as shortest job first, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processes should know in advance how much time process will take.

**Priority Based Scheduling**

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

**Shortest Remaining Time (SRT)**

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

**Round Robin Scheduling**

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a quantum.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.

**Algorithm Evaluation** How do we select a CPU scheduling algorithm for a particular system? There are many scheduling

algorithms, each with its own parameters. As a result, selecting an algorithm can be difficult. The first problem is defining the criteria to be used in selecting an algorithm. Criteria are often defined in terms of CPU utilization, response time, or throughput. To select an algorithm, we must first define the relative importance of these measures.

**Deterministic Modeling** This method takes a particular predetermined workload and defines the performance of each algorithm for that workload. For example, assume that we have the workload shown below. All five processes arrive at time 0, in the order given, with the length of the CPU burst given in milliseconds.

**Queuing Models:** If we define a queue for the CPU and a queue for each I/O device we can test the various scheduling algorithms using queuing theory. One useful formula is little's Formula.  $n = \lambda w$  Where  $n$  is the average queue length,  $\lambda$  is the average arrival rate for new processes (e.g. five a second) and  $w$  is the average waiting time in the queue.

**Simulations:** Rather than using queuing models we simulate a computer. A variable, representing a clock is incremented. At each increment the state of the simulation is updated. Statistics are gathered at each clock tick so that the system performance can be analyzed. The data to drive the simulation can be generated in the same way as the queuing model, although this leads to similar problems.

**Implementation:** The best way to compare algorithms is to implement them on real machines. This will give the best results but does have a number of disadvantages.

- It is expensive as the algorithm has to be written and then implemented on real hardware.
- If typical workloads are to be monitored, the scheduling algorithm must be used in a live situation. Users may not be happy with an environment that is constantly changing.
- If we find a scheduling algorithm that performs well there is no guarantee that this state will continue if the workload or environment changes.

**System Calls for Process Management**

Basic process management is done with a number of system calls, each with a single (simple) purpose. These system calls can then be combined to implement more complex behaviors.

**Figure: System Calls for Process Management**

The following system calls are used for basic process management:

- **fork:** A parent process uses fork to create a new child process. The child process is a copy of the parent. After fork, both parent and child executes the same program but in separate processes.
- **exec:** Replaces the program executed by a process. The child may use exec after a fork to replace the process' memory space with a new program executable making the child execute a different program than the parent.
- **exit:** Terminates the process with an exit status.
- **wait:** The parent may use wait to suspend execution until a child terminates. Using wait the parent can obtain the exit status of a terminated child.

**Concept of Threads:**

- A thread is a flow of execution through the process code, with its own program counter.
- A thread shares with its peer threads little information like code segment, data segment and open files.
- A thread is also called a lightweight process. Threads provide a way to improve application performance through parallelism.
- Each thread belongs to exactly one process and no thread can exist outside a process.

**Types of Thread**

Threads are implemented in following two ways –

- **User Level Threads–** User managed threads.
- **Kernel Level Threads–** Operating System managed threads acting on kernel, an operating system core.

**User Level Threads:** In this case, the thread management kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application starts with a single thread.

**Kernel Level Threads:** In this case, thread management is done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.

**Multithreading Models**

Some operating system provides a combined user level thread and Kernel level thread facility. Solaris is a good example of this combined approach. In a combined system, multiple threads within the same application can run in parallel on multiple processors and a blocking system call need not block the entire process.

Multithreading models are three types

- Many too many relationships.
- Many to one relationship.
- One to one relationship.

**Many too Many Models**

The many-to-many model multiplexes any number of user threads onto an equal or smaller number of kernel threads.

**Figure: Many to Many Model**

**Many to One Model**

Many-to-one model maps many user level threads to one Kernel-level thread.

**Figure: Many to One Model**

**One to One Model**

There is one-to-one relationship of user-level thread to the kernel-level thread. OS/2, Windows NT and windows 2000 use one to one relationship model.

**Figure: One to One Model**

**Process management in unix**

Whenever you issue a command in Unix, it creates, or starts, a new process. When you tried out the ls command to list the directory contents, you started a process. A process, in simple terms, is an instance of a running program. Each process in the system has a unique pid. Pids eventually repeat because all the possible numbers are used up and the next pid rolls or starts over. At any point of time, no two processes with the same pid exist in the system because it is the pid that Unix uses to track each process.

**Starting a Process:**

When you start a process, the operating system allocates resources to it, such as memory and CPU time. The process then begins to execute its instructions. The operating system keeps track of the process's state and manages its execution. When the process finishes executing, it ends and the operating system releases its resources back to the system.

process (run a command), there are two ways you can run it –

- Foreground Processes
- Background Processes

**Foreground Processes** By default, every process that you start runs in the foreground. It gets its input from the keyboard and sends its output to the screen. You can see this happen with the ls command. If you wish to list all the files in your current directory, You can use the following command – \$ls ch\*.doc This would display all the files, the names of which start with ch and end with .doc – ch01-1.doc ch010.doc ch02.doc ch03-2.doc ch04-1.doc ch040.doc ch05.doc ch06-2.doc ch01-2.doc ch02-1.doc The process runs in the foreground, the output is directed to my screen, and if the ls command wants any input (which it does not), it waits for it from the keyboard. While a program is running in the foreground and is time-consuming, no other commands can be run (start any other processes) because the prompt would not be available until the program finishes processing and comes out.

**Background Processes:** A background process runs without being connected to your keyboard. If the background process requires any keyboard input, it waits. The advantage of running a process in the background is that you can run other commands; you do not have to wait until it completes to start another! The simplest way to start a background process is to add an ampersand (&) at the end of the command. This displays all those files the names of which start with ch and end with .doc – ch01-1.doc ch010.doc ch02.doc ch03-2.doc ch04-1.doc ch040.doc ch05.doc ch06-2.doc ch01-2.doc ch02-1.doc Here, if the ls command wants any input (which it does not), it goes into a stop state until we move it into the foreground and give it the data from the keyboard. That first line contains information about the background process - the job number and the process ID. You need to know the job number to manipulate it between the background and the foreground. Press the Enter key and you will see the following – [1] + Done ls ch\*.doc & \$ The first line tells you that the ls command background process finishes successfully. The second is a prompt for another command.

**Inter Process Communication** A process can be of two types:

- Independent process.
- Co-operating process.

An independent process is not affected by the execution of other processes while a cooperating process can be affected by other executing processes. Though one can think that those processes, which are running independently, will execute very efficiently but in practical, there are many situations when co-operative nature can be utilized for increasing computational speed, convenience and modularity. Inter process communication (IPC) is a mechanism which allows processes to communicate each other and synchronize their actions. The communication between these processes can be seen as a method of co-operation between them. Processes can communicate with each other using these two ways:

1. Shared Memory
2. Message passing

The Figure below shows a basic structure of communication between processes via shared memory method and via message passing. An operating system can implement both method of communication. First, we will discuss the shared memory method of communication and then message passing. Communication between processes using shared memory requires processes to share some variable and it completely depends on how programmer will implement it. One way of communication using shared memory can be imagined like this: Suppose process1 and process2 are executing simultaneously and they share some resources or use some information from other process, process1 generate information about certain computations or resources being used and keeps it as a record in shared memory. When process2 need to use the shared information, it will check in the record stored in shared memory and take note of the information generated by process1 and act accordingly. Processes can use shared memory for extracting information as a record from other process as well as for delivering any specific information to other process. Let's discuss an example of communication between processes using shared memory method.

**Process Synchronization** Process Synchronization means sharing system resources by processes in such a way that, Concurrent access to shared data is handled thereby minimizing the chance of inconsistent data. Maintaining data consistency demands mechanisms to ensure synchronized execution of cooperating processes. Process Synchronization was introduced to handle problems that arose while multiple process executions. Some of the problems are discussed below.

**Real and virtual Concurrency** Distributed processing involves multiple processes on multiple systems. All of these involve cooperation, competition, and communication between processes that either run simultaneously or are interleaved in arbitrary ways to give the appearance of running simultaneously. Concurrent processing is thus central to operating systems and their design. Principles and Problems in Concurrency Concurrency is the interleaving of processes in time to give the appearance of simultaneous execution. Thus it differs from parallelism, which offers genuine simultaneous execution. However the issues and difficulties raised by the two overlap to a large extent:

- Sharing global resources safely is difficult
- Optimal allocation of resources is difficult
- Locating programming errors can be difficult, because the contexts in which errors occur cannot always be reproduced easily. Parallelism also introduces the issue that different processors may run at different speeds, but again this problem is mirrored in concurrency because different processes progress at

different rates. A Simple Example The fundamental problem in concurrency is processes interfering with each other while accessing a shared global resource. This can be illustrated with a surprisingly simple example: `chin = getchar(); chout = chin; putchar(chout);` Imagine two processes P1 and P2 both executing this code at the “same” time, with the following interleaving due to multi-programming.

- P1 enters this code, but is interrupted after reading the character x into chin.
- P2 enters this code, and runs it to completion, reading and displaying the character y.
- P1 is resumed, but chin now contains the character y, so P1 displays the wrong character.

The essence of the problem is the shared global variable chin. P1 sets chin, but this write is subsequently lost during the execution of P2. The general solution is to allow only one process at a time to enter the code that accesses chin: such code is often called a critical section. When one process is inside a critical section of code, other processes must be prevented from entering that section. This requirement is known as mutual exclusion. Mutual Exclusion: Mutual exclusion is in many ways the fundamental issue in concurrency. It is the requirement that when a process P is accessing a shared resource R, no other process should be able to access R until P has finished with R. Examples of such resources includes files, I/O devices such as printers, and shared data structures.

There are essentially three approaches to implementing mutual exclusion.

- Leave the responsibility with the processes themselves: this is the basis of most software approaches. These approaches are usually highly error-prone and carry high overheads.
- Allow access to shared resources only through special-purpose machine instructions: i.e. a hardware approach. These approaches are faster but still do not offer a complete solution to the problem, e.g. they cannot guarantee the absence of deadlock and starvation.
- Provide support through the operating system, or through the programming language. We shall outline three approaches in this category: semaphores, monitors, and message passing.

Synchronization Synchronization is the coordination of execution of multiple processes in a multi-process system to ensure that they access shared resources in a controlled and predictable manner. It aims to resolve the problem of race conditions and other synchronization issues in a concurrent system. The main objective of process synchronization is to ensure that multiple processes access shared resources without interfering with each other, and to prevent the possibility of inconsistent data due to concurrent access. To achieve this, various synchronization techniques such as semaphores, monitors, and critical sections are used. In a multi-process system, synchronization is necessary to ensure data consistency and integrity, and to avoid the risk of deadlocks and other synchronization problems. Process synchronization is an important aspect of modern operating systems, and it plays a crucial role in ensuring the correct and efficient functioning of multi-process systems. On the basis of synchronization, processes are categorized as one of the following two types:

- Independent Process: The execution of one process does not affect the execution of other processes.
- Cooperative Process: A process that can affect or be affected by other processes executing in the system.

Race Condition: When more than one process is executing the same code or accessing the same memory or any shared variable in that condition there is a possibility that the output or the value of the shared variable is wrong so for that all the processes doing the race to say that my output is correct this condition known as a race condition

Critical Section Problem A Critical Section is a code segment that accesses shared variables and has to be executed as an atomic action. It means that in a group of cooperating processes, at a given point of time, only one process must be executing its critical section. If any other process also wants to execute its critical section, it must wait until the first one finishes.

Solution to Critical Section Problem A solution to the critical section problem must satisfy the following three conditions:

1. Mutual Exclusion Out of a group of cooperating processes, only one process can be in its critical section at a given point of time.
2. Progress If no process is in its critical section, and if one or more threads want to execute their critical section then any one of these threads must be allowed to get into its critical section.
3. Bounded Waiting After a process makes a request for getting into its critical section, there is a limit for how many other processes can get into their critical section, before this process's request is granted. So after the limit is reached, system must grant the process permission to get into its critical section.

Synchronization Hardware Many systems provide hardware support for critical section code. The critical section problem could be solved easily in a single-processor environment if we could disallow interrupts to occur while a shared variable or resource is being modified. In this manner, we could be sure that the current sequence of instructions would be allowed to execute in order without pre-emption. Unfortunately, this solution is not feasible in a multiprocessor environment. Disabling interrupt on a multiprocessor environment can be time consuming as the message is passed to all the processors. This message transmission lag, delays entry of threads into critical section and the system efficiency decreases.

Mutex Locks As the synchronization hardware solution is not easy to implement for everyone, a strict software approach called Mutex Locks was introduced. In this approach, in the entry section of code, a LOCK is acquired over the

critical resources modified and used inside critical section, and in the exit section that LOCK is released. As the resource is locked while a process executes its critical section hence no other process can access it. Semaphores In 1965, Dijkstra proposed a new and very significant technique for managing concurrent processes by using the value of a simple integer variable to synchronize the progress of interacting processes. This integer variable is called semaphore. So it is basically a synchronizing tool and is accessed only through two low standard atomic operations, Wait and Signal by P(S) and V(S) respectively. In very simple words, semaphore is a variable which can hold only a non-negative Integer value, shared between all the threads, with operations wait and signal, which work as follow: The classical definitions of wait and signal are:

- Wait: Decrements the value of its argument S, as soon as it would become non-negative (greater than or equal to 1).
- Signal: Increments the value of its argument S, as there is no more process blocked on the queue.

Properties of Semaphores

1. It's simple and always has a non-negative Integer value. P(S): if  $S \geq 1$  then  $S := S - 1$  else ; V(S): if then else  $S := S + 1$ ;
2. Works with many processes.
3. Can have many different critical sections with different semaphores.
4. Each critical section has unique access semaphores.
5. Can permit multiple processes into the critical section at once, if desirable.

Types of Semaphores

Semaphores are mainly of two types:

1. Binary Semaphore: It is a special form of semaphore used for implementing mutual exclusion, hence it is often called a Mutex. A binary semaphore is initialized to 1 and only takes the values 0 and 1 during execution of a program.
2. Counting Semaphores: These are used to implement bounded concurrency.

Limitations of Semaphores

1. Priority Inversion is a big limitation of semaphores.
2. Their use is not enforced, but is by convention only.
3. With improper use, a process may block indefinitely. Such a situation is called Deadlock.

Deadlock Every process needs some resources to complete its execution. However, the resource is granted in a sequential order.

1. The process requests for some resource.
2. OS grant the resource if it is available otherwise let the process waits.
3. The process uses it and release on the completion.

Example of Use Here is a simple step wise implementation involving declaration and usage of semaphore.

Deadlock A Deadlock is a situation where each of the computer process waits for a resource which is being assigned to some another process. In this situation, none of the process gets executed since the resource it needs, is held by some other process which is also waiting for some other resource to be released.

Deadlock Characterization

Deadlock can arise if four conditions hold simultaneously:

1. Mutual Exclusion: A resource can only be shared in mutually exclusive manner. It implies, if Shared var mutex: semaphore = 1; Process i begin . . P(mutex); execute CS; V(mutex); . . End; two processes cannot use the same resource at the same time.
2. Hold and Wait: A process waits for some resources while holding another resource at the same time.
3. No preemption: The process which once scheduled will be executed till the completion. No other process can be scheduled by the scheduler meanwhile.
4. Circular Wait: All the processes must be waiting for the resources in a cyclic manner so that the last process is waiting for the resource which is being held by the first process.

Deadlocks Prevention

Deadlocks can be prevented by prevent at least one of the four conditions, because all this four conditions are required simultaneously to cause deadlock.

1. Mutual Exclusion Resources shared such as read-only files do not lead to deadlocks but resources, such as printers and tape drives, requires exclusive access by a single process.
2. Hold and Wait In this condition processes must be prevented from holding one or more resources while simultaneously waiting for one or more others.
3. No Preemption Preemption of process resource allocations can avoid the condition of deadlocks, where ever possible.
4. Circular Wait Circular wait can be avoided if we number all resources, and require that processes request resources only in strictly increasing (or decreasing) order.

Handling Deadlock

The above points focus on preventing deadlocks. But what to do once a deadlock has occurred. Following three strategies can be used to remove deadlock after its occurrence.

1. Preemption We can take a resource from one process and give it to other. This will resolve the deadlock situation, but sometimes it does causes problems.
2. Rollback In situations where deadlock is a real possibility, the system can periodically make a record of the state of each process and when deadlock occurs, roll everything back to the last checkpoint, and restart, but allocating resources differently so that deadlock does not occur.
3. Kill one or more processes This is the simplest way, but it works.

Deadlock Avoidance

- The general idea behind deadlock avoidance is to prevent deadlocks from ever happening, by preventing at least one of the aforementioned conditions.
- This requires more information about each process, AND tends to lead to low device utilization. (it is a conservative approach. )
- In some algorithms the scheduler only needs to know the maximum number of each resource that a process might potentially use. In more complex algorithms the scheduler can also take advantage of the schedule of exactly what resources may be needed in what order.
- When a scheduler sees that starting a process or granting resource requests may lead to future deadlocks, then that process is just not started or the request is not granted.
- A resource allocation state is defined by the number of available and allocated

resources and the maximum requirements of all processes in the system. Safe State • A state is safe if the system can allocate all resources requested by all processes (up to their stated maximums) without entering a deadlock state. • More formally, a state is safe if there exists a safe sequence of processes  $\{P_0, P_1, P_2, \dots, P_N\}$  such that all of the resource requests for  $P_i$  can be granted using the resources currently allocated to  $P_i$  and all processes  $P_j$  where  $j < i$ . (I.e. if all the processes prior to  $P_i$  finish and free up their resources, then  $P_i$  will be able to finish also, using the resources that they have freed up.) • If a safe sequence does not exist, then the system is in an unsafe state, which MAY lead to deadlock. (All safe states are deadlock free, but not all unsafe states lead to deadlocks.) Figure: Safe, unsafe, and deadlocked state spaces For example, consider a system with 12 tape drives, allocated as follows. Is this a safe state? What is the safe sequence? Maximum Needs Current Allocation  $P_0 10 5 P_1 4 2 P_2 9 2$  • What happens to the above table if process  $P_2$  requests and is granted one more tape drive? • Key to the safe state approach is that when a request is made for resources, the request is granted only if the resulting allocation state is a safe one. Resource-Allocation Graph Algorithm • If resource categories have only single instances of their resources, then deadlock states can be detected by cycles in the resource-allocation graphs. • In this case, unsafe states can be recognized and avoided by augmenting the resourceallocation graph with claim edges, noted by dashed lines, which point from a process to a resource that it may request in the future. • In order for this technique to work, all claim edges must be added to the graph for any particular process before that process is allowed to request any resources. (Alternatively, processes may only make requests for resources for which they have already established claim edges, and claim edges cannot be added to any process that is currently holding resources.) • When a process makes a request, the claim edge  $P_i \rightarrow R_j$  is converted to a request edge. Unsafe Safe Dead Similarly when a resource is released, the assignment reverts back to a claim edge. • This approach works by denying requests that would produce cycles in the resourceallocation graph, taking claim edges into effect. • Consider for example what happens when process  $P_2$  requests resource  $R_2$ : Figure: Resource allocation graph for deadlock avoidance • The resulting resource-allocation graph would have a cycle in it, and so the request cannot be granted. Figure: An unsafe state in a resource allocation graph Banker's algorithm • For resource categories that contain more than one instance the resource-allocation graph method does not work, and more complex (and less efficient) methods must be chosen. • The Banker's Algorithm gets its name because it is a method that bankers could use to assure that when they lend out resources they will still be able to satisfy all their clients. (A banker won't loan out a little money to start building a house unless they are assured that they will later be able to loan out the rest of the money to finish the house.) • When a process starts up, it must state in advance the maximum allocation of resources it may request, up to the amount available on the system. • When a request is made, the scheduler determines whether granting the request would leave the system in a safe state. If not, then the process must wait until the request can be granted safely. • The banker's algorithm relies on several key data structures: (where  $n$  is the number of processes and  $m$  is the number of resource categories.) o Available[  $m$  ] indicates how many resources are currently available of each type. o Max[  $n$  ][  $m$  ] indicates the maximum demand of each process of each resource. o Allocation[  $n$  ][  $m$  ] indicates the number of each resource category allocated to each process. o Need[  $n$  ][  $m$  ] indicates the remaining resources needed of each type for each process. ( Note that  $Need[i][j] = Max[i][j] - Allocation[i][j]$  for all  $i, j$ .) • For simplification of discussions, we make the following notations / observations: o One row of the Need vector,  $Need[i]$ , can be treated as a vector corresponding to the needs of process  $i$ , and similarly for Allocation and Max. o A vector  $X$  is considered to be  $\leq$  a vector  $Y$  if  $X[i] \leq Y[i]$  for all  $i$ . Safety Algorithm • In order to apply the Banker's algorithm, we first need an algorithm for determining whether or not a particular state is safe. • This algorithm determines if the current state of a system is safe, according to the following steps: 1. Let Work and Finish be vectors of length  $m$  and  $n$  respectively. ♣ Work is a working copy of the available resources, which will be modified during the analysis. ♣ Finish is a vector of Booleans indicating whether a particular process can finish. ( or has finished so far in the analysis. ) ♣ Initialize Work to Available, and Finish to false for all elements. 2. Find an  $i$  such that both (A)  $Finish[i] == \text{false}$ , and (B)  $Need[i] < Work$ . This process has not finished, but could with the given available working set. If no such  $i$  exists, go to step 4. 3. Set  $Work = Work + Allocation[i]$ , and set  $Finish[i]$  to true. This corresponds to process  $i$  finishing up and releasing its resources back into the work pool. Then loop back to step 2. 4. If  $finish[i] == \text{true}$  for all  $i$ , then the state is a safe state, because a safe sequence has been found. • JTB's Modification: 1. In step 1. instead of making Finish an array of booleans initialized to false, make it an array of ints initialized to 0. Also initialize an int  $s = 0$  as a step counter. 2. In step 2, look for  $Finish[i] == 0$ . 3. In step 3, set

Finish[ i ] to ++s. S is counting the number of finished processes. 4. For step 4, the test can be either Finish[ i ] > 0 for all i, or s >= n. The benefit of this method is that if a safe state exists, then Finish[ ] indicates one safe sequence ( of possibly many. ) Resource-Request Algorithm (The Bankers Algorithm) • Now that we have a tool for determining if a particular state is safe or not, we are now ready to look at the Banker's algorithm itself. • This algorithm determines if a new request is safe, and grants it only if it is safe to do so. • When a request is made (that does not exceed currently available resources), pretend it has been granted, and then see if the resulting state is a safe one. If so, grant the request, and if not, deny the request, as follows: 1. Let Request[ n ][ m ] indicate the number of resources of each type currently requested by processes. If Request[ i ] > Need[ i ] for any process i, raise an error condition. 2. If Request[ i ] > Available for any process i, then that process must wait for resources to become available. Otherwise the process can continue to step 3. 3. Check to see if the request can be granted safely, by pretending it has been granted and then seeing if the resulting state is safe. If so, grant the request, and if not, then the process must wait until its request can be granted safely. The procedure for granting a request ( or pretending to for testing purposes ) is:

- ♣ Available = Available - Request
- ♣ Allocation = Allocation + Request
- ♣ Need = Need - Request

An Illustrative Example • Consider the following situation:

- And now consider what happens if process P1 requests 1 instance of A and 2 instances of C. ( Request[ 1 ] = ( 1, 0, 2 ) )
- What about requests of ( 3, 3, 0 ) by P4? or ( 0, 2, 0 ) by P0? Can these be safely granted? Why or why not?

Recovery from Deadlock There are three basic approaches to recovery from deadlock: 1. Inform the system operator, and allow him/her to take manual intervention. 2. Terminate one or more processes involved in the deadlock 3. Preempt resources. Process Termination • Two basic approaches, both of which recover resources allocated to terminated processes:

- Terminate all processes involved in the deadlock. This definitely solves the deadlock, but at the expense of terminating more processes than would be absolutely necessary.
- Terminate processes one by one until the deadlock is broken. This is more conservative, but requires doing deadlock detection after each step.

• In the latter case there are many factors that can go into deciding which processes to Terminate Next:

1. Process priorities.
2. How long the process has been running, and how close it is to finishing.
3. How many and what type of resources is the process holding. (Are they easy to preempt and restore?)
4. How many more resources does the process need to complete.
5. How many processes will need to be terminated.
6. Whether the process is interactive or batch.
7. (Whether or not the process has made non-restorable changes to any resource.)

Resource Preemption When preempting resources to relieve deadlock, there are three important issues to be addressed:

1. Selecting a victim - Deciding which resources to preempt from which processes involves many of the same decision criteria outlined above.
2. Rollback - Ideally one would like to roll back a preempted process to a safe state prior to the point at which that resource was originally allocated to the process. Unfortunately it can be difficult or impossible to determine what such a safe state is, and so the only safe rollback is to roll back all the way back to the beginning. ( I.e. abort the process and make it start over. )
3. Starvation - How do you guarantee that a process won't starve because its resources are constantly being preempted? One option would be to use a priority system, and increase the priority of a process every time its resources get preempted. Eventually it should get a high enough priority that it won't get preempted any more.

Difference between Starvation and Deadlock Sr. Deadlock Starvation 1 Deadlock is a situation where no process got blocked and no process proceeds Starvation is a situation where the low priority process got blocked and the high priority processes proceed. 2 Deadlock is an infinite waiting. Starvation is a long waiting but not infinite. 3 Every Deadlock is always a starvation. Every starvation need not be deadlock. 4 The requested resource is blocked by the other process. The requested resource is continuously be used by the higher priority processes. 5 Deadlock happens when Mutual exclusion, hold and wait, No preemption and circular wait occurs simultaneously. It occurs due to the uncontrolled priority and resource management. Ipc in unix Pipes are a simple synchronized way of passing information between two processes. A pipe can be viewed as a special file that can store only a limited amount of data and uses a FIFO access scheme to retrieve data. In a logical view of a pipe, data is written to one end and read from the other. The processes on the ends of a pipe have no easy way to identify what process is on the other end of the pipe. The system provides synchronization between the reading and writing process. It also solves the producer/consumer problem: writing to a full pipe automatically blocks, as does reading from an empty pipe. The system also assures that there are processes on both ends of the pipe at all time. The programmer is still responsible, however, for preventing deadlock between processes. Pipes come in two varieties:

- Unnamed. Unnamed pipes can only be used by related processes (i.e. a process and one of its child processes, or two of its children). Unnamed pipes cease to exist after the processes are done using them.
- Named. Named pipes exist as directory entries, complete with permissions. This means that they are

persistent and that unrelated processes can use them. IPC in windows Windows operating system supports various techniques for IPC, these are:

- Clipboard: A loosely coupled data sharing method. When user uses copy or cut command in any application/windows, the copied data is saved in clipboard by windows (temporary storage). The other application can access the data from the clipboard.
- COM: Component Object Model offers a platform to interact in Server and Client pattern between processes. COM server can be a local server or In-Process server. There can be also multiple COM clients which interact with the COM server and exchange data.
- Copy Data: Windows provides a message i.e. WM\_COPYDATA which enables a process to share data with another process. It can be used with SendMessage API of win32 and COPYDATASTRUCT is used as a parameter. This message is used in case of local computer only.
- DDE: Dynamic Data Exchange is a protocol that contains a set of guidelines and rules to send data across processes. A process can use SendMessage API with WM\_DDE\_INITIATE or WM\_DDE\_ACK message sent in response to WM\_DDE\_INITIATE message. It uses shared memory to exchange data.
- File Mapping: File Mapping, a fast communication mechanism between processes and gives an efficient way to use the file content in the virtual memory or by accessing the memory sharing. In this IPC the data of the file is treated as a part of the address space of the process so that process can easily access the address of the content. Any other process with access to the shared memory should implement synchronization to mitigate the risks of data getting corrupted. File Mapping is done on the same system/machine and is not available for network processes.
- Pipes: Pipes can be used as both single and bi-directional data sharing mechanism. Windows supports two types of pipes i.e. pipe and anonymous pipe. Anonymous pipes can be used in the same network or between the related processes only, while named pipe can be used over a network within different processes. Pipe can be considered as a FIFO queue where one end acts as a server and other as the client.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Name: Data Science Subject Code: AD 404 Subject Notes Syllabus: Statistical & Probabilistic analysis of Data, Multiple hypothesis testing, Parameter Estimation methods, Confidence intervals, Correlation & Regression analysis, logistic regression, Shrinkage Methods, Lasso Regression, Bayesian statistics. L1 and L2 regularizations. POWERFUL DATA ANALYSIS—SUMIFS, SUMPRODUCT, VLOOKUP |XLOOKUP, INDEX + MATCH, Handling Formula Errors, Dynamic Array Formulas, Circular References, Formula Auditing, Pivoting.

Course

Objectives:

- The purpose of this subject is to cover the underlying concepts and techniques used in Data Science. Some of these techniques can be used in Data Analysis & in prediction.
- To understand modern way to get insights from the data.

Course

Outcome (CO2): Data visualization techniques and ability to implement data visualization techniques. Unit-3 Statistical & Probabilistic analysis of Data Probability and Statistics form the basis of Data Science. The probability theory is very much helpful for making the prediction. Estimates and predictions form an important part of Data science. With the help of statistical methods, we make estimates for the further analysis. Thus, statistical methods are largely dependent on the theory of probability and all of probability and statistics is dependent on Data. Multiple Hypothesis Testing There is always a minimum of two different hypotheses; Null Hypothesis and Alternative Hypothesis. The hypothesis could be anything, but the most common one is the one I presented below. Null Hypothesis ( $H_0$ ): There is no relationship between the variables Alternative Hypothesis ( $H_1$ ): There is a relationship between variables In the hypothesis testing, we test the hypothesis against our chosen  $\alpha$  level or p-value (often, it is 0.05). If the p-value is significant, we can reject the null hypothesis and claim that the findings support the alternative hypothesis. Often case that we use hypothesis testing to select which features are useful for our prediction model; for example, there are 20 features you are interested in as independent (predictor) features to create your machine learning model. You might think to test each feature using hypothesis testing separately with some level of significance  $\alpha$  0.05. This is feasible and seems like a good idea. However, remember you have 20 hypotheses to test against your target with a significance level of 0.05. What's the probability of one significant result just due to chance? Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science With 20 hypotheses were made, there is around a 64% chance that at least one hypothesis testing result is significant, even if all the tests are actually not significant. With a higher number of features to consider, the chance would even higher. That is why there are methods developed for dealing with multiple testing error. This method is what

we called the multiple testing correction. What was actually corrected? The old way of the correction is by adjusting the  $\alpha$  level in the Family-wise error rate (FWER). Still, there is also a way of correction by controlling the Type I error/False Positive Error or controlling the False Discovery Rate (FDR).

**Parameter Estimation Methods:** Parameter Estimation is a branch of statistics that involves using sample data to estimate the parameters of a distribution.

**Methods of Parameter Estimation** The techniques used for parameter estimation are called estimators. Some estimators are:

- Probability Plotting:** A method of finding parameter values where the data is plotted on special plotting paper and parameters are derived from the visual plot.
- Rank Regression (Least Squares):** A method of finding parameter values that minimizes the sum of the squares of the residuals.
- Maximum Likelihood Estimation:** A method of finding parameter values that, given a set of observations, will maximize the likelihood function.
- Bayesian Estimation Methods:** A family of estimation methods that tries to minimize the posterior expectation of what is called the utility function. In practice, what this means is that existing knowledge about a situation is formulated, data is gathered, and then posterior knowledge is used to update our beliefs.

**Confidence Intervals** A confidence interval is a type of interval calculation in statistics derived from observed data and holds the actual value of an unknown parameter. It's linked to the confidence level, which measures how confident the interval is in estimating the deterministic parameter.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science A confidence interval shows the probability that a parameter will fall between a pair of values around the mean. Confidence intervals show the degree of uncertainty or certainty in a sampling method. They are constructed using confidence levels of 95% or 99%. Figure 3.1 : Confidence Intervals When Do You Use Confidence Intervals? The size of a 90% confidence interval for a given estimate is one method to gauge how "excellent" it is; the greater the range, the more care must be used when utilising the estimate. Confidence intervals serve as a crucial reminder of the estimates' limits. The 95% confidence interval is the range that you can be 95% confident that the similarly constructed intervals will contain the parameter being estimated. The sample mean (center of the CI) will vary from sample to sample because of natural sampling variability. Statisticians use confidence intervals to measure the uncertainty in a sample variable. The confidence is in the method, not in a particular CI. Approximately 95% of the intervals constructed would capture the true population mean if the sampling method was repeated many times.

**Confidence Interval Formula** The formula to find Confidence Interval is:  $X \bar{}$  is the sample mean. Z is the number of standard deviations from the sample mean. S is the standard deviation in the sample.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science n is the size of the sample.

**Correlation & Regression Analysis** Correlation and Regression are popular tools and have been widely used in businesses and research since a long time. The size and application of Big Data in Data Analytics are proving crucial for decision-making. Since the data size and its complex nature is impossible to handle manually, the importance of statistical tools like Correlation and Regression in applying them to business problems has become more valuable. Machine learning and Deep learning algorithms utilize them to provide accurate predictions in fields like computer vision, anomaly detection, etc.

**Basis of Difference Correlation Regression Definition** Correlation is a statistical metric that determines the relationship or association between two variables. Regression indicates how an independent variable may be mathematically connected to any dependent variable.

**Coefficient** The coefficient Correlation ranges from -1 to +1 and thus, a relative measure. The Regression coefficient is generally an absolute value.

**Dependent / Independent variables** Both variables are mutually dependent. The first variable is independent, whereas the second is dependent. Indicates It denotes the extent and manner in which two variables move together.

**Regression** displays the effect of any unit change in the value of the known variable (x) on the value of the estimated variable (y).

**Nature of Coefficient** Mutual and symmetrical correlation coefficients exist. Regression describes one variable as a linear function of another one in case of a linear relationship.

**Objective** To determine the numerical value that specifies the strength and direction of dependence between two variables. To explain the variability in a dependent variable by means of one or more of independent variables in simple or multiple regression respectively.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science

**Basis of Difference Correlation Regression Responding Nature** The Correlation coefficient is designed to be independent of any changes in Scale or Origin. The Regression coefficient is affected by changes in Scale but is unaffected by changes in Origin.

Correlation is a statistical measure used when we want to find out whether there exists a relationship that can link two variables with each other. This linking is beneficial when it is essential to know what is going to be the impact of some chosen parameter on a target to be achieved as to whether it will be positive or negative. The impact of linkage may then be estimated once the strength and direction are known.

**Correlation can be a positive or negative value.**

- Positive Correlation Two variables

are considered to be positively correlated when the value of one variable increases or decreases following an increase or decrease in the value of the other variable respectively.

- Negative Correlation Two variables are considered to be negatively correlated when the value of one variable increases following a decrease in the value of the other variable.
- Zero Correlation This indicates that there is no relationship between two variables. It is also known as a zero correlation.

This is when a change in one variable doesn't affect the other variable in any way. Regression Regression is another vital statistical tool usually supporting the conclusion of Correlation. Once a link of the impact of one variable (input or independent) over the other, usually the target or output, is established, either positive or negative, then Regression plays its role in estimating this impact in the quantitative term. Once developed, this relation can then be used to estimate output when the input variable changes. There are different types of regression and some of them have been listed below:

- Linear Regression
- Logistic Regression
- Ridge Regression
- Lasso Regression

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Logistic Regression Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1. Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems. In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1). The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc. Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets. Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function: Figure 3.2: Logistic Regression Shrinkage Methods Shrinkage methods also known as regularization. Regularization is one of the most important concepts of machine learning. It is a technique to prevent the model from overfitting by adding extra information to it.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Sometimes the machine learning model performs well with the training data but does not perform well with the test data. It means the model is not able to predict the output when deals with unseen data by introducing noise in the output, and hence the model is called overfitted. This problem can be deal with the help of a regularization technique. This technique can be used in such a way that it will allow to maintain all variables or features in the model by reducing the magnitude of the variables. Hence, it maintains accuracy as well as a generalization of the model. It mainly regularizes or reduces the coefficient of features toward zero. In simple words, "In regularization technique, we reduce the magnitude of the features by keeping the same number of features." Techniques of Regularization There are mainly two types of regularization techniques, which are given below:

- Ridge Regression
- Lasso Regression

Ridge Regression • Ridge regression is one of the types of linear regression in which a small amount of bias is introduced so that we can get better long-term predictions.

- Ridge regression is a regularization technique, which is used to reduce the complexity of the model. It is also called as L2 regularization.
- In this technique, the cost function is altered by adding the penalty term to it. The amount of bias added to the model is called Ridge Regression penalty. We can calculate it by multiplying with the lambda to the squared weight of each individual feature. The equation for the cost function in ridge regression will be:

- In the above equation, the penalty term regularizes the coefficients of the model, and hence ridge regression reduces the amplitudes of the coefficients that decreases the complexity of the model.
- As we can see from the above equation, if the values of  $\lambda$  tend to zero, the equation becomes the cost function of the linear regression model. Hence, for the minimum value of  $\lambda$ , the model will resemble the linear regression model.
- A general linear or polynomial regression will fail if there is high collinearity between the independent variables, so to solve such problems, Ridge regression can be used.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science • It helps to solve the problems if we have more parameters than samples.

Lasso Regression: • Lasso regression is another regularization technique to reduce the complexity of the model. It stands for Least Absolute and Selection Operator.

- It is similar to the Ridge Regression except that the penalty term contains only the absolute weights instead of a square of weights.
- Since it takes absolute values, hence, it can shrink the slope to 0, whereas Ridge Regression can only shrink it near to 0.
- It is also

called as L1 regularization. The equation for the cost function of Lasso regression will be:

- Some of the features in this technique are completely neglected for model evaluation.
- Hence, the Lasso regression can help us to reduce the overfitting in the model as well as the feature selection.

Bayesian Statistics is a theory in the field of statistics based on the Bayesian interpretation of probability where probability expresses a degree of belief in an event. The degree of belief may be based on prior knowledge about the event, such as the results of previous experiments, or on personal beliefs about the event. This differs from a number of other interpretations of probability, such as the frequentist interpretation that views probability as the limit of the relative frequency of an event after many trials. Bayesian statistical methods use Bayes theorem to compute and update probabilities after obtaining new data. Bayes' theorem describes the conditional probability of an event based on data as well as prior information or beliefs about the event or conditions related to the event. For example, in Bayesian inference, Bayes' theorem can be used to estimate the parameters of a probability distribution or statistical model. Since Bayesian statistics treats probability as a degree of belief, Bayes' theorem can directly assign a probability distribution that quantifies the belief to the parameter or set of parameters. Bayesian statistics is named after Thomas Bayes, who formulated a specific case of Bayes' theorem in a paper published in 1763. In several papers spanning from the late 18th to the early 19th centuries, Pierre-Simon Laplace developed the Bayesian interpretation of probability. Laplace used methods that would now be considered Bayesian to solve a number of statistical problems. Many Bayesian methods were developed by later authors, but the term was not commonly used to describe such methods until the 1950s. During much of the 20th century, Bayesian methods were viewed unfavorably by many statisticians due to philosophical and Chamel Devi Group of Institutions Department of Artificial Intelligence & Data Science practical considerations. Many Bayesian methods required much computation to complete, and most methods that were widely used during the century were based on the frequentist interpretation. However, with the advent of powerful computers and new algorithms like Markov chain Monte Carlo, Bayesian methods have seen increasing use within statistics in the 21st century. Bayes' theorem is used in Bayesian methods to update probabilities, which are degrees of belief, after obtaining new data. Given two events A and B, the conditional probability of A given that B is true is expressed as follows

L1 and L2 Regularization Methods

A regression model that uses L1 regularization technique is called Lasso Regression and model which uses L2 is called Ridge Regression. The key difference between these two is the penalty term. Ridge regression adds "squared magnitude" of coefficient as penalty term to the loss function. Here the highlighted part represents L2 regularization element. Cost function

Here, if lambda is zero then you can imagine we get back OLS. However, if lambda is very large then it will add too much weight and it will lead to under-fitting. Having said that it's important how lambda is chosen. This technique works very well to avoid over-fitting issue. Lasso Regression (Least Absolute Shrinkage and Selection Operator) adds "absolute value of magnitude" of coefficient as penalty term to the loss function. Cost function

Again, if lambda is zero then we will get back OLS whereas very large value will make coefficients zero hence it will under-fit. The key difference between these techniques is that Lasso shrinks the less important feature's coefficient to zero thus, removing some feature altogether. So, this works well for feature selection in case we have a huge number of features. Chamel Devi Group of Institutions Department of Artificial Intelligence & Data Science

Traditional methods like cross-validation, stepwise regression to handle overfitting and perform feature selection work well with a small set of features but these techniques are a great alternative when we are dealing with a large set of features.

**POWERFUL DATA ANALYSIS: SUMIFS**

The SUMIFS function is a premade function in Excel, which calculates the sum of a range based on one or more true or false condition. It is typed =SUMIFS: =SUMIFS(sum\_range, criteria\_range1, criteria1, [criteria\_range2, criteria2] ...) The conditions are referred to as criteria1, criteria2, and so on, which can check things like:

- If a number is greater than another number >
- If a number is smaller than another number <
- If a number or text is equal to something =

The criteria\_range1, criteria\_range2, and so on, are the ranges where the function check for the conditions. The [sum\_range] is the range where the function calculates the sum.

**Figure 3.3: SUMIFS SUMPRODUCT**

To calculate the sum of the products of corresponding numbers in one or more ranges, use Excel's powerful SUMPRODUCT function. Chamel Devi Group of Institutions Department of Artificial Intelligence & Data Science

**Figure 3.4: SUMPRODUCT Explanation:**

the SUMPRODUCT function performs this calculation:  $(2 * 1000) + (4 * 250) + (4 * 100) + (2 * 50) = 3500$ .

VLOOKUP, HLOOKUP HLOOKUP and VLOOKUP are functions in Microsoft Excel that allow you to use a section of your spreadsheet as a lookup table. When the VLOOKUP function is called, Excel searches for a lookup value in the leftmost column of a section of your spreadsheet called the table array. The function returns another value in the same row, defined by the column index number. HLOOKUP is similar to VLOOKUP, but searches a row instead of a column, and

the result is offset by a row index number. The V in VLOOKUP stands for vertical search (in a single column), while the H in HLOOKUP stands for horizontal search (within a single row). Figure 3.5: VLOOKUP VLOOKUP allows you to search a table that is set up vertically. That is, all of the data is set up in columns and each column is responsible for one kind of data. In the Student Record example, there would be a separate column of data for Student Names, one for Student ID numbers, etc. HLOOKUP is the exact same function, but looks up data that has been formatted by rows instead of columns. Figure 3.6: HLOOKUP XLOOKUP Use the XLOOKUP function to find things in a table or range by row. For example, look up the price of an automotive part by the part number, or find an employee name based on their employee ID. With XLOOKUP, Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science you can look in one column for a search term and return a result from the same row in another column, regardless of which side the return column is on.

Syntax The XLOOKUP function searches a range or an array, and then returns the item corresponding to the first match it finds. If no match exists, then XLOOKUP can return the closest (approximate) match. =XLOOKUP(lookup\_value, lookup\_array, return\_array, [if\_not\_found], [match\_mode], [search\_mode]) Example Uses XLOOKUP to look up a country name in a range, and then return its telephone country code. It includes the lookup\_value (cell F2), lookup\_array (range B2:B11), and return\_array (range D2:D11) arguments. It doesn't include the match\_mode argument, as XLOOKUP produces an exact match by default. Figure 3.7: XLOOKUP INDEX + MATCH The Excel INDEX function returns the value at a given location in a range or array. You can use INDEX to retrieve individual values, or entire rows and columns. The MATCH function is often used together with INDEX to provide row and column numbers. Figure 3.8: INDEX + MATCH Handling Formula Errors Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Below is the list of some most common errors that we can find in the Excel formula:

1. #NAME? Error: This Excel error usually occurs because of the non-existent function.
2. #DIV/0! Error: This Excel error because if we try to divide the number by zero or vice-versa.
3. #REF! Error: This error arises due to a reference missing.
4. #NULL! Error: This error comes due to unnecessary spaces inside the function.
5. #N/A Error: The function cannot find the required data. Maybe the wrong reference is given.
6. ##### Error: This is not a true Excel formula error, but it occurs because of a formatting issue. Probably, the value in the cell is more than the column width.
7. #VALUE! Error: This is one of the common Excel formula errors we see in Excel. It occurs due to the wrong data type of the parameter given to the function.
8. #NUM! Error: This Excel formula error because the number we have supplied to the formula is not proper.

Dynamic Array Formulas Dynamic arrays are resizable arrays that calculate automatically and return value into multiple cells based on a formula entered in a single cell. The new array (multiple cells) that we get is known as spilling and the new array has been placed in neighboring cells. It is not necessary to use Ctrl + Shift + Enter to enter an array formula. Dynamic array formulas are only available in Excel 365 and Excel 2021.

Dynamic Array Formulas Below mentioned array formulas are the dynamic array formulas, let's learn in detail how these formulas are applied in excel on a certain dataset.

1. UNIQUE
2. SORT
3. SORT BY
4. SEQUENCE
5. RANDARRAY
6. FILTER
7. LOOKUP

UNIQUE Formula Unique formulas return a list of unique values in a list. Unique means that element that appears only once in an array. Values can be text, number, etc. A unique function is not case-sensitive means 'Apple' and 'apple' are the same.

Syntax =UNIQUE(A3:A17)

SORT Formula Sort formula returns a list in increasing order or decreasing order. Values can be text, number, etc.

Syntax, Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science =SORT(A4: A10)

increasing order. SORTBY Formula SORTBY formula sorts the contents of a range based on values from another array. Values can be text, number, etc.

Syntax, =SORTBY(A4:A10,B4:B10,-1)

SEQUENCE Formula The sequence formula allows you to generate a list of sequential numbers in an array. List can be 1D, 2D determined by rows and columns arguments.

Syntax, =SEQUENCE (5,6)

RANDARRAY Formula RANDARRAY formula generates an array of random numbers. Generated values can be either decimals or whole numbers. The size of the array is specified by rows and columns arguments.

Syntax, =RANDARRAY(1,2)

FILTER Formula The filter formula filters a range of data based on given criteria. It returned an array of filtered values. In this formula we print only those values which are greater than 100.

Syntax, =FILTER(A4:A12,A4:A12>100)

LOOKUP Formula The Excel LOOKUP function returns a value from a range (one row or one column) or an array. If you want to search any number from an array then we use this function.

Syntax, =LOOKUP(value, lookup\_range, [result\_range])

Circular References When an Excel formula refers back to its own cell, either directly or indirectly, it creates a circular reference. For instance, if you select cell A1 and type =A1 in it, this would create an Excel circular reference. Entering any other formula or calculation referring to A1 would have the same affect, e.g. =A1\*5 or =IF(A1=1, "OK"). As soon as you hit Enter to complete such a formula, you'll get the following warning message:

Chameli Devi Group of Institutions

Department of Artificial Intelligence & Data Science Figure-3.9 Circular References Formula Auditing Formula auditing is an essential tool in Excel that enables users to show the relationship between formulas and cells. Excel Formula Auditing toolbar helps the user to quickly and easily find:

- the cells contribute to calculating a formula present in the active cell.
- the formulas that refer to the active cell. The output of Formula Auditing is presented graphically by arrow lines, thereby making the entire formula visualization effortless. It allows the user to show all the formulas in the active worksheet with a single command. In case your formulas are further referring to cells present in a different workbook, it also opens that workbook. Microsoft Excel provides several different methods to audit a Formula. If you click on the Formulas ribbon tab of Excel, you will find the Formula Auditing section. However, many users might also need to customize the ribbon to see this option. Following are the different ways using which you can audit a formula:

1. Trace Precedents
2. Trace Dependents
3. Remove Arrows
4. Show Formulas
5. Error Checking

All these methods help you with formula auditing and troubleshooting formulas.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Name: Data Science Subject Code: AD 404 Subject Notes Syllabus: Data Manipulation With Pandas- Introduction to Pandas, understanding DataFrame, Missing Values, Data operation, String Manipulation, Regular Expressions and Data learning, Outlier and Error. Visualization tool in Python: Representation of Pie Chart, Bar Chart, Histogram, Scatterplots using Python. Data Analysis, performance metrics, ROC curve, types of errors, Overfitting & Under fitting, evaluating performance of learning model: Holdout, Random sampling, cross validation and Bootstrap method. Bagging & boosting, Gradient Boosting, Random Forests, Committee Machines.

---

Course

Objectives:

- The purpose of this subject is to cover the underlying concepts and techniques used in Data Science. Some of these techniques can be used in Data Analysis & in prediction.
- To understand modern way to get insights from the data.

---

Course

Outcome (CO4): Student should be able to get insights from the data. Unit-4 Introduction to Pandas Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008. Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Why Use Pandas? Pandas allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant. Relevant data is very important in data science. Pandas gives you answers about the data. Like:  
♣ Is there a correlation between two or more columns?  
♣ What is average value?  
♣ Max value?  
♣ Min value? Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called cleaning the data. Pandas DataFrame Pandas DataFrame is a widely used data structure which works with a two-dimensional array with labeled axes (rows and columns). DataFrame is defined as a standard way to store data that has two different indexes, i.e., row index and column index. It consists of the following properties:

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science

- The columns can be heterogeneous types like int, bool, and so on.
- It can be seen as a dictionary of Series structure where both the rows and columns are indexed. It is denoted as "columns" in case of columns and "index" in case of rows.

Parameter & Description:

- data: It consists of different forms like ndarray, series, map, constants, lists, array.
- index: The Default np.arange(n) index is used for the row labels if no index is passed.
- columns: The default syntax is np.arange(n) for the column labels. It shows only true if no index is passed.
- dtype: It refers to the data type of each column.
- copy(): It is used for copying the data.

Figure-4.1 Pandas DataFrame Missing Values Missing Data can occur when no information is provided for one or more items or for a whole unit. Missing Data is a very big problem in a real-life scenarios. Missing Data can also refer to as NA(Not Available) values in pandas. In DataFrame sometimes many datasets simply arrive with missing data, either because it exists and was not collected or it never existed. For Example, Suppose different users being surveyed may choose not to share their income, some users may choose not to share the address in this way many datasets went missing. In Pandas missing data is represented by two value:

- None: None is a

Python singleton object that is often used for missing data in Python code. • NaN : NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation Pandas Data operations In Pandas, there are different useful data operations for DataFrame, which are as follows : Row and column selection We can select any row and column of the DataFrame by passing the name of the rows and column. When you select it from the DataFrame, it becomes one-dimensional and considered as Series. Filter Data Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science We can filter the data by providing some of the boolean expression in DataFrame. Null values A Null value can occur when no data is being provided to the items. The various columns may contain no values which are usually represented as NaN. In Pandas, several useful functions are available for detecting, removing, and replacing the null values in Data Frame. These functions are as follows: isnull(): The main task of isnull() is to return the true value if any row has null values. notnull(): It is opposite of isnull() function and it returns true values for not null value. dropna(): This method analyzes and drops the rows/columns of null values. fillna(): It allows the user to replace the NaN values with some other values. replace(): It is a very rich function that replaces a string, regex, series, dictionary, etc. interpolate(): It is a very powerful function that fills null values in the DataFrame or series. String Manipulation A set of a string function is available in Pandas to operate on string data and ignore the missing/NaN values. There are different string operation that can be performed using .str. option. These functions are as follows: lower(): It converts any strings of the series or index into lowercase letters. upper(): It converts any string of the series or index into uppercase letters. strip(): This function helps to strip the whitespaces including a new line from each string in the Series/index. split(' '): It is a function that splits the string with the given pattern. cat(sep=' '): It concatenates series/index elements with a given separator. contains(pattern): It returns True if a substring is present in the element, else False. replace(a,b): It replaces the value a with the value b. repeat(value): It repeats each element with a specified number of times. count(pattern): It returns the count of the appearance of a pattern in each element. startswith(pattern): It returns True if all the elements in the series starts with a pattern. endswith(pattern): It returns True if all the elements in the series ends with a pattern. find(pattern): It is used to return the first occurrence of the pattern. findall(pattern): It returns a list of all the occurrence of the pattern. swapcase: It is used to swap the case lower/upper. islower(): It returns True if all the characters in the string of the Series/Index are in lowercase. Otherwise, it returns False. isupper(): It returns True if all the characters in the string of the Series/Index are in uppercase. Otherwise, it returns False. isnumeric(): It returns True if all the characters in the string of the Series/Index are numeric. Otherwise, it returns False. Count Values This operation is used to count the total number of occurrences using 'value\_counts()' option. Regular Expressions in Pandas There are several pandas methods which accept the regex in pandas to find the pattern in a String within a Series or Dataframe object. These methods works on the same line as Pythons re module. Its really helpful if you want to find the Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science names starting with a particular character or search for a pattern within a dataframe column or extract the dates from the text. Here are the pandas functions that accepts regular expression: Methods Description count() Count occurrences of pattern in each string of the Series/Index replace() Replace the search string or pattern with the given value contains() Test if pattern or regex is contained within a string of a Series or Index. Calls re.search() and returns a boolean extract() Extract capture groups in the regex pat as columns in a DataFrame and returns the captured groups findall() Find all occurrences of pattern or regular expression in the Series/Index. Equivalent to applying re.findall() on all elements match() Determine if each string matches a regular expression. Calls re.match() and returns a boolean split() Equivalent to str.split() and Accepts String or regular expression to split on rsplit() Equivalent to str.rsplit() and Splits the string in the Series/Index from the end Data learning Pandas is one of the tools in Machine Learning which is used for data cleaning and analysis. It has features which are used for exploring, cleaning, transforming and visualizing from data. Figure-4.2 Machine Learning Process using Pandas Outlier and Error An Outlier is a data-item/object that deviates significantly from the rest of the (so-called normal)objects. They can be caused by measurement or execution errors. The analysis for outlier detection is referred to as outlier mining. There are many ways to detect the outliers, and the removal process is the data frame same as removing a data item from the panda's data frame. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Here pandas data frame is used for a more realistic approach as in real-world project need to detect the outliers arouse during the data analysis step, the same approach can be used on lists and series-type objects. Removing outliers from your dataset is not necessarily the only approach to take. As a rule of thumb there are three choices that you can take when wanting to deal with outliers in your dataset. 1. Remove - The observations are incorrect

or not representative of what you are modelling 2. Re-scale - You want to keep the observations but need to reduce their extreme nature 3. Mark - Label the outliers to understand if they had an effect on the model afterwards Methods to detect outliers in a Pandas DataFrame 1. Standard deviation - Remove the values which are a certain number of standard deviations away from the mean, if the data has a Gaussian distribution 2. Automatic outlier detection - Train a machine learning model on a smaller normal set of observations which can then predict data points outside of this normal set 3. Interquartile range - Remove the values which are above the 75th percentile or below the 25th percentile, doesn't require the data to be Gaussian There are trade-offs for each of these options, however the method most commonly used in industry is the standard deviation, or z-score, approach. Visualization tool in Python Tpoic cover in Unit-1 and Unit-2 Data Analysis Although many groups, organizations, and experts have different ways of approaching data analysis, most of them can be distilled into a one-size-fits-all definition. Data analysis is the process of cleaning, changing, and processing raw data and extracting actionable, relevant information that helps businesses make informed decisions. The procedure helps reduce the risks inherent in decision-making by providing useful insights and statistics, often presented in charts, images, tables, and graphs. Types of data analysis • Descriptive analysis • Diagnostic analysis • Predictive analysis • Prescriptive analysis Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Performance Metrics Evaluating the performance of a Machine learning model is one of the important steps while building an effective ML model. To evaluate the performance or quality of the model, different metrics are used, and these metrics are known as performance metrics or evaluation metrics. These performance metrics help us understand how well our model has performed for the given data. In this way, we can improve the model's performance by tuning the hyper-parameters. Each ML model aims to generalize well on unseen/new data, and performance metrics help determine how well the model generalizes on the new dataset. In machine learning, each task or problem is divided into classification and Regression. Not all metrics can be used for all types of problems; hence, it is important to know and understand which metrics should be used. Different evaluation metrics are used for both Regression and Classification tasks. Performance Metrics for Classification In a classification problem, the category or classes of data is identified based on training data. The model learns from the given dataset and then classifies the new data into classes or groups based on the training. It predicts class labels as the output, such as Yes or No, 0 or 1, Spam or Not Spam, etc. To evaluate the performance of a classification model, different metrics are used, and some of them are as follows: • Accuracy • Confusion Matrix • Precision • Recall • F-Score • AUC(Area Under the Curve)-ROC Regression is a supervised learning technique that aims to find the relationships between the dependent and independent variables. A predictive regression model predicts a numeric or discrete value. The metrics used for regression are different from the classification metrics. It means we cannot use the Accuracy metric (explained above) to evaluate a regression model; instead, the performance of a Regression model is reported as errors in the prediction. Following are the popular metrics that are used to evaluate the performance of Regression models. • Mean Absolute Error • Mean Squared Error • R2 Score • Adjusted R2 ROC curve In Machine Learning, only developing an ML model is not sufficient as we also need to see whether it is performing well or not. It means that after building an ML model, we need to evaluate and validate how good or bad it is, and for such cases, we use different Evaluation Metrics. AUC-ROC curve is such an evaluation Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science metric that is used to visualize the performance of a classification model. It is one of the popular and important metrics for evaluating the performance of the classification model. AUC-ROC curve is a performance measurement metric of a classification model at different threshold values. Firstly, let's understand ROC (Receiver Operating Characteristic curve) curve. ROC or Receiver Operating Characteristic curve represents a probability graph to show the performance of a classification model at different threshold levels. The curve is plotted between two parameters, which are: • True Positive Rate or TPR • False Positive Rate or FPR In the curve, TPR is plotted on Y-axis, whereas FPR is on the X-axis. TPR: TPR or True Positive rate is a synonym for Recall, which can be calculated as: FPR or False Positive Rate can be calculated as: Here, TP: True Positive FP: False Positive TN: True Negative FN: False Negative Now, to efficiently calculate the values at any threshold level, we need a method, which is AUC. AUC: Area Under the ROC curve AUC is known for Area Under the ROC curve. As its name suggests, AUC calculates the two-dimensional area under the entire ROC curve ranging from (0,0) to (1,1), as shown below image: Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Figure-4.3 ROC curve In the ROC curve, AUC computes the performance of the binary classifier across different thresholds and provides an aggregate measure. The value of AUC ranges from 0 to 1, which means an excellent model will have AUC near 1, and hence it will show a good measure of Separability. Types of error

Error (statistical error) describes the difference between a value obtained from a data collection process and the 'true' value for the population. The greater the error, the less representative the data are of the population. Data can be affected by two types of error: sampling error and non-sampling error. Sampling error Sampling error occurs solely as a result of using a sample from a population, rather than conducting a census (complete enumeration) of the population. It refers to the difference between an estimate for a population based on data from a sample and the 'true' value for that population which would result if a census were taken. Sampling errors do not occur in a census, as the census values are based on the entire population. Sampling error can occur when:

- the proportions of different characteristics within the sample are not similar to the proportions of the characteristics for the whole population (i.e. if we are taking a sample of men and women and we know that 51% of the total population are women and 49% are men, then we should aim to have similar proportions in our sample)
- the sample is too small to accurately represent the population
- the sampling method is not random

Sampling error can be measured and controlled in random samples where each unit has a chance of selection, and that chance can be calculated. In general, increasing the sample size will reduce the sample error.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Non-sampling error Non-sampling error is caused by factors other than those related to sample selection. It refers to the presence of any factor, whether systemic or random, that results in the data values not accurately reflecting the 'true' value for the population. Non-sampling error can occur at any stage of a census or sample study, and are not easily identified or quantified. Non-sampling error can include (but is not limited to):

- Coverage error: this occurs when a unit in the sample is incorrectly excluded or included, or is duplicated in the sample (e.g. a field interviewer fails to interview a selected household or some people in a household).
- Non-response error: this refers to the failure to obtain a response from some unit because of absence, non-contact, refusal, or some other reason. Non-response can be complete non-response (i.e. no data has been obtained at all from a selected unit) or partial non-response (i.e. the answers to some questions have not been provided by a selected unit).
- Response error: this refers to a type of error caused by respondents intentionally or accidentally providing inaccurate responses. This occurs when concepts, questions or instructions are not clearly understood by the respondent; when there are high levels of respondent burden and memory recall required; and because some questions can result in a tendency to answer in a socially desirable way (giving a response which they feel is more acceptable rather than being an accurate response).
- Interviewer error: this occurs when interviewers incorrectly record information; are not neutral or objective; influence the respondent to answer in a particular way; or assume responses based on appearance or other characteristics.
- Processing error: this refers to errors that occur in the process of data collection, data entry, coding, editing and output.

Overfitting Overfitting occurs when our machine learning model tries to cover all the data points or more than the required data points present in the given dataset. Because of this, the model starts caching noise and inaccurate values present in the dataset, and all these factors reduce the efficiency and accuracy of the model. The overfitted model has low bias and high variance. The chances of occurrence of overfitting increase as much we provide training to our model. It means the more we train our model, the more chances of occurring the overfitted model. Overfitting is the main problem that occurs in supervised learning. Example: The concept of the overfitting can be understood by the below graph of the linear regression output: Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Figure-4.4 Overfitting As we can see from the above graph, the model tries to cover all the data points present in the scatter plot. It may look efficient, but in reality, it is not so. Because the goal of the regression model to find the best fit line, but here we have not got any best fit, so, it will generate the prediction errors. How to avoid the Overfitting in Model Both overfitting and underfitting cause the degraded performance of the machine learning model. But the main cause is overfitting, so there are some ways by which we can reduce the occurrence of overfitting in our model.

- Cross-Validation
- Training with more data
- Removing features
- Early stopping the training
- Regularization
- Ensembling

Underfitting Underfitting occurs when our machine learning model is not able to capture the underlying trend of the data. To avoid the overfitting in the model, the fed of training data can be stopped at an early stage, due to which the model may not learn enough from the training data. As a result, it may fail to find the best fit of the dominant trend in the data. In the case of underfitting, the model is not able to learn enough from the training data, and hence it reduces the accuracy and produces unreliable predictions. An underfitted model has high bias and low variance. Example: We can understand the underfitting using below output of the linear regression model: Figure-4.5 Underfitting Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science As we can see from the above diagram, the model is unable to capture the data points present in the plot. How to avoid underfitting.

- By

increasing the training time of the model. • By increasing the number of features. Evaluating performance of learning model Model evaluation is the process that uses some metrics which help us to analyze the performance of the model. As we all know that model development is a multi-step process and a check should be kept on how well the model generalizes future predictions. Therefore evaluating a model plays a vital role so that we can judge the performance of our model. The evaluation also helps to analyze a model's key weaknesses. There are many metrics like Accuracy, Precision, Recall, F1 score, Area under Curve, Confusion Matrix, and Mean Square Error. Cross Validation is one technique that is followed during the training phase and it is a model evaluation technique as well. Holdout Holdout is the simplest approach. It is used in neural networks as well as in many classifiers. In this technique, the dataset is divided into train and test datasets. The dataset is usually divided into ratios like 70:30 or 80:20. Normally a large percentage of data is used for training the model and a small portion of the dataset is used for testing the model. cross validation Cross Validation is a method in which we do not use the whole dataset for training. In this technique, some part of the dataset is reserved for testing the model. There are many types of Cross-Validation out of which K Fold Cross Validation is mostly used. In K Fold Cross Validation the original dataset is divided into k subsets. The subsets are known as folds. This is repeated k times where 1 fold is used for testing purposes. Rest k-1 folds are used for training the model. So each data point acts as a test subject for the model as well as acts as the training subject. It is seen that this technique generalizes the model well and reduces the error rate. Random sampling. As the name suggests, in this method of sampling, the data is collected at random. It means that every item of the universe has an equal chance of getting selected for the investigation purpose. In other words, each item has an equal probability of being in the sample, which makes the method impartial. As there is no control of the investigator in selecting the sample, the random sampling method is used for homogeneous items. As there is no tool or a large number of people required for collecting data through random sampling, this method is economical. There are two ways of collecting data through the random sampling method. These are the Lottery Method and Tables of Random Numbers. • Lottery Method: In Lottery Method, the investigator prepares paper slips for each of the items of the universe and shuffles these slips in a box. After that, some slips are impartially drawn out from the box to obtain a sample. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science • Table of Random Numbers: A Table of Random Numbers has been prepared by a group of statisticians. In this method of collecting data through random sampling, this table is referred by the investigator to frame a sample. There are many Tables of Random Numbers available from which Tippet's Table is used by most of the investigators. In this Table, Tippet has used 41,600 figures and has involved 10,400 numbers with four units in each of the numbers. Now, through this method, the items available in the universe are first arranged in an order, and then using Tippet's Table, the investigator selects the required number of items to form a sample. Merits of Random Sampling Method 1. Random Sampling method is economical as the items are selected randomly, which can be done by fewer people and with fewer resources. 2. Random Sampling method is impartial and free from personal biases, as it randomly selects the numbers, and each of the items has an equal probability of getting selected. 3. This method fairly represents the universe through samples. 4. It is a straightforward and simple method of collecting data. Demerits of Random Sampling Method 1. Despite its various advantages, the random sampling method does not give proper weightage to some important items of the universe. 2. Also, there is no guarantee that different items of the universe are proportionately represented. Bootstrap method A statistical concept, Bootstrapping is a resampling method used to stimulate samples out of a data set using the replacement technique. The process of bootstrapping allows one to infer data about the population, derive standard errors, and ensure that data is tested efficiently. In simple terms, the Bootstrapping Method, in Statistics and Machine Learning, is a resampling statistical technique that evaluates statistics of a given population by testing a dataset by replacing the sample. This technique involves repeatedly sampling a dataset with random replacement. A statistical test that falls under the category of resampling methods, this method ensures that the statistics evaluated are accurate and unbiased as much as possible. Unlike other sampling distribution methods, the Bootstrapping method uses the samples procured from a study over and over again in order to use the replacement technique and ensure that the stimulated samples lead to an accurate evaluation. Other than ensuring the sampling of the accuracy of a given dataset, bootstrapping in statistics also allows one to estimate the confidence intervals of a given dataset. A confidence interval is defined as the level of certainty with which an estimated statistic contains the true value of the parameter. There are 2 types of bootstrapping methods that are applicable in statistics and Machine Learning. 1. Parametric Bootstrap Method In this method, the distribution parameter must be known. This means that the assumption of the kind of distribution the sample has must be provided beforehand.

For instance, it must be known to the user if the Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science sample has Gaussian Distribution or Skewed distribution. This type of bootstrap method is more efficient since it already knows the nature of distribution.

## 2. Non-Parametric Bootstrap Method

Unlike the parametric bootstrap method, this type does not require the parameter of distribution to be known beforehand. Therefore, this type of bootstrap method works without assuming the nature of the sample distribution.

**Bagging** Bootstrap Aggregating, also known as bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It decreases the variance and helps to avoid overfitting. It is usually applied to decision tree methods. Bagging is a special case of the model averaging approach.

**Boosting** Boosting is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models is added.

**Gradient Boosting** Gradient Boosting is a powerful boosting algorithm that combines several weak learners into strong learners, in which each new model is trained to minimize the loss function such as mean squared error or cross-entropy of the previous model using gradient descent. In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimize this gradient. The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met.

**Random Forest** Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science

Figure-4.6 Random Forest

Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

**Committee Machines**

A committee machine is a machine learning algorithm that is trained using a committee of models, each of which is trained on a different subset of the data. The predictions of the committee are then combined to make a final prediction. The advantages of using a committee machine are that it can reduce overfitting and improve accuracy. The disadvantage is that it can be computationally expensive to train a large number of models.

Benefits of using a committee machine

- There are many benefits of using a committee machine in AI. A committee machine is a machine learning algorithm that combines the predictions of multiple models to produce a more accurate prediction.
- One benefit of using a committee machine is that it can help to reduce overfitting. Overfitting is a problem that can occur when a machine learning model is too closely fit to the training data. This can cause the model to perform poorly on new data. A committee machine can help to reduce overfitting by combining the predictions of multiple models.
- Another benefit of using a committee machine is that it can help to improve the accuracy of predictions. This is because a committee machine can make use of the different strengths of each individual model.
- Finally, a committee machine can also help to reduce the amount of time needed to train a machine learning model. This is because the training of each individual model can be parallelized.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science

- Overall, there are many benefits of using a committee machine in AI. A committee machine can help to reduce overfitting, improve the accuracy of predictions, and reduce the amount of time needed to train a machine learning model.

How does a committee machine work?

A committee machine is a machine learning algorithm that is used to ensemble multiple models. It works by training multiple models on the same data and then combining the predictions of the models. The predictions of the models are combined using a weighted average, where the weights are based on the accuracy of the models. The committee machine is a powerful machine learning algorithm that can be used to improve the accuracy of predictions. It is especially useful when there is a

lot of data, as the multiple models can learn from different parts of the data. The committee machine is also robust to overfitting, as the multiple models can average out the noise in the data.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Name: Data Science Subject Code: AD 404 Subject Notes Syllabus: Introduction to Business Intelligence- Introduction, Types of Business Intelligence, Modern Business Intelligence Tools, Modern Business Intelligence. Data Science and Ethical Issues- Unfair discrimination, Reinforcing human biases, Lack of transparency. Discussions on privacy, security, ethics, Role of Nextgeneration data scientists.

Course

Objectives: • The purpose of this subject is to cover the underlying concepts and techniques used in Data Science. Some of these techniques can be used in Data Analysis & in prediction. • To understand modern way to get insights from the data.

Course

Outcome (CO5): To Understand the Concept of Business Intelligence Unit-5 Introduction Business intelligence (BI) is all about turning an organization's data into insights that can be used to inform business decisions. BI analysts will use BI tools, software or services to access and analyze datasets and translate their findings into reports, summaries, dashboards, graphs, charts or maps. In recent years, the advent of modern data visualization and reporting tools has transformed the discipline, empowering businesses to use big data insights to identify, develop and create new business opportunities. Howard Dresner played a key role in popularizing business intelligence during his time as a VP and Research Fellow at Gartner. In 1989, he defined business intelligence as an umbrella term which describes concepts and methods that can help businesses use "fact-based support systems" to improve their decision-making. Today, Forrester defines business intelligence as "a set of methodologies, processes, architectures and technologies that transform raw data into meaningful and useful information". This can then be "used to enable more effective strategic, tactical and operational insights and decision-making". This definition acknowledges that data cannot be effectively analyzed or used to generate meaningful insights if it is poor quality. Some would argue that data management and ingestion should not be the business intelligence team's responsibility. But it is widely acknowledged that BI professionals currently spend a significant amount of their time on data preparation and cleaning data. BI should not be confused with 'business analytics'. Business intelligence is descriptive and uses metrics to generate clear snapshots of business performance. Meanwhile, business analytics is predictive, and describes what organizations should do in future to generate better outcomes. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Importance of Business Intelligence Business intelligence helps companies make better decisions by displaying current and historical data in a business context.

Analysts can use business intelligence to provide performance and set benchmarks to their peers, thus, allowing organizations to run more smoothly and efficiently. Analysts can also easily spot market trends to increase sales and revenue. Properly used data helps you with everything from compliance to recruitment. Here are some ways business intelligence can help organizations make wiser, data-driven decisions. • Identification of multiple ways to increase profits

• Analyzing customer behavior • Compare data with competitors • Track performance • Operational optimization •

Predict success • Spot market trends • Discover problems and find solutions Types of Business Intelligence 1. Executive scorecard and dashboard

The company's topmost-level managers generally don't have time to create ad hoc reports. As a result, many business intelligence tools provide executives with real-time monitoring of key performance indicators to provide a graphical dashboard that gives an overview of the company's overall situation at a glance. 2. OLAP analysis Business analysts and other power users, who are responsible for revealing the most sophisticated trends in the corporate data, need to be able to analyze the information in detail. Most business intelligence tools provide a wide range of online analytical processing (OLAP) capabilities that give users instant access to their data in an unlimited number of ways, so they can see their data from multiple perspectives. 3. Ad hoc report In many cases, updating the content of existing reports is sufficient to meet the specific information needs of employees. However, employees often need to quickly create new reports to answer urgent questions, make voluntary decisions, and resolve the following issues:

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Many of today's most popular business intelligence tools have the ability to quickly, and Easily create and generate your own custom reports, even for non-technical users. 4. Operation report Every day, every company performs thousands of operational tasks. Organizations

perform these activities, and inefficiencies and errors can have a significant impact on performance. Business intelligence tools can be used to support the type of operational reporting that enables realtime monitoring of daily events. In this way, you can quickly identify and fix the problem.

5. Forecast The ability of a company to predict trends is important for maintaining organizational flexibility and agility. In addition, historical data should be used to predict future events to ensure effective strategic planning. For this reason, many business intelligence tools include predictive analytics, which allows for quick and highly accurate forecasts.

6. Data mining Today's businesses manage vast amounts of information. Sorting that data to find the ones that are most relevant can also be a daunting task, especially for non-technical business users. The data mining capabilities of many business intelligence tools help you find and extract the most important information from large datasets, so you don't have to waste time searching for the data, you need to get the information you really need. This proves to be much easier to access and use.

7. Customer intelligence As companies aim to be more customer-centric, they are looking to business intelligence tools to collect and integrate customer data that resides in various systems such as CRM, accounting, and help desk applications. This gives enterprises a complete 360-degree view of all customer interactions so that companies better understand their needs, behaviors, and preferences and use that knowledge for more successful loyalty and upsell/cross-sell programs.

**Modern Business Intelligence**

Modern BI addresses the new business reality with Self-Serve Tools to satisfy the needs of an average business user with sophisticated features in an easy-to-use integrated BI environment. Modern BI shifts the focus away from IT and data scientist analysis and reporting and offers mainstream tools with access and flexibility so that business users can produce reports and analysis on-the-fly and share data with other users to make decisions and optimize business results.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science BI dashboards are more flexible and support personalization. They are no longer restricted to formats and delivery designed by IT. Disparate data sources are integrated into a uniform view, so users can drag and drop data and interact with the tools. The design of modern BI tools recognizes the reality of time-sensitive, rapidly changing business markets and competitive positioning and provides more options and flexibility to support data popularity, Social BI and power users within the organization. Every business understands the value of objective metrics and accurate analysis and the modern BI environment is designed to support these goals at every level of the organization.

**Modern BI Different From Traditional BI**

The primary difference between traditional BI and the new, more modern approach to BI lies in the flexibility and accessibility. The more traditional tools were designed for use by analysts or IT staff and, while these tools provided sophisticated features, these features were not accessible or easy to understand for anyone outside the analyst or technical community. The traditional tools were not scalable or flexible enough for mobile, nor did they provide guided, auto-recommendations in a natural language environment. Because traditional BI solutions were not designed to support use by team members within the line of business or business users in general, the enterprise could not capitalize on the unique perspective, knowledge or skill of these users to advance business results, plan for future results or solve problems.

The September 12, 2017 Gartner Report (ID G00331857), entitled, 'Technology Insight for Modern Analytics and Business Intelligence Platforms' predicts that 'By 2020, smart/augmented, nonrelational-, search- and visual-based data discovery capabilities will converge into a single set of next-generation data discovery capabilities as components of modern BI and analytics platforms. Modern BI solutions allow for and support user adoption, and deliver more benefit, better ROI and lower TCO to the organization by empowering business users and holding each team member accountable for results. While traditional BI was the domain of IT and the analyst community, the modern BI environment expands the use of analytical tools throughout the organization. Modern BI frees the IT and analyst team to focus on more strategic goals. Modern BI supports collaboration, while providing appropriate data governance and data security. The modern BI environment provides a foundation of data source integration, sophisticated analytical models and techniques and an easy-to-use, intuitive interface with auto-suggestions and recommendations for analytical techniques and report formats and publishing tools, alerts and analysis sharing that are easy for the average business user to adopt.

**Business Intelligence Tools**

Business intelligence tools (BI tools) are all about helping you understand trends and deriving insights from your data so that you can make tactical and strategic business decisions. They also help you identify patterns in the mountains of data your business builds up

1. SAP Business Objects
2. Datapine
3. MicroStrategy
4. SAS Business Intelligence
5. Yellowfin BI
6. QlikSense
7. Zoho Analytics
8. Sisense
9. Microsoft Power BI
10. Tableau

Data Science and Ethical Issues Today, data science has a significant impact on how businesses are conducted in disciplines as diverse as medical sciences, smart cities, and transportation. Whether it's the protection of personally

identifiable data, implicit bias in automated decision-making, the illusion of free choice in psychographics, the social impacts of automation, or the apparent divorce of truth and trust in virtual communication, the dangers of data science without ethical considerations are as clear as ever. The need for a focus on data science ethics extends beyond a balance sheet of these potential problems because data science practices challenge our understanding of what it means to be human. Algorithms, when implemented correctly, offer enormous potential for good in the world. When we employ them to perform jobs that previously required a person, the benefits may be enormous: cost savings, scalability, speed, accuracy, and consistency, to name a few. And because the system is more precise and reliable than a human, the outcomes are more balanced and less prone to social prejudice. Unfair Discrimination What is a fair defendant evaluation and how can data scientists develop a corresponding algorithm that works for any defendant. Northpointe's definition of a fair algorithm states that the proportion of defendants who re-offend in each risk category is approximately the same regardless of race. In other words, a risk score of e.g. 7 predicts an equal likelihood of re-offending for black and white defendants. Any alternative definition would, by definition, discriminate against white defendants by artificially boosting their risk levels. Researchers from Stanford and Berkeley have validated that a single risk score represents an approximately equal risk of actual recidivism based on roughly 5,000 samples from Broward County, Florida (Corbett-Davies et al., 2016). However, ProPublica, an investigative news organisation, identified unfair discrimination against black defendants by analysing over 10,000 defendant samples from the Broward County Sheriff's Office in Florida. They compared recidivism risk scores issued by COMPAS and actual recidivism rates two years after scores were issued and found that the algorithm classified black defendants as more than twice as likely to be considered medium to high risk than its white counterparts, even though neither black nor white defendants went on to commit a crime. Although black defendants did not re-offend, they were subject to harsher treatment by the court system (Larson et al., 2016). Discriminatory treatment is not ethically problematic in itself; the effects of the treatment, i.e. being sent to jail or released on bail depending on race, are the ethical problems of this application (Schermer, 2011). Chamelei Devi Group of Institutions Department of Artificial Intelligence & Data Science Can an algorithm mitigate ProPublica's legitimate concern while satisfying Northpointe's definition of fairness. Reinforcing human biases Both aspects of Data Science defined earlier (vast amounts of data and statistical models) lead to an inherent limitation for making decisions based on patterns in past data. COMPAS' developers presumably fed the system with recidivism data to let the model identify variables correlating with a likelihood to re-offend — a standard way to train classification algorithms. If the data were perfectly unbiased there would be no problems, but all COMPAS can do is parrot back to us our own biases. These include severe, racist biases of judges that see black defendants as "more likely to kill again than whites" (Epps, 2017) or a Mexican defendant to have committed sexual assault "because he's Mexican and Mexican men take whatever they want" (Supreme Court of Colorado, 2017). Abdicating the responsibility for sentencing to a computer allows judges to make decisions based not only on relevant factors such as the seriousness of the offence but also on ethically problematic factors like inferred race and gender. Assume COMPAS predicts a defendant to be high risk. However, a judge disagrees and releases the defendant, who later re-offends. Why did the judge not comply with the software's recommendation? This outcome is a lot easier to spot than when COMPAS suggests low risk, but the judge sentences the defendant and (probably) prevents recidivism. Lack of transparency Hiding COMPAS' inner workings from the public prohibits the understanding and discussion around mitigating learned biases. However, Northpointe understandably argues that publishing its proprietary algorithms would lead to a competitive disadvantage "because it's certainly a core piece of our business" (Liptak, 2017) as an executive is quoted in the New York Times. This dilemma comes as no surprise when government agencies authorise private companies to apply Data Science to sensitive governmental data. There are two key areas requiring transparency. First, transparency with regards to statistical modelling requires Northpointe to disclose the step-by-step process leading to risk scores, the underlying model and its parameters. Depending on the model(s) used, this task can be very difficult as the exact workings of models like neural networks are still unclear. Should lawmakers only allow algorithms that are fully understood to make such wide-reaching recommendations. Second, it remains unclear what data is used. While race is not an input variable, other variables like gender are as made evident in the proceedings of Eric L. Loomis' case (Supreme Court of Wisconsin, 2016). It would be ethically very problematic if the dataset used for training could infer race based on other variables. With over 100 input variables, chances are that some subset of variables can accurately predict race. In that case, the statistical model cannot distinguish between the predictive power of a single variable and that set of variables. Northpointe's Chief Scientist Tim Brennan even admitted that "it is difficult to construct a score that doesn't

include items that can be correlated with race" (Angwin et al., 2016). Sharing the correlation between variables does not conflict with trade secrets. Should defendants have a claim right to know how COMPAS computes its score? Does COMPAS violate a defendant's right to an individualised sentence and right to be sentenced based on accurate information ("State v. Loomis," 2017) Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Discussions on privacy Data scientists are continually involved in producing, developing, and receiving information. Data concerning client affiliates, customers, workers, or other parties with whom the clients have a confidentiality agreement is often included in this category. Then, regardless of the sort of sensitive information, it is the data scientist's responsibility to protect it. Only when the customer provides permission for data scientists to share or talk about this type of information should it be disclosed or spoken about. Complete privacy of clients' or customers' data must be maintained. Even if a consumer consents to your organization collecting, storing, and analyzing their personally identifiable information (PII), that doesn't mean they want it made public. Personally, identifiable information includes: Phone Number, Address, Full Name, PAN card number, and so on. To preserve people's privacy, make sure you're keeping the information in a secure database so it doesn't get into the wrong hands. Dual-authentication password protection and file encryption are two data security solutions that assist safeguard privacy. Security We are all living in a digital world, where our day-to-day life is dependent on applications, run by tech companies. We need to take a taxi, we call an Uber. We need to order food, we use Zomato and so on. These companies have our personal data. Our email ID, phone numbers, address, purchase history, etc, and so on. The protection of personal data is thus an important aspect in the present day. Perhaps no aspect of data science ethics has gotten greater attention in recent years than the safeguarding of personal data. Our relationships with social and economic networks have undergone a digital revolution, revealing who we are, what we believe, and what we do. In India, the Personal Data Protection Bill affirms the rights of digital citizens and addresses the hazards of commercial exploitation of personal and personally identifiable data. The Data Protection Bill is a long-awaited and desperately needed piece of legislation that would replace India's present antiquated, obsolete, and inadequate data protection policy. It has the potential to raise user understanding of their privacy and hold data custodians and processors accountable. Ethics The term "ethics" comes from the Greek word Ethos, which means "habit" or "custom." Ethics instructs us on what is good and wrong. Philosophers have pondered this crucial topic for a long time and have a lot to say about it. Most people associate ethics with morality: a natural sense of what is "good." We as humans live in a society, and society has rules and regulations. We must be able to decide what is right and what is wrong. Ethics deals with feelings, laws, and social norms which determine right from wrong. Our ways of life must be reasonable and live up to the standards of society. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Role of Next-generation data scientists With the current generation of data scientists working tirelessly to satisfy the accelerating demand for datadriven insights on behalf of industries pretty much across the board, it's natural to take a step back and ask what the "data scientist 2.0" might look like in the next 5–10 years. we'll take out our crystal ball and try to characterize this next generation. Ideally, the next generation of data scientist should be much more evolved past someone who is technically proficient to the degree necessary to land a cushy job with a life-changing salary (although those things are certainly nice). In contrast, next-generation data scientists should be encouraged to become good problem solvers who follow the scientific method, to think deeply about the appropriate use of the data science process, and to use data responsibly and for the common good.

- Technical Skills Fully Honed.
- Slow Down and Proceed Methodically
- Soft Skills are King
- Apply the Scientific Method
- Proceed with Ethics

Chameli Devi Group of Institutions Department of Artificial intelligence and Data Science Subject Notes AD405 Operating Systems UNIT-III Memory Management: Different Memory Management Techniques – Partitioning, Swapping, Segmentation, Paging, Paged Segmentation, Comparison of these Techniques, Techniques for supporting the execution of large programs: Overlay, Dynamic Linking and Loading, Virtual Memory – Concept, Implementation by Demand Paging etc. Memory management in UNIX & Windows Memory Management

- Memory Management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution.
- It checks how much memory is to be allocated to processes. It decides which process

will get memory at what time. • It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

**Process Address Space** The process address space is the set of logical addresses that a process references in its code. For example, when 32-bit addressing is in use, addresses can range from 0 to 0xffffffff; that is,  $2^{31}$  possible numbers, for a total theoretical size of 2 gigabytes. The operating system takes care of mapping the logical addresses to physical addresses at the time of memory allocation to the program. There are three types of addresses used in a program before and after memory is allocated – S. No. Memory Addresses & Description

- 1 Symbolic addresses - The addresses used in a source code. The variable names, constants, and instruction labels are the basic elements of the symbolic address space.
- 2 Relative addresses - At the time of compilation, a compiler converts symbolic addresses into relative addresses.
- 3 Physical addresses - The loader generates these addresses at the time when a Table: Types of addresses used in a program Virtual and Physical addresses are the same in compile-time and load-time address-binding schemes. Virtual and physical addresses differ in execution-time address-binding scheme. The set of all logical addresses generated by a program is referred to as a logical address space. The set of all physical addresses corresponding to these logical addresses is referred to as a physical address space. The runtime mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device. MMU uses following mechanism to convert virtual address to physical address:

- The value in the base register is added to every address generated by a user process, which is treated as offset at the time it is sent to memory. For example, if the base register value is 10000, then an attempt by the user to use address location 100 will be dynamically reallocated to location 10100.
- The user program deals with virtual addresses; it never sees the real physical addresses.

**Memory Management Technique- (Partitioning)** Main memory usually has two partitions –

- Low Memory– Operating system resides in this memory.
- High Memory– User processes are held in high memory.

Operating system uses the following memory allocation mechanism:

- S. No. Memory Allocation & Description
- 1 Single-partition allocation - In this type of allocation, relocation-register scheme is used to protect user processes from each other, and from changing operating system code and data. Relocation register contains value of smallest physical address whereas limit register contains range of logical addresses. Each logical address must be less than the limit register.
- 2 Multiple-partition allocation - In this type of allocation, main memory is divided into a number of fixed-sized partitions where each partition should contain only one process. When a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for another process.

**Table: Memory Allocation (partitioning) Schemes** Swapping

Swapping is mechanisms in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes. At some later time, the system swaps back the process from the secondary storage to main memory. Though performance is usually affected by swapping process but it helps in running multiple and big processes in parallel and that's the reason. Swapping is also known as a technique for memory compaction.

**Figure: Swapping** The total time taken by swapping process includes the time it takes to move the entire process to a secondary disk and then to copy the process back to memory, as well as the time the process takes to regain main memory.

**Physical Address = Frame number + page offset**

**Logical Address = Page number + page offset**

**Segmentation** Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program. When a process is to be executed, its corresponding segmentation is loaded into noncontiguous memory though every segment is loaded into a contiguous block of available memory. Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size. A program segment contains the program's main function, utility functions, data structures, and so on. The operating system maintains a segment map table for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory. For each segment, the table stores the starting address of the segment and the length of the segment. A reference to a memory location includes a value that identifies a segment and an offset.

**Figure 3.11: Segmentation**

**Paging** is a memory management technique in which process address space is broken into blocks of the same size called pages (size is power of 2, between 512 bytes and 8192 bytes). The size of the process is measured in the number of pages. Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called frames and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.

**Address Translation** Page address is called logical address and represented by page number and the offset. Frame address is called physical address and represented by a frame number and the offset. A data structure called page map table is

used to keep track of the relation between a pages of a process to a frame in physical memory. Figure: Paging Figure: Address Translation Segmented Paging Pure segmentation is not very popular and not being used in many of the operating systems. However, Segmentation can be combined with Paging to get the best features out of both the techniques. In Segmented Paging, the main memory is divided into variable size segments which are further divided into fixed size pages. 1. Pages are smaller than segments. 2. Each Segment has a page table which means every program has multiple page tables. 3. The logical address is represented as Segment Number (base address), Page number and page offset. Segment Number → It points to the appropriate Segment Number. Page Number → It Points to the exact page within the segment Page Offset → Used as an offset within the page frame Figure: Segmented Paging Fragmentation As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as Fragmentation. Fragmentation is of two types – S. No. Fragmentation & Description 1 External fragmentation - Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used. 2 Internal fragmentation - Memory block assigned to process is bigger. Some portion of memory is left unused, as it cannot be used by another process. Table: Types of fragmentations The following diagram shows how fragmentation can cause waste of memory and a compaction technique can be used to create more free memory out of fragmented memory – Figure: Fragmentation External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic. The internal fragmentation can be reduced by effectively assigning the smallest partition but large enough for the process. Translation of logical address to physical address The CPU generates a logical address which is divided into two parts: Segment Number and Segment Offset. The Segment Offset must be less than the segment limit. Offset is further divided into Page number and Page Offset. To map the exact page number in the page table, the page number is added into the page table base. The actual frame number with the page offset is mapped to the main memory to get the desired word in the page of the certain segment of the process. Figure: Translation of logical address to physical address Comparison of Memory Management Technique S.NO Paging Segmentation 1. In paging, the program is divided into fixed or mounted size pages. In segmentation, the program is divided into variable size sections. 2. For the paging operating system is accountable. For segmentation compiler is accountable. 3. Page size is determined by hardware. Here, the section size is given by the user. 4. It is faster in comparison to segmentation. Segmentation is slow. 5. Paging could result in internal fragmentation. Segmentation could result in external fragmentation. 6. In paging, the logical address is split into a page number and page offset. Here, the logical address is split into section number and section offset. 7. Paging comprises a page table that encloses the base address of every page. While segmentation also comprises the segment table which encloses segment S.NO Paging Segmentation number and segment offset. 8. The page table is employed to keep up the page data. Section Table maintains the section data. 9. In paging, the operating system must maintain a free frame list. In segmentation, the operating system maintains a list of holes in the main memory. 10. Paging is invisible to the user. Segmentation is visible to the user. 11. In paging, the processor needs the page number, and offset to calculate the absolute address. In segmentation, the processor uses segment number, and offset to calculate the full address. 12. It is hard to allow sharing of procedures between processes. Facilitates sharing of procedures between the processes. 13. In paging, a programmer cannot efficiently handle data structure. It can efficiently handle data structures. 14. This protection is hard to apply. Easy to apply for protection in segmentation. 15. The size of the page needs always be equal to the size of frames. There is no constraint on the size of segments. 16. A page is referred to as a physical unit of information. A segment is referred to as a logical unit of information. 17. Paging results in a less efficient system. Segmentation results in a more efficient system. Technique for Execution of large Programs: Overlay: The main problem in fixed partitioning is the size of a process has to be limited by the maximum size of the partition, which means a process can never be span over another. In order to solve this problem, earlier people have used some solution which is called as Overlays. The concept of overlays is that whenever a process is running it will not use the complete program at the same time, it will use only some part of it. Then overlays concept says that whatever part you required, you load it an once the part is done, then you just unload it, means just pull it back and get the new part you required and run it. “The process of transferring a block of program code or other data into internal memory, replacing what is already stored”. Sometimes it happens that compare to the size of the biggest partition, the size of the program will be even more, then, in that case, you should go with overlays. So overlay is a technique to run a program that is bigger than the size of the physical memory

by keeping only those instructions and data that are needed at any given time. Divide the program into modules in such a way that not all modules need to be in the memory at the same time. Static vs Dynamic Loading The choice between Static or Dynamic Loading is to be made at the time of computer program being developed. If you have to load your program statically, then at the time of compilation, the complete programs will be compiled and linked without leaving any external program or module dependency. The linker combines the object program with other necessary object modules into an absolute program, which also includes logical addresses. If you are writing a dynamically loaded program, then your compiler will compile the program and for all the modules which you want to include dynamically, only references will be provided and rest of the work will be done at the time of execution. At the time of loading, with static loading, the absolute program (and data) is loaded into memory in order for execution to start. If you are using dynamic loading, dynamic routines of the library are stored on a disk in relocatable form and are loaded into memory only when they are needed by the program. Static vs Dynamic Linking As explained above, when static linking is used, the linker combines all other modules needed by a program into a single executable program to avoid any runtime dependency. When dynamic linking is used, it is not required to link the actual module or library with the program, rather a reference to the dynamic module is provided at the time of compilation and linking. Dynamic Link Libraries (DLL) in Windows and Shared Objects in UNIX are good examples of dynamic libraries. Virtual Memory A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard disk that's set up to emulate the computer's RAM. The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address. Following are the situations, when entire program is not required to be loaded fully in main memory:

- User written error handling routines are used only when an error occurred in the data or computation.
- Certain options and features of a program may be used rarely.
- Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.
- The ability to execute a program that is only partially in memory would counter many benefits.
- Less number of I/O would be needed to load or swap each user program into memory.
- A program would no longer be constrained by the amount of physical memory that is available.
- Each user program could take less physical memory; more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.

Modern microprocessors intended for general-purpose use, a memory management unit, or MMU, is built into the hardware. The MMU's job is to translate virtual addresses into physical addresses. A basic example is given below: Figure: Virtual Memory

Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory. Demand Paging A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance. When a context switch occurs, the operating system does not copy any of the old program's pages out to the disk or any of the new program's pages into the main memory Instead, it just begins executing the new program after loading the first page and fetches that program's pages as they are referenced. Figure 3: Demand Paging While executing a program, if the program references a page which is not available in the main memory because it was swapped out a little ago, the processor treats this invalid memory reference as a page fault and transfers control from the program to the operating system to demand the page back into the memory. Page Replacement Algorithm Page replacement algorithms are the techniques using which an Operating System decides which memory pages to swap out, write to disk when a page of memory needs to be allocated. Paging happens whenever a page fault occurs and a free page cannot be used for allocation purpose accounting to reason that pages are not available or the number of free pages is lower than required pages. The string of memory references is called reference string. Reference strings are generated artificially or by tracing a given system and recording the address of each memory reference. The latter choice produces a large number of data, where we note two things.

- For a given page size, we need to consider only the page number, not the entire address.
- If we have a reference to a page p, then any immediately following references to page p will never cause a page fault. Page p will be in memory after the first reference; the immediately following references will not fault.

First in First out (FIFO) algorithm • Oldest page in main memory is the one which will be selected for replacement.

- Easy to implement, keep a list, replace pages from the tail and add new pages at the head.

Optimal Page algorithm • An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal page-replacement algorithm exists, and has been called OPT or LRU. • Replace the page that will not be used for

the longest period of time. Use the time when a page is to be used. Least Recently Used (LRU) algorithm • Page which has not been used for the longest time in main memory is the one which will be selected for replacement. • Easy to implement, keep a list, replace pages by looking back into time. Page buffering algorithm • To get a process start quickly, keep a pool of free frames. • On page fault, select a page to be replaced. • Write the new page in the frame of free pool, mark the page table and restart the process. • Now write the dirty page out of disk and place the frame holding replaced page in free pool. Least frequently Used (LFU) algorithm • The page with the smallest count is the one which will be selected for replacement. • This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again. Most frequently Used (MFU) algorithm • This algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used. Memory management in unixUNIX is a portable, multi-tasking and multi-user operating system. Memory is an important resource in computer. Memory management is the process of managing the computer memory which consists of primary memory and secondary memory. × The goal for memory management is to keep track of which parts of memory are in use and which parts are not in use, to allocate memory to processes when they need it and deallocate it when they are done. UNIX memory management scheme includes swapping and demand paging UNIX Memory Management Strategies Overlays • Program will be placed into memory during execution. • However, a large program will divide into small pieces and loading the pieces as they needed. → Overlays will replace the new pieces with the program which is unused. • UNIX is using this technique to run a new program by fork the running process which is also known as fork-exec. Swapping• • The process of moving some pages out of main memory and moving others in, is called swapping. • A page fault occurs when the CPU tries to access a page that is not in main memory, thus forcing the CPU to wait for the page to be swapped in. • Since moving data to and from disks takes a significant amount of time, the goal of the memory manager is to minimize the number of page faults Virtual Memory UNIX operating system allows user to fully utilize the physical memory installed in the system as well as part of the hard disk called swap space which have been designated for use by the kernel while the physical memory is insufficient to handle the tasks. Virtual memory managers will create a virtual address space in secondary memory (hard disk) and it will determine the part of address space to be loaded into physical memory at any given time. The benefit of virtual memory relies on separation of logical and physical memory. Conclusion Every operating system has different memory management. UNIX also has their exclusive memory management strategies to manage the memory resource optimally. nearly UNIX: variable partitioning with no virtual memory scheme current implementations of UNIX and Solaris make use of paged virtual memory UNIX is using multiple and variable partitioning so that the memory can be stored and used more flexibly. UNIX also fully utilized the virtual memory (physical memory and swap space) by using demand paging Windows Memory Management Microsoft Windows has its own virtual address space for each 32-bit process, allowing up to 4 gigabytes of memory to be viewed. Each process has 8-terabyte address space on 64-bit Windows. All threads have access to the visible address space of the process. Threads, on the other hand, do not have access to the memory of another process, which protects one process from being damaged by another. Architecture for 32-bit Windows: The automatic configuration of the 32-bit Windows Operating System (OS) allocates 4 GB (232) of accessible memory space to the kernel and user programs equally. With 4 GB physical memory available, the kernel will receive 2 GB and the app memory will receive 2 GB. Kernel-mode address space is shared by all processes, but application mode access space is provided for each user process. Architecture for 64-bit Windows: The automatic configuration of the 64-bit Windows Operating System (OS) allocates up to 16 TB (254) of accessible memory space to the kernel and user programs equally. As 16 TB real memory is available, the kernel will have 8 TB of virtual address (VA) space and user application memory will have 8 TB of VA space. Visible address space in the kernel is allocated for all processes. Each 64-bit functionality gets its place, but each 32-bit system works on a 2 GB (Windows) virtual machine.

storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

**File Structure:** A File Structure should be according to a required format that the operating system can understand.

- A file has a certain defined structure according to its type.
- A text file is a sequence of characters organized into lines.
- A source file is a sequence of procedures and functions.
- An object file is a sequence of bytes organized into blocks that are understandable by the machine.
- When operating system defines different file structures, it also contains the code to support these file structure.

**UNIX, MS-DOS** support minimum number of file structure.

**File Type:** File type refers to the ability of the operating system to distinguish different types of file such as text files source files and binary files etc. Many operating systems support many types of files. Operating system like MS-DOS and UNIX has the following types of files:

1. **Ordinary files:** • These are the files that contain user information.
- These may have text, databases or executable program.
- The user can apply various operations on such files like add, modify, delete or even remove the entire file.

2. **Directory files:** These files contain list of file names and other information related to these files.

3. **Special files:** These files are also known as device files and these files represent physical device like disks, terminals, printers, networks, tape drive etc. Special files are of two types:
  - **Character special files:** Data is handled character by character as in case of terminals or printers.
  - **Block special files:** Data is handled in blocks as in the case of disks and tapes.

**User's and System Programmer's view of File System**

User View: The user view of the computer refers to the interface being used. Such systems are designed for one user to monopolize its resources, to maximize the work that the user is performing. In these cases, the operating system is designed mostly for ease of use, with some attention paid to performance, and none paid to resource utilization.

System View: Operating system can be viewed as a resource allocator also. A computer system consists of many resources like - hardware and software - that must be managed efficiently. The operating system acts as the manager of the resources, decides between conflicting requests, controls execution of programs etc.

**Disk Organization:** A hard disk is a memory storage device which looks like this:

Fig: Hard Disk

The disk is divided into tracks. Each track is further divided into sectors. The point to be noted here is that outer tracks are bigger in size than the inner tracks but they contain the same number of sectors and have equal storage capacity. This is because the storage density is high in sectors of the inner tracks whereas the bits are sparsely arranged in sectors of the outer tracks. Some space of every sector is used for formatting. So, the actual capacity of a sector is less than the given capacity.

Read-Write(R-W) head moves over the rotating hard disk. It is this Read-Write head that performs all the read and write operations on the disk and hence, position of the R-W head is a major concern. To perform a read or write operation on a memory location, we need to place the R-W head over that position.

Some important terms must be noted here:

1. **Seek time** – The time taken by the R-W head to reach the desired track from its current position.
2. **Rotational latency** – Time taken by the sector to come under the R-W head.
3. **Data transfer time** – Time taken to transfer the required amount of data. It depends upon the rotational speed.
4. **Controller time** – The processing time taken by the controller.
5. **Average Access time** – seek time + Average Rotational latency + data transfer time + controller time.

**Magnetic Tape:**

- Appears as same that of music cassettes. It is plastic tape with magnetic coating on it.
- Data is stored in the form of tiny segments of magnetized and demagnetized portions on the surface of the material.
- The magnetized portion of the tape refers to the bit value "1" and demagnetized portion refers "0"

**Tape Organization:**

- Divided into vertical columns (frames) and horizontal rows (channels or tracks).
- Data stored in a string of frames with one character per frame, each frame spans multiple tracks.
- A single bit is stored in each track, one byte per frame. The remaining track will store the parity bit.
- When a byte is written to the tape, the number of 1s in byte is counted; the parity bit is then used to make the number of 1s even or odd.
- When the tape is read again, the parity bit is checked to see if any bit is lost.

• Magnetic tape drives two reels: Supply reel and take-up reel. Both are mounted on the hubs and the tape moves from the supply reel to take-up reel

- The magnetic oxide coated side of the tape of passes directly over the read/write head assembly, thus making contact with the heads.
- As the tape passes directly under the read/write head, the data can be either read and transferred to the primary memory or read from the primary memory and written on to the tape.

Fig : Tape organization

**Different Modules of a File System:**

- The basic file system level works directly with the device drivers in terms of retrieving and storing raw blocks of data, without any consideration for what is in each block. Depending on the system, blocks may be referred to with a single block number or with headsector-cylinder combinations.
- The file organization module knows about files and their logical blocks, and how they map to physical blocks on the disk. In addition to translating from logical to physical blocks, the file organization module also maintains the

list of free blocks, and allocates free blocks to files as needed. • The logical file system deals with all of the metadata associated with a file (UID, GID, mode, dates, etc), i.e. everything about the file except the data itself. This level manages the directory structure and the mapping of file names to file control blocks, FCBs, which contain all of the Meta data as well as block number information for finding the data on the disk. • The layered approach to file systems means that much of the code can be used uniformly for a wide variety of different file systems, and only certain layers need to be file system specific. Common file systems in use include the UNIX file system, UFS, the Berkeley Fast File System, FFS, Windows systems FAT, FAT32, NTFS, CD-ROM systems ISO 9660, and for Linux the extended file systems ext2 and ext3 .

Fig : Layered File System Disk Space Allocation Methods: There are three major methods of storing files on disks: contiguous, linked, and indexed. Contiguous Allocation • Contiguous Allocation requires that all blocks of a file be kept together contiguously. • Performance is very fast, because reading successive blocks of the same file generally requires no movement of the disk heads, or at most one small step to the next adjacent cylinder. • Storage allocation involves the same issues discussed earlier for the allocation of contiguous blocks of memory (first fit, best fit, fragmentation problems, etc.) The distinction is that the high time penalty required for moving the disk heads from spot to spot may now justify the benefits of keeping files contiguously when possible. • Even file systems that do not by default store files contiguously can benefit from certain utilities that compact the disk and make all files contiguous in the process. • Problems can arise when files grow, or if the exact size of a file is unknown at creation time:• Over-estimation of the file's final size increases external fragmentation and wastes disk space. • Under-estimation may require that a file be moved or a process aborted if the file grows beyond its originally allocated space. • If a file grows slowly over a long time period and the total final space must be allocated initially, then a lot of space becomes unusable before the file fills the space. • A variation is to allocate file space in large contiguous chunks, called extents. When a file outgrows its original extent, then an additional one is allocated. (For example an extent may be the size of a complete track or even cylinder, aligned on an appropriate track or cylinder boundary. ) The high-performance files system Veritas uses extents to optimize performance. Fig :

Contiguous allocation of Disk space Linked Allocation: • Disk files can be stored as linked lists, with the expense of the storage space consumed by each link. (Example: A block may be 508 bytes instead of 512.) • Linked allocation involves no external fragmentation, does not require pre-known file sizes, and allows files to grow dynamically at any time. •

Unfortunately linked allocation is only efficient for sequential access files, as random access requires starting at the beginning of the list for each new location access. • Allocating clusters of blocks reduces the space wasted by pointers, at the cost of internal fragmentation. • Another big problem with linked allocation is reliability if a pointer is lost or damaged. Doubly linked lists provide some protection, at the cost of additional overhead and wasted space. Fig : Linked allocation of Disk space The File Allocation Table, FAT, used by DOS is a variation of linked allocation, where all the links are stored in a separate table at the beginning of the disk. The benefit of this approach is that the FAT table can be cached in memory, greatly improving random access speeds. Fig : File-allocation Indexed Allocation: Indexed Allocation combines all of the indexes for accessing each file into a common block (for that file), as opposed to spreading them all over the disk or storing them in a FAT table. Fig : Indexed allocation of Disk space

Directory Structure: • Directory can be defined as the listing of the related files on the disk. The directory may store some or the entire file attributes. • To get the benefit of different file systems on the different operating systems, a hard disk can be divided into the number of partitions of different sizes. The partitions are also called volumes or mini disks. • Each partition must have at least one directory in which, all the files of the partition can be listed. A directory entry is maintained for each file in the directory which stores all the information related to that file. Fig : Directory Structure A directory can be viewed as a file which contains the Meta data of the bunch of files. Every Directory supports a number of common operations on the file: 1. File Creation 2. Search for the file 3. File deletion 4. Renaming the file 5. Traversing Files 6. Listing of files A directory structure can be classified in to single level and hierarchical directory system: Single level directory structure: In this type of directory structure, there is a root directory which has all files. It has a simple architecture and there are no sub directories. Advantage of single level directory system is that it is easy to find a file in the directory. This type of directory system is used in cameras and phones. Fig : Directory System Hierarchical directory structure: In a hierarchical directory structure, files are grouped together to form a sub directory at the top of the hierarchy is the root directory and then there are sub directories which has files. Advantage of hierarchical directory system is that users can be provided access to a sub directory rather than the entire directory. It provides a better structure to file system. Also, managing millions of files is easy with this architecture.

Personal computers use hierarchical directory system for managing files. Fig : Hierarchical Directory System File Protection: When information is stored in a computer system, we want to keep it safe from physical damage (the issue of reliability) and improper access (the issue of protection). Reliability is generally provided by duplicate copies of files. Many computers have systems programs that automatically (or through computer-operator intervention) copy disk files to tape at regular intervals (once per day or week or month) to maintain a copy should a file system be accidentally destroyed. File systems can be damaged by hardware problems (such as errors in reading or writing), power surges or failures, head crashes, dirt, temperature extremes, and vandalism. Files may be deleted accidentally. Bugs in the file-system software can also cause file contents to be lost. Protection can be provided in many ways. For a small single-user system, we might provide protection by physically removing the floppy disks and locking them in a desk drawer or file cabinet. In a multiuser system, however, other mechanisms are needed. System calls when working with files:

- System call OPEN Opening or creating a file can be done using the system call open. The syntax is: #include #include #include int open(const char \*path, int flags,... /\* mode\_t mod \*/);
- System call CREAT A new file can be created by: #include #include #include int creat(const char \*path, mode\_t mod);
- System call READ When we want to read a certain number of bytes starting from the current position in a file, we use the read call. The syntax is: #include ssize\_t read(int fd, void\* buf, size\_t nopt);
- System call WRITE For writing a certain number of bytes into a file starting from the current position we use the write call. Its syntax is: #include ssize\_t write(int fd, const void\* buf, size\_t nopt);
- System call CLOSE For closing a file and thus eliminating the assigned descriptor we use the system call close. #include int close(int fd);
- System call LSEEK To position a pointer that points to the current position in an absolute or relative way can be done by calling the lseek function. Read and write operations are done relative to the current position in the file. The syntax for lseek is: #include #include off\_t lseek(int fd, off\_t offset, int ref);
- System call LINK To link an existing file to another directory (or to the same directory) link can be used. To make such a link in fact means to set a new name or a path to an existing file. The link system call creates a hard link. Creating symbolic links can be done using symlink system call. The syntax of link is: #include int link(const char\* oldpath, const char\* newpath); int symlink(const char\* oldpath, const char\* newpath);

Disk Scheduling Algorithms: I/O request issues a system call to the OS. If desired disk drive or controller is available, request is served immediately. If busy, new request for service will be placed in the queue of pending requests. When one request is completed, the OS has to choose which pending request to service next.

FCFS Scheduling: It is the simplest Disk Scheduling algorithm. It services the IO requests in the order in which they arrive. There is no starvation in this algorithm, every request is serviced. Example: Consider the following disk request sequence for a disk with 200 tracks: 23, 89, 132, 42, 187 with disk head initially at 100 If the requests for cylinders 23 and 42 could be serviced together, total head movement could be decreased substantially.

SSTF Scheduling: Like SJF, select the disk I/O request that requires the least movement of the disk arm from its current position, regardless of direction reduces total seek time compared to FCFS. Its disadvantage is that starvation is possible; stay in one area of the disk if very busy switching directions slow things down not the most optimal.

Fig : SSTF Scheduling SCAN: It is also called as Elevator Algorithm. In this algorithm, the disk arm moves into a particular direction till the end, satisfying all the requests coming in its path, and then it turns backend moves in the reverse direction satisfying requests coming in its path. It works in the way an elevator works, elevator moves in a direction completely till the last floor of that direction and then turns back.

Fig : SCAN Scheduling C-SCAN In C-SCAN algorithm, the arm of the disk moves in a particular direction servicing requests until it reaches the last cylinder, then it jumps to the last cylinder of the opposite direction without servicing any request then it turns back and start moving in that direction servicing the remaining requests.

Fig 2: C-SCAN Scheduling LOOK: Like SCAN but stops moving inwards (or outwards) when no more requests in that direction exist.

Fig : LOOK Scheduling Compared to SCAN, LOOK saves going from 23 to 0 and then back. Most efficient for this sequence of requests.

File Systems In Unix: Everything in Unix is considered to be a file, including physical devices such as DVD-ROMs, USB devices, and floppy drives.

- Unix uses a hierarchical file system structure, much like an upside-down tree, with root (/) at the base of the file system and all other directories spreading from there.
- A Unix filesystem is a collection of files and directories that has the following properties :-
  - o It has a root directory (/) that contains other files and directories.
  - o Each file or directory is uniquely identified by its name, the directory in which it resides, and a unique identifier, typically called an inode.
  - o It is self-contained. There are no dependencies between one filesystem and another.

Following are the directories that exist on the major versions of Unix :

Sr.No.	Directory & Description
1	This is the root directory which should contain only the directories needed at the top level of the file structure
2	/bin This is where the executable files are located. These files are

available to all users 3 /dev These are device drivers 4 /etc Supervisor directory commands, configuration files, disk configuration files, valid user lists, groups, ethernet, hosts, where to send critical messages 5 /lib Contains shared library files and sometimes other kernel-related files 6 /boot Contains files for booting the system 7 /home Contains the home directory for users and other accounts 8 /mnt Used to mount other temporary file systems, such as cdrom and floppy for the CDROM drive and floppy diskette drive, respectively 9 /proc Contains all processes marked as a file by process number or other information that is dynamic to the system 10 /tmp Holds temporary files used between system boots 11 /usr Used for miscellaneous purposes, and can be used by many users. Includes administrative commands, shared files, library files, and others 12 /var Typically contains variable-length files such as log and print files and any other type of file that may contain a variable amount of data 13 /sbin Contains binary (executable) files, usually for system administration. For example, fdisk and ifconfig utilities 14 /kernel Contains kernel files File System In Windows: For Windows operating systems, FAT16, FAT32, exFAT and NTFS are four types of file systems most widely adopted by users. Here are some brief introductions about the different features of these four file systems.

- FAT16, also known as File Allocation Table 16, was created for old systems like MS-DOS, Windows 95. FAT16 uses a 16-bit binary number to keep clusters, and this is why it is called FAT16. If a file exceeds the capacity of singular sector of a FAT16 partition, it would take more space than the size of the file itself. FAT16 is outdated because it has a big weakness: it supports partition with a capacity of no more than 2GB. In today's world, it is really difficult to find a disk under 2GB, and a 2GB disk cannot meet the demand of most users.
- FAT32 file system was firstly introduced by Microsoft in 1996 to be taken as the advanced edition of FAT16. It uses 32-bit binary number to hold clusters, limiting the partition or volume size up to 2TB with a sector size of 512 bytes. And it works with most of Windows, even Mac, and game consoles. But it also has a size limit. It can only support a maximum of 32GB partition, and a maximum of 4GB single file. If you copy a file like a movie that is usually larger than 4GB to a FAT32 hard drive, you will be told that the file is too large for the destination disk.
- NTFS stands for New Technology File System. It is a great improvement in many aspects over FAT32 and FAT16 file system. It uses B-tree structure that allows users to use hard disk larger than 2TB and provides much fast speed, making itself a popular choice among an increasing number of Windows users. Besides, NTFS is a journaling file system that resists data loss and damage. It has additional permission settings that can encrypt files to control access of files and folders. However, NTFS file system also has its weakness: it is only compatible with Windows 2000 and later versions. And NTFS doesn't support PS4, android smartphone, camera, and other devices. Mac OS X can only read NTFS partition.
- exFAT is a new file system launched in 2006, which was usually used for flash memory like USB flash drive, SD card. It can be seen as a lightweight of FAT32 without additional features of NTFS. exFAT makes up for the disadvantages of FAT32 in file size and the compatibility of NTFS, which means it can store a file larger than 4GB and can both work in Windows and Mac OS.

Chameli Devi Group of Institutions Department of Artificial intelligence and Data Science Subject Notes AD405 Operating Systems UNIT-V UNIT 5: Input/Output Management: Principles and Programming, Input/Output Problems, Different I/O operations: Program Controlled, Interrupt Driven, Concurrent I/O, Asynchronous Operations, Logical structure of I/O function, I/O Buffering, Kernel I/o Subsystem .Introduction to Network, Distributed and Multiprocessor Operating Systems. I/O management in UNIX & Window Input/Output: Principles and Programming One of the important jobs of an Operating System is to manage various I/O devices including mouse, keyboards, touch pad, disk drives, display adapters, USB devices, Bit-mapped screen, LED, Analog-to-digital converter, On/off switch, network connections, audio I/O, printers etc. An I/O system is required to take an application I/O request and send it to the physical device, then take whatever response comes back from the device and send it to the application. I/O devices can be divided into two categories –

- Block devices: A block device is one with which the driver communicates by sending entire blocks of data. For example, Hard disks, USB cameras, Disk-On-Key etc.
- Character devices: A character device is one with which the driver communicates by sending and receiving single characters (bytes, octets). For example, serial ports, parallel ports, sound cards etc.

Device Controllers Device drivers are software modules that can be plugged into an OS to handle a particular device. Operating System takes help from device drivers to handle all I/O devices. The Device Controller works like an interface between a device and a device driver. I/O units (Keyboard, mouse, printer, etc.) typically consist of a mechanical

component and an electronic component where electronic component is called the device controller. There is always a device controller and a device driver for each device to communicate with the Operating Systems. A device controller may be able to handle multiple devices. As an interface its main task is to convert serial bit stream to block of bytes, perform error correction as necessary. Any device connected to the computer is connected by a plug and socket, and the socket is connected to a device controller. Following is a model for connecting the CPU, memory, controllers, and I/O devices where CPU and device controllers all use a common bus for communication.

**Input /Output Problems:** When we analyze device communication, we notice that communication is required at the following three levels:

- The need for a human to input information and receive output from a computer.
- The need for a device to input information and receive output from a computer.
- The need for computers to communicate (receive/send information) over network.

The first kind of IO devices operate at rates good for humans to interact. These may be character-oriented devices like a keyboard or an event-generating device like a mouse. Usually, human input using a key board will be a few key depressions at a time. This means that the communication is rarely more than a few bytes. Also, the mouse events can be encoded by a small amount of information (just a few bytes). Even though a human input is very small, it is stipulated that it is very important, and therefore requires an immediate response from the system. A communication which attempts to draw attention often requires the use of an interrupt mechanism or a programmed data mode of operation.

The second kind of IO requirement arises from devices which have a very high character density such as tapes and disks. With these characteristics, it is not possible to regulate communication with devices on a character by character basis. The information transfer, therefore, is regulated in blocks of information. Additionally, sometimes this may require some kind of format control to structure the information to suit the device and/or data characteristics. For instance, a disk drive differs from a line printer or an image scanner. For each of these devices, the format and structure of information is different. It should be observed that the rate at which a device may provide data and the rates at which an end application may consume it may be considerably different. In spite of these differences, the OS should provide uniform and easy to use IO mechanisms. Usually, this is done by providing a buffer. The OS manages this buffer so as to be able to comply with the requirements of both the producer and consumer of data.

The third kind of IO requirements emanate from the need to negotiate system IO with the communications infrastructure. The system should be able to manage communications traffic across the network. This form of IO facilitates access to internet resources to support e-mail, filetransfer amongst machines or Web applications. Additionally, now we have a large variety of options available as access devices. These access devices may be in the form of Personal Digital Assistant (PDA), or mobile phones which have infrared or wireless enabled communications. This rapidly evolving technology makes these forms of communications very challenging.

**Different Input/Output Operations:**

- Programmed I/O:** In program-controlled I/O, the processor program controls the complete data transfer. So only when an I/O transfer instruction is executed, the transfer could take place. It is required to check that device is ready/not for the data transfer in most cases. Usually, the transfer is to & from a CPU register & peripheral. Here, CPU constantly monitors the peripheral. Here, until the I/O unit indicates that it is ready for transfer, the CPU wait & stays in a loop. It is timeconsuming as it keeps the CPU busy needlessly.
- Interrupt—Driven I/O:** To overcome the disadvantage of Programmed I/O,i.e., keeping CPU busy needlessly, Interrupt — Driven I/O is used. In this approach, when a peripheral sends an interrupt signal to the CPU whenever it is ready to transfer data. This indicates that the I/O data transfer is initiated by the external I/O device. The processor stops the execution of the current program & transfers the control to interrupt the service routine when interrupted. The interrupt service routine then performs the data transfer. After the completion of data transfer, it returns control to the main program to the point it was interrupted.

**Concurrent I/O:**

- In AIX operating systems, you can use concurrent I/O in addition to direct I/O for chunks that use cooked files. Concurrent I/O can improve performance, because it allows multiple reads and writes to a file to occur concurrently, without the usual serialization of noncompeting read and write operations.
- Concurrent I/O can be especially beneficial when you have data in a single chunk file striped across multiple disks.
- Concurrent I/O, which you enable by setting the DIRECT\_IO configuration parameter to 2, includes the benefit of avoiding file system buffering and is subject to the same limitations and use of KAIO as occurs if you use direct I/O without concurrent I/O. Thus, when concurrent I/O is enabled, you get both un-buffered I/O and concurrent I/O.

**Asynchronous Operations:**

- **Synchronous I/O-** In this scheme CPU execution waits while I/O proceeds
- **Asynchronous I/O-** I/O proceeds concurrently with CPU execution

**Logical Structure of the I/O function:**

- Logical I/O-- open, close, read, write.
- Device I/O-- buffering, controller commands.
- Scheduling and Control-- queuing and scheduling, interrupts handled here.
- Directory management--

symbolic names converted to pointers, create, delete, rename. • File system-- the logical operations on the files, open, close, read, write, locate, access rights. • Physical organization-- mapping of the file blocks to the device blocks. I/O Buffering: Stream oriented I/O -- data is transferred on a byte-by-byte basis; not fixed block oriented; may be line oriented Block oriented I/O -- a block of data is transferred in one operation, e.g., disk sector, tape block. Rather than waiting for an I/O operation to be initiated and then for it to complete, have the system pre-read the input stream. Single Buffering: The process gets a block of input upon an input request and then can proceed further with input operations without waiting for the data. On output the process collects a block of output before the data is transferred Double Buffering: The process can be processing one block while the system is transferring another Circular Buffering: A generalization of the double buffering idea. Kernel I/O Subsystem: The kernel provides many services related to I/O. Several services such as scheduling, caching, spooling, device reservation, and error handling – are provided by the kernel, its I/O subsystem built on the hardware and device-driver infrastructure. The I/O subsystem is also responsible for protecting itself from errant processes and malicious users. Importance of Kernel I/O Subsystem: The Kernel I/O Subsystem is an essential part of any modern Operating System. It provides a unified and consistent interface to the I/O devices, which enables the user applications to access them without knowing the details of the underlying hardware. The Kernel I/O Subsystem also manages the concurrency and synchronization issues that arise when multiple applications try to access the same device simultaneously. Advantages of Kernel I/O Subsystem: Device Independence: The Kernel I/O Subsystem provides device independence to the user applications. It abstracts the hardware details and provides a unified interface to the devices. Efficient Resource Management: The Kernel I/O Subsystem provides efficient resource management for the I/O devices. It manages the I/O requests and schedules them in a way that optimizes the usage of the available resources Concurrency Management: The Kernel I/O Subsystem manages the concurrency issues that arise when multiple applications try to access the same device simultaneously. It ensures that the applications get exclusive access to the device when needed and allows multiple applications to share the device when appropriate. Disadvantages of Kernel I/O Subsystem: Complex Implementation: The Kernel I/O Subsystem is a complex software component that requires a lot of resources to implement and maintain. Any issues with the Kernel I/O Subsystem can affect the performance and stability of the entire system. Security Risks: The Kernel I/O Subsystem can pose security risks to the system if not implemented correctly. Attackers can exploit vulnerabilities in the Kernel I/O Subsystem to gain unauthorized access to the system or cause a denial-of-service attack. Network operating System: Network Operating System is one of the important type of operating system. Network Operating System runs on a server and gives the server the capability to manage data, users, groups, security, applications, and other networking functions. The basic purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks. Some examples of network operating systems include Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BS Types of Network Operating System: Network operating systems can be specialized serve as:

- Peer To Peer System
- Client-Server System

Peer To Peer Network Operating System: Peer To Peer networks are the network resources in which each system has the same capabilities and responsibilities, i.e., none of the systems in this architecture is superior to the others in terms of functionality. There is no master-slave relationship among the systems, i.e., every node is equal in a Peer To Peer Network Operating System. All the nodes at the Network have an equal relationship with others and have a similar type of software that helps the sharing of resources. Client-Server Network Operating System: In Client-Server systems, there are two broad categories of systems:

- The server is called the backend.
- A client called as frontend.

Client-Server Network Operating System is a server-based Network in which storage and processing workload is shared among clients and servers. The client requests offerings which include printing and document storage, and servers satisfy their requests. Normally all community offerings like digital mail, printing are routed through the server Advantages:

- Centralized servers are highly stable.
- Security is server managed.
- Upgradation of new technologies and hardware can be easily integrated into the system.
- It is possible to remote access to servers from different locations and types of systems.

Disadvantages:

- High cost of buying and running a server.
- Dependency on a central location for most operations.
- Regular maintenance and updates are required.

Distributed Operating System: A distributed operating system (DOS) is an essential type of operating system. Distributed systems use many central processors to serve multiple real-time applications and users. As a result, data processing jobs are distributed between the processors. It connects multiple computers via a single communication channel. Furthermore, each of these systems has its own processor and memory.

Additionally, these CPUs communicate via high-speed buses or telephone lines. Individual systems that communicate via a single channel are regarded as a single entity. They're also known as loosely coupled systems. This operating system consists of numerous computers, nodes, and sites joined together via LAN/WAN lines. It enables the distribution of full systems on a couple of center processors, and it supports many real-time products and different users. Distributed operating systems can share their computing resources and I/O files while providing users with virtual machine abstraction.

**Types of Distributed Operating System:** There are various types of Distributed Operating systems. Some of them are as follows:

1. Client-Server Systems
2. Peer-to-Peer Systems
3. Middleware
4. Three-tier
5. N-tier

**Client-Server System:** This type of system requires the client to request a resource, after which the server gives the requested resource. When a client connects to a server, the server may serve multiple clients at the same time. Client-Server Systems are also referred to as "Tightly Coupled Operating Systems". This system is primarily intended for multiprocessors and homogenous multicomputer. Client-Server Systems function as a centralized server since they approve all requests issued by client systems.

**Peer-to-Peer System:** The nodes play an important role in this system. The task is evenly distributed among the nodes. Additionally, these nodes can share data and resources as needed. Once again, they require a network to connect. The Peer-to-Peer System is known as a "Loosely Coupled System". This concept is used in computer network applications since they contain a large number of processors that do not share memory or clocks. Each processor has its own local memory, and they interact with one another via a variety of communication methods like telephone lines or high-speed buses.

**Middleware:** Middleware enables the interoperability of all applications running on different operating systems. Those programs are capable of transferring all data to one other by using these services.

**Three-Tier:** The information about the client is saved in the intermediate tier rather than in the client, which simplifies development. This type of architecture is most commonly used in online applications.

**N-Tier:** When a server or application has to transmit requests to other enterprise services on the network, n-tier systems are used.

**Advantages of Distributed Operating System:**

- Failure of one will not affect the other network communication, as all systems are independent from each other
- Electronic mail increases the data exchange speed
- Since resources are being shared, computation is highly fast and durable
- Load on host computer reduces
- These systems are easily scalable as many systems can be easily added to the network
- Delay in data processing reduces

**Disadvantages of Distributed Operating System:**

- Failure of the main network will stop the entire communication
- To establish distributed systems the language which are used are not well defined yet
- These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet

**Examples of Distributed Operating System are-** LOCUS etc

**Multiprocessor Operating System:** It refers to the use of two or more central processing units (CPU) within a single computer system. These multiple CPUs are in a close communication sharing the computer bus, memory and other peripheral devices. These systems are referred as tightly coupled systems. These types of systems are used when very high speed is required to process a large volume of data. These systems are generally used in environment like satellite control, weather forecasting etc. For Example: UNIX Operating system is one of the most widely used multiprocessing systems. The basic organization of multiprocessing system is shown in fig. The different fields of Multiprocessor Operating Systems used are as follows –

- **Asymmetric Multiprocessor** – Every processor is given seeded tasks in this operating system, and the master processor has the power for running the entire system. In the course, it uses the master-slave relationship.
- **Symmetric Multiprocessor** – In this system, every processor owns a similar copy of the OS, and they can make communication in between one another. All processors are connected with peering relationship nature, meaning it won't be using master & slave relation.
- **Shared memory Multiprocessor** – As the name indicates, each central processing unit contains distributable common memory.
- **Uniform Memory Access Multiprocessor (UMA)** – In this system, it allows accessing all memory at a consistent speed rate for all processors.
- **Distributed memory Multiprocessor** – A computer system consisting of a range of processors, each with its own local memory, connected through a network, which means all the kinds of processors consist of their own private memory.
- **NUMA Multiprocessor** – The abbreviation of NUMA is Non-Uniform Memory Access Multiprocessor. It entails some areas of the memory for approaching at a swift rate and the remaining parts of memory are used for other tasks.

The Advantages of Multiprocessor Systems are as follows:

- If there are multiple processors working at the same time, more processes can be executed parallel at the same time. Therefore the throughput of the system will increase.
- Multiprocessor systems are more reliable. Due to the fact that there are more than one processor, in case of failure of any one processor will not make the system come to a halt. Although the system will become slow if it happens but still it will work.
- Electricity consumption of a multiprocessor system is less

than the single processor system. This is because, in single processor systems, many processes have to be executed by only one processor so there is a lot of load on it. But in case of multiple processor systems, there are many processors to execute the processes so the load on each processor will be comparatively less so electricity consumed will also be less.

**Disadvantages of Multiprocessing Operating System:**

- Operating system of multiprocessing is more complex and sophisticated as it takes care of multiple CPUs at the same time.

**Input Output Management in Unix:**

Most Unix system commands take input from your terminal and send the resulting output back to your terminal. A command normally reads its input from the standard input, which happens to be your terminal by default. Similarly, a command normally writes its output to standard output, which is again your terminal by default. Following is a complete list of commands which we can use for redirection:

- Sr.No. Command & Description
- 1 pgm > file Output of pgm is redirected to file
- 2 pgm < file Program pgm reads its input from file
- 3 pgm >> file Output of pgm is appended to file
- 4 n > file Output from stream with descriptor n redirected to file
- 5 n >> file Output from stream with descriptor n appended to file
- 6 n >& m Merges output from stream n with stream m
- 7 n <& m Merges input from stream n with stream m
- 8 << tag Standard input comes from here through next tag at the start of line
- 9 | Takes output from one program, or process, and sends it to another

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Name: Machine Learning Subject Code: AD 502 Subject Notes Syllabus: Introduction to machine learning, Machine learning life cycle, Types of Machine Learning System (supervised and unsupervised learning, Batch and online learning, Instance-Based and Model based Learning), scope and limitations, Challenges of Machine learning, data visualization, hypothesis function and testing, data preprocessing, data augmentation, normalizing data sets, Relation between AI (Artificial Intelligence), ML (Machine Learning), DL (Deep Learning) and DS (Data Science).

---

#### Course

**Outcome (CO1):** To understand the basic theory underlying machine learning. Unit-I Machine Learning Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for building mathematical models and making predictions using historical data or information. Currently, it is being used for various tasks such as image recognition, speech recognition, email filtering, Facebook auto-tagging, recommender system, and many more. In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does. Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed. With the help of sample historical data, which is known as training data, machine learning algorithms build a mathematical model that helps in making predictions or decisions without being explicitly programmed. Machine learning brings computer science and statistics together for creating predictive models. Machine learning constructs or uses the algorithms that learn from historical data. The more we will provide the information, the higher will be the performance. Machine Learning is said as a subset of artificial intelligence that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by Arthur Samuel in 1959. Machine learning life cycle Machine learning has given the computer systems the abilities to automatically learn without being explicitly programmed. But how does a machine learning system work? So, it can be described using the life cycle of machine learning. Machine learning life cycle is a cyclic process to build an efficient machine learning project. The main purpose of the life cycle is to find a solution to the problem or project. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Machine learning life cycle involves seven major steps, which are given below:

- Gathering Data
- Data preparation
- Data Wrangling
- Analyse Data
- Train the model
- Test the model
- Deployment

**Fig-1.1 ML Life-Cycle**

The most important thing in the complete process is to understand the problem and to know the purpose of the problem. Therefore, before starting the life cycle, we need to understand the problem because the good result depends on the better understanding of the problem. In the complete life cycle process, to solve a problem, we create a machine learning system called "model", and this model is created by providing "training". But to train a model, we need data, hence, life cycle starts by collecting data.

**Gathering Data:** Data Gathering is the first step of the machine learning life cycle. The goal of this step is to identify and obtain all data related problems. In this step, we need to identify the different data sources, as data can be

collected from various sources such as files, database, internet, or mobile devices. It is one of the most important steps of the life cycle. The quantity and quality of the collected data will determine the efficiency of the output. The more will be the data, the more accurate will be the prediction. This step includes the below tasks:

- Identify various data sources
- Collect data
- Integrate the data obtained from different sources

By performing the above task, we get a coherent set of data, also called as a dataset. It will be used in further steps.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science

Data preparation

After collecting the data, we need to prepare it for further steps. Data preparation is a step where we put our data into a suitable place and prepare it to use in our machine learning training. In this step, first, we put all data together, and then randomize the ordering of data. This step can be further divided into two processes:

- Data exploration: It is used to understand the nature of data that we have to work with. We need to understand the characteristics, format, and quality of data. A better understanding of data leads to an effective outcome. In this, we find Correlations, general trends, and outliers.
- Data pre-processing: Now the next step is preprocessing of data for its analysis.

Data Wrangling

Data wrangling is the process of cleaning and converting raw data into a useable format. It is the process of cleaning the data, selecting the variable to use, and transforming the data in a proper format to make it more suitable for analysis in the next step. It is one of the most important steps of the complete process. Cleaning of data is required to address the quality issues. It is not necessary that data we have collected is always of our use as some of the data may not be useful. In real-world applications, collected data may have various issues, including:

- Missing Values
- Duplicate data
- Invalid data
- Noise

So, we use various filtering techniques to clean the data. It is mandatory to detect and remove the above issues because it can negatively affect the quality of the outcome.

Data Analysis

Now the cleaned and prepared data is passed on to the analysis step. This step involves:

- Selection of analytical techniques
- Building models
- Review the result

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science

The aim of this step is to build a machine learning model to analyze the data using various analytical techniques and review the outcome. It starts with the determination of the type of the problems, where we select the machine learning techniques such as Classification, Regression, Cluster analysis, Association, etc. then build the model using prepared data, and evaluate the model.

Train Model

Now the next step is to train the model, in this step we train our model to improve its performance for better outcome of the problem. We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and, features.

Test Model

Once our machine learning model has been trained on a given dataset, then we test the model. In this step, we check for the accuracy of our model by providing a test dataset to it. Testing the model determines the percentage accuracy of the model as per the requirement of project or problem.

Deployment

The last step of machine learning life cycle is deployment, where we deploy the model in the real-world system. If the above-prepared model is producing an accurate result as per our requirement with acceptable speed, then we deploy the model in the real system. But before deploying the project, we will check whether it is improving its performance using available data or not. The deployment phase is similar to making the final report for a project.

Types of Machine Learning System

Based on the methods and way of learning, machine learning is divided into mainly four types, which are:

1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Semi-Supervised Machine Learning
4. Reinforcement Learning

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science

Fig-1.2 Type of ML

### Supervised Machine Learning

As its name suggests, Supervised machine learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More precisely, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

Example Suppose we have an input dataset of cats and dog images. So, first, we will provide the training to the machine to understand the images, such as the shape & size of the tail of cat and dog, Shape of eyes, colour, height (dogs are taller, cats are smaller), etc. After completion of training, we input the picture of a cat and ask the machine to identify the object and predict the output. Now, the machine is well trained, so it will check all the features of the object, such as height, shape, colour, eyes, ears, tail, etc., and find that it's a cat. So, it will put it in the Cat category. This is the process of how the machine identifies the objects in Supervised Learning. The main goal of the supervised learning technique is to map the input variable(x) with the output variable(y). Some real-world applications of supervised learning are Risk Assessment, Fraud Detection, Spam filtering, etc.

Categories of Supervised Machine Learning

Supervised machine learning can be classified into two types of problems, which are given below:

- o Classification
- o Regression

Chameli Devi

Group of Institutions Department of Artificial Intelligence & Data Science Unsupervised Machine Learning Unsupervised learning is different from the Supervised learning technique; as its name suggests, there is no need for supervision. It means, in unsupervised machine learning, the machine is trained using the unlabeled dataset, and the machine predicts the output without any supervision. In unsupervised learning, the models are trained with the data that is neither classified nor labelled, and the model acts on that data without any supervision. The main aim of the unsupervised learning algorithm is to group or categories the unsorted dataset according to the similarities, patterns, and differences. Machines are instructed to find the hidden patterns from the input dataset. Example suppose there is a basket of fruit images, and we input it into the machine learning model. The images are totally unknown to the model, and the task of the machine is to find the patterns and categories of the objects. So, now the machine will discover its patterns and differences, such as colour difference, shape difference, and predict the output when it is tested with the test dataset.

Categories of Unsupervised Machine Learning Unsupervised Learning can be further classified into two types, which are given below:

- o Clustering
- o Association

Semi-Supervised Learning Semi-Supervised learning is a type of Machine Learning algorithm that lies between Supervised and Unsupervised machine learning. It represents the intermediate ground between Supervised (With Labelled training data) and Unsupervised learning (with no labelled training data) algorithms and uses the combination of labelled and unlabeled datasets during the training period.

Reinforcement Learning Reinforcement learning works on a feedback-based process, in which an AI agent (A software component) automatically explores its surroundings by hitting & trail, taking action, learning from experiences, and improving its performance. Agent gets rewarded for each good action and gets punished for each bad action; hence the goal of reinforcement learning agent is to maximize the rewards.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science In reinforcement learning, there is no labelled data like supervised learning, and agents learn from their experiences only. The reinforcement learning process is similar to a human being; for example, a child learns various things by experiences in his day-to-day life. An example of reinforcement learning is to play a game, where the Game is the environment, moves of an agent at each step define states, and the goal of the agent is to get a high score. Agent receives feedback in terms of punishment and rewards. Due to its way of working, reinforcement learning is employed in different fields such as Game theory, Operation Research, Information theory, multi-agent systems.

Batch and online learning Batch or offline learning Batch learning represents the training of machine learning models in a batch manner. In other words, batch learning represents the training of the models at regular intervals such as weekly, bi-weekly, monthly, quarterly, etc. In batch learning, the system is not capable of learning incrementally. The models must be trained using all the available data every single time. The data gets accumulated over a period of time. The models then get trained with the accumulated data from time to time at periodic intervals. This model training takes a lot of time and computing resources. Hence, it is typically done offline. After the models are trained, they are launched into production and they run without learning anymore. Batch learning is also called offline learning. The models trained using batch or offline learning are moved into production only at regular intervals based on the performance of models trained with new data.

Online Learning In online learning, the training happens in an incremental manner by continuously feeding data as it arrives or in a small group / mini batches. Each learning step is fast and cheap, so the system can learn about new data on the fly, as it arrives. Online learning is great for machine learning systems that receive data as a continuous flow (e.g., stock prices) and need to adapt to change rapidly or autonomously. It is also a good option if you have limited computing resources: once an online learning system has learned about new data instances, it does not need them anymore, so you can discard them (unless you want to be able to roll back to a previous state and “replay” the data) or move the data to another form of storage (warm or cold storage) if you are using the data lake. This can save a huge amount of space and cost.

Instance-Based and Model based Learning Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Instance-Based Learning The Machine Learning systems which are categorized as instance-based learning are the systems that learn the training examples by heart and then generalizes to new instances based on some similarity measure. It is called instance-based because it builds the hypotheses from the training instances. It is also known as memory-based learning or lazy-learning (because they delay processing until a new instance must be classified).

Model based Learning Model-based learning (also known as structure-based or eager learning) takes a different approach by constructing models from the training data that can generalize better than instance-based methods. This involves using algorithms like linear regression, logistic regression, random forest, etc. trees to create an underlying model from which predictions can be made for new data points.

scope and limitations Future Scope of Machine Learning The scope of Machine Learning is not limited to the

investment sector. Rather, it is expanding across all fields such as banking and finance, information technology, media & entertainment, gaming, and the automotive industry. As the Machine Learning scope is very high, there are some areas where researchers are working toward revolutionizing the world for the future. Automotive Industry Robotics Quantum Computing Computer Vision Limitations of ML Nothing is perfect in the world. Machine Learning has some serious limitations, which are bigger than human errors. 1. Data Acquisition 2. Time and Resources 3. Results Interpretations 4. High Error Chances Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science 5. Social Changes 6. Elimination of Human Interface 7. Changing Nature of Jobs 8. Highly Expensive 9. Privacy Concern 10. Research and Innovations Challenges of Machine learning 1. Poor Quality of Data 2. Underfitting of Training Data 3. Overfitting of Training Data 4. Machine Learning is a Complex Process 5. Lack of Training Data 6. Slow Implementation 7. Imperfections in the Algorithm When Data Grows Data visualization Data visualization is a crucial aspect of machine learning that enables analysts to understand and make sense of data patterns, relationships, and trends. Through data visualization, insights and patterns in data can be easily interpreted and communicated to a wider audience, making it a critical component of machine learning. 1. Line Charts 2. Scatter Plots 3. Bar Charts 4. Heat Maps 5. Tree Maps 6. Box Plots Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Hypothesis Testing Hypothesis testing is a statistical method that is used in making a statistical decision using experimental data. Hypothesis testing is basically an assumption that we make about a population parameter. It evaluates two mutually exclusive statements about a population to determine which statement is best supported by the sample data. Need for Hypothesis Testing Hypothesis testing is an important procedure in statistics. Hypothesis testing evaluates two mutually exclusive population statements to determine which statement is most supported by sample data. When we say that the findings are statistically significant, it is thanks to hypothesis testing. Parameters of hypothesis testing Null hypothesis(H0): In statistics, the null hypothesis is a general given statement or default position that there is no relationship between two measured cases or no relationship among groups. In other words, it is a basic assumption or made based on the problem knowledge. Example: A company production is = 50 units/per day etc. Alternative hypothesis(H1): The alternative hypothesis is the hypothesis used in hypothesis testing that is contrary to the null hypothesis. Example: A company's production is not equal to 50 units/per day etc. Level of significance: It refers to the degree of significance in which we accept or reject the null hypothesis. 100% accuracy is not possible for accepting a hypothesis, so we, therefore, select a level of significance that is usually 5%. This is normally denoted with  $\alpha$  and generally, it is 0.05 or 5%, which means your output should be 95% confident to give a similar kind of result in each sample. P-value : The P value, or calculated probability, is the probability of finding the observed/extreme results when the null hypothesis(H0) of a study-given problem is true. If your P-value is less than the chosen significance level then you reject the null hypothesis i.e. accept that your sample claims to support the alternative hypothesis. Data pre-processing A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science It involves below steps: o Getting the dataset o Importing libraries o Importing datasets o Finding Missing Data o Encoding Categorical Data o Splitting dataset into training and test set o Feature scaling Data Augmentation Data augmentation is the process of modifying, or "augmenting" a dataset with additional data. This additional data can be anything from images to text, and its use in machine learning algorithms helps improve their performance. For example, say we wanted to build a model to classify dog breeds, and we have a lot of images of most breeds, except for pugs. As a result, the model wouldn't be able to classify pugs well. We could augment the data by adding some (real or fake) images of pugs, or by multiplying our existing pug images (e.g. by replicating and distorting them to make them artificially unique). Data augmentation is crucial for many AI applications, as accuracy increases with the amount of training data. In fact, research studies have found that basic data augmentation can greatly improve accuracy on image tasks, such as classification and segmentation. Further, large neural networks, or deep learning models, need a huge amount of data, so they benefit even more from data augmentation techniques. Normalizing Data Sets Normalization is a scaling technique in Machine Learning applied during data preparation to change the values of numeric columns in the dataset to use a common scale. It is not necessary for all datasets in a model. It is required only when features of machine learning models have different ranges. Mathematically, we can calculate normalization with the below formula:  $X_n = (X - X_{\text{minimum}}) / (X_{\text{maximum}} - X_{\text{minimum}})$  o  $X_n$  = Value of Normalization o  $X_{\text{maximum}}$  = Maximum

value of a feature o  $X_{\min}$  = Minimum value of a feature Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Normalization techniques in Machine Learning Although there are so many feature normalization techniques in Machine Learning, few of them are most frequently used. These are as follows: Min-Max Scaling: This technique is also referred to as scaling. The Min-Max scaling method helps the dataset to shift and rescale the values of their attributes, so they end up ranging between 0 and 1. Standardization Scaling: Standardization scaling is also known as Z-score normalization, in which values are centered around the mean with a unit standard deviation, which means the attribute becomes zero and the resultant distribution has a unit standard deviation. Mathematically, we can calculate the standardization by subtracting the feature value from the mean and dividing it by standard deviation. Hence, standardization can be expressed as follows: Relation Between AI , ML , DL and DS AI: The Dream of Intelligent Machines Artificial Intelligence, or AI, is a field of computer science that aims to create intelligent machines capable of performing tasks that typically require human intelligence. It encompasses a wide range of techniques that enable computers to mimic human cognitive abilities like learning, problem-solving, reasoning, and decisionmaking. ML: The Essence of AI Machine Learning is a subset of AI that focuses on designing algorithms and statistical models that allow machines to learn from data without being explicitly programmed. In other words, instead of hand-coding specific instructions, we let the machine learn patterns and relationships from the data to make predictions or decisions. DL: Unleashing the Power of Neural Networks Deep Learning is a subfield of ML that has gained tremendous popularity in recent years. It is inspired by the structure and function of the human brain and involves training artificial neural networks with vast amounts of data. Deep learning has revolutionized AI by achieving remarkable breakthroughs in image recognition, natural language processing, and much more. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science DS: The Art of Extracting Insights from Data Data Science is a multidisciplinary field that combines AI, ML, and other techniques to extract valuable insights and knowledge from data. It involves data collection, cleaning, analysis, and visualization to uncover hidden patterns and trends. Data scientists play a crucial role in guiding decision-making processes in various industries. The Interplay between AI, ML, DL, and DS: AI serves as the overarching concept that aims to create intelligent systems. Within AI, ML is a powerful tool that enables machines to learn from data and make decisions. DL, as a subset of ML, leverages neural networks to achieve deeper levels of understanding and is particularly effective in complex tasks like image and speech recognition. Data Science acts as the bridge that connects AI, ML, and DL together. Without quality data, machine learning models, and deep learning algorithms would be ineffective. Data science plays a pivotal role in curating, processing, and preparing data for the learning algorithms to work their magic.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Name: Machine Learning Subject Code: AD 502 Subject Notes Syllabus: Introduction to machine learning, Machine learning life cycle, Types of Machine Learning System (supervised and unsupervised learning, Batch and online learning, Instance-Based and Model based Learning), scope and limitations, Challenges of Machine learning, data visualization, hypothesis function and testing, data preprocessing, data augmentation, normalizing data sets, Relation between AI (Artificial Intelligence), ML (Machine Learning), DL (Deep Learning) and DS (Data Science).

Course  
Outcome (CO1): To understand the basic theory underlying machine learning. Unit-I Machine Learning Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for building mathematical models and making predictions using historical data or information. Currently, it is being used for various tasks such as image recognition, speech recognition, email filtering, Facebook auto-tagging, recommender system, and many more. In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does. Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed. With the help of sample historical data, which is known as training data, machine learning algorithms build a mathematical model that helps in making predictions or decisions without being explicitly programmed. Machine learning brings computer science and statistics together for creating predictive models. Machine learning constructs or uses the algorithms that learn from historical data. The more we will provide the information, the higher will be the

performance. Machine Learning is said as a subset of artificial intelligence that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by Arthur Samuel in 1959. Machine learning life cycle Machine learning has given the computer systems the abilities to automatically learn without being explicitly programmed. But how does a machine learning system work? So, it can be described using the life cycle of machine learning. Machine learning life cycle is a cyclic process to build an efficient machine learning project. The main purpose of the life cycle is to find a solution to the problem or project. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Machine learning life cycle involves seven major steps, which are given below:

- Gathering Data
- Data preparation
- Data Wrangling
- Analyse Data
- Train the model
- Test the model
- Deployment

Fig-1.1 ML Life-Cycle The most important thing in the complete process is to understand the problem and to know the purpose of the problem. Therefore, before starting the life cycle, we need to understand the problem because the good result depends on the better understanding of the problem. In the complete life cycle process, to solve a problem, we create a machine learning system called "model", and this model is created by providing "training". But to train a model, we need data, hence, life cycle starts by collecting data. Gathering Data: Data Gathering is the first step of the machine learning life cycle. The goal of this step is to identify and obtain all data-related problems. In this step, we need to identify the different data sources, as data can be collected from various sources such as files, database, internet, or mobile devices. It is one of the most important steps of the life cycle. The quantity and quality of the collected data will determine the efficiency of the output. The more will be the data, the more accurate will be the prediction. This step includes the below tasks:

- Identify various data sources
- Collect data
- Integrate the data obtained from different sources

By performing the above task, we get a coherent set of data, also called as a dataset. It will be used in further steps.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Data preparation After collecting the data, we need to prepare it for further steps. Data preparation is a step where we put our data into a suitable place and prepare it to use in our machine learning training. In this step, first, we put all data together, and then randomize the ordering of data. This step can be further divided into two processes:

- Data exploration: It is used to understand the nature of data that we have to work with. We need to understand the characteristics, format, and quality of data. A better understanding of data leads to an effective outcome. In this, we find Correlations, general trends, and outliers.
- Data pre-processing: Now the next step is preprocessing of data for its analysis.

Data Wrangling Data wrangling is the process of cleaning and converting raw data into a useable format. It is the process of cleaning the data, selecting the variable to use, and transforming the data in a proper format to make it more suitable for analysis in the next step. It is one of the most important steps of the complete process. Cleaning of data is required to address the quality issues. It is not necessary that data we have collected is always of our use as some of the data may not be useful. In real-world applications, collected data may have various issues, including:

- Missing Values
- Duplicate data
- Invalid data
- Noise

So, we use various filtering techniques to clean the data. It is mandatory to detect and remove the above issues because it can negatively affect the quality of the outcome.

Data Analysis Now the cleaned and prepared data is passed on to the analysis step. This step involves:

- Selection of analytical techniques
- Building models
- Review the result

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science The aim of this step is to build a machine learning model to analyze the data using various analytical techniques and review the outcome. It starts with the determination of the type of the problems, where we select the machine learning techniques such as Classification, Regression, Cluster analysis, Association, etc. then build the model using prepared data, and evaluate the model.

Train Model Now the next step is to train the model, in this step we train our model to improve its performance for better outcome of the problem. We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and, features.

Test Model Once our machine learning model has been trained on a given dataset, then we test the model. In this step, we check for the accuracy of our model by providing a test dataset to it. Testing the model determines the percentage accuracy of the model as per the requirement of project or problem.

Deployment The last step of machine learning life cycle is deployment, where we deploy the model in the real-world system. If the above-prepared model is producing an accurate result as per our requirement with acceptable speed, then we deploy the model in the real system. But before deploying the project, we will check whether it is improving its performance using available data or not. The deployment phase is similar to making the final report for a project.

Types of Machine Learning System Based on the methods and way of learning, machine learning is divided into mainly four types, which are.

1. Supervised Machine

Learning 2. Unsupervised Machine Learning 3. Semi-Supervised Machine Learning 4. Reinforcement Learning Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Fig-1.2 Type of ML Supervised Machine Learning As its name suggests, Supervised machine learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More precisely, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset. Example Suppose we have an input dataset of cats and dog images. So, first, we will provide the training to the machine to understand the images, such as the shape & size of the tail of cat and dog, Shape of eyes, colour, height (dogs are taller, cats are smaller), etc. After completion of training, we input the picture of a cat and ask the machine to identify the object and predict the output. Now, the machine is well trained, so it will check all the features of the object, such as height, shape, colour, eyes, ears, tail, etc., and find that it's a cat. So, it will put it in the Cat category. This is the process of how the machine identifies the objects in Supervised Learning. The main goal of the supervised learning technique is to map the input variable(x) with the output variable(y). Some real-world applications of supervised learning are Risk Assessment, Fraud Detection, Spam filtering, etc. Categories of Supervised Machine Learning Supervised machine learning can be classified into two types of problems, which are given below: o Classification o Regression Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Unsupervised Machine Learning Unsupervised learning is different from the Supervised learning technique; as its name suggests, there is no need for supervision. It means, in unsupervised machine learning, the machine is trained using the unlabeled dataset, and the machine predicts the output without any supervision. In unsupervised learning, the models are trained with the data that is neither classified nor labelled, and the model acts on that data without any supervision. The main aim of the unsupervised learning algorithm is to group or categories the unsorted dataset according to the similarities, patterns, and differences. Machines are instructed to find the hidden patterns from the input dataset. Example suppose there is a basket of fruit images, and we input it into the machine learning model. The images are totally unknown to the model, and the task of the machine is to find the patterns and categories of the objects. So, now the machine will discover its patterns and differences, such as colour difference, shape difference, and predict the output when it is tested with the test dataset. Categories of Unsupervised Machine Learning Unsupervised Learning can be further classified into two types, which are given below: o Clustering o Association Semi-Supervised Learning Semi-Supervised learning is a type of Machine Learning algorithm that lies between Supervised and Unsupervised machine learning. It represents the intermediate ground between Supervised (With Labelled training data) and Unsupervised learning (with no labelled training data) algorithms and uses the combination of labelled and unlabeled datasets during the training period. Reinforcement Learning Reinforcement learning works on a feedback-based process, in which an AI agent (A software component) automatically explores its surroundings by hitting & trail, taking action, learning from experiences, and improving its performance. Agent gets rewarded for each good action and gets punished for each bad action; hence the goal of reinforcement learning agent is to maximize the rewards. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science In reinforcement learning, there is no labelled data like supervised learning, and agents learn from their experiences only. The reinforcement learning process is similar to a human being; for example, a child learns various things by experiences in his day-to-day life. An example of reinforcement learning is to play a game, where the Game is the environment, moves of an agent at each step define states, and the goal of the agent is to get a high score. Agent receives feedback in terms of punishment and rewards. Due to its way of working, reinforcement learning is employed in different fields such as Game theory, Operation Research, Information theory, multi-agent systems. Batch and online learning Batch or offline learning Batch learning represents the training of machine learning models in a batch manner. In other words, batch learning represents the training of the models at regular intervals such as weekly, bi-weekly, monthly, quarterly, etc. In batch learning, the system is not capable of learning incrementally. The models must be trained using all the available data every single time. The data gets accumulated over a period of time. The models then get trained with the accumulated data from time to time at periodic intervals. This model training takes a lot of time and computing resources. Hence, it is typically done offline. After the models are trained, they are launched into production and they run without learning anymore. Batch learning is also called offline learning. The models trained using batch or offline learning are moved into production only at regular intervals based on the performance of models trained with new data. Online Learning In online learning, the training happens in an incremental manner by continuously feeding data as it arrives or in a small group /

mini batches. Each learning step is fast and cheap, so the system can learn about new data on the fly, as it arrives. Online learning is great for machine learning systems that receive data as a continuous flow (e.g., stock prices) and need to adapt to change rapidly or autonomously. It is also a good option if you have limited computing resources: once an online learning system has learned about new data instances, it does not need them anymore, so you can discard them (unless you want to be able to roll back to a previous state and “replay” the data) or move the data to another form of storage (warm or cold storage) if you are using the data lake. This can save a huge amount of space and cost.

**Instance-Based and Model based Learning**

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science

**Instance-Based Learning**

The Machine Learning systems which are categorized as instance-based learning are the systems that learn the training examples by heart and then generalizes to new instances based on some similarity measure. It is called instance-based because it builds the hypotheses from the training instances. It is also known as memory-based learning or lazy-learning (because they delay processing until a new instance must be classified).

**Model-based Learning**

Model-based learning (also known as structure-based or eager learning) takes a different approach by constructing models from the training data that can generalize better than instance-based methods. This involves using algorithms like linear regression, logistic regression, random forest, etc. trees to create an underlying model from which predictions can be made for new data points.

**scope and limitations**

Future Scope of Machine Learning

The scope of Machine Learning is not limited to the investment sector. Rather, it is expanding across all fields such as banking and finance, information technology, media & entertainment, gaming, and the automotive industry. As the Machine Learning scope is very high, there are some areas where researchers are working toward revolutionizing the world for the future.

Automotive Industry Robotics Quantum Computing Computer Vision

**Limitations of ML**

Nothing is perfect in the world. Machine Learning has some serious limitations, which are bigger than human errors.

1. Data Acquisition
2. Time and Resources
3. Results Interpretations
4. High Error Chances

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science

5. Social Changes
6. Elimination of Human Interface
7. Changing Nature of Jobs
8. Highly Expensive
9. Privacy Concern
10. Research and Innovations

**Challenges of Machine learning**

1. Poor Quality of Data
2. Underfitting of Training Data
3. Overfitting of Training Data
4. Machine Learning is a Complex Process
5. Lack of Training Data
6. Slow Implementation
7. Imperfections in the Algorithm

When Data Grows

Data visualization

Data visualization is a crucial aspect of machine learning that enables analysts to understand and make sense of data patterns, relationships, and trends. Through data visualization, insights and patterns in data can be easily interpreted and communicated to a wider audience, making it a critical component of machine learning.

1. Line Charts
2. Scatter Plots
3. Bar Charts
4. Heat Maps
5. Tree Maps
6. Box Plots

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science

**Hypothesis Testing**

Hypothesis testing is a statistical method that is used in making a statistical decision using experimental data. Hypothesis testing is basically an assumption that we make about a population parameter. It evaluates two mutually exclusive statements about a population to determine which statement is best supported by the sample data.

**Need for Hypothesis Testing**

Hypothesis testing is an important procedure in statistics. Hypothesis testing evaluates two mutually exclusive population statements to determine which statement is most supported by sample data. When we say that the findings are statistically significant, it is thanks to hypothesis testing.

**Parameters of hypothesis testing**

Null hypothesis(H0): In statistics, the null hypothesis is a general given statement or default position that there is no relationship between two measured cases or no relationship among groups. In other words, it is a basic assumption or made based on the problem knowledge.

Example: A company production is = 50 units/per day etc.

Alternative hypothesis(H1): The alternative hypothesis is the hypothesis used in hypothesis testing that is contrary to the null hypothesis.

Example: A company's production is not equal to 50 units/per day etc.

**Level of significance:** It refers to the degree of significance in which we accept or reject the null hypothesis. 100% accuracy is not possible for accepting a hypothesis, so we, therefore, select a level of significance that is usually 5%. This is normally denoted with  $\alpha$  and generally, it is 0.05 or 5%, which means your output should be 95% confident to give a similar kind of result in each sample.

**P-value :** The P value, or calculated probability, is the probability of finding the observed/extreme results when the null hypothesis(H0) of a study-given problem is true. If your P-value is less than the chosen significance level then you reject the null hypothesis i.e. accept that your sample claims to support the alternative hypothesis.

**Data pre-processing**

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

Chameli Devi Group of Institutions Department of

**Artificial Intelligence & Data Science** It involves below steps:

- o Getting the dataset
- o Importing libraries
- o Importing datasets
- o Finding Missing Data
- o Encoding Categorical Data
- o Splitting dataset into training and test set
- o Feature scaling

**Data Augmentation** Data augmentation is the process of modifying, or “augmenting” a dataset with additional data. This additional data can be anything from images to text, and its use in machine learning algorithms helps improve their performance. For example, say we wanted to build a model to classify dog breeds, and we have a lot of images of most breeds, except for pugs. As a result, the model wouldn’t be able to classify pugs well. We could augment the data by adding some (real or fake) images of pugs, or by multiplying our existing pug images (e.g. by replicating and distorting them to make them artificially unique). Data augmentation is crucial for many AI applications, as accuracy increases with the amount of training data. In fact, research studies have found that basic data augmentation can greatly improve accuracy on image tasks, such as classification and segmentation. Further, large neural networks, or deep learning models, need a huge amount of data, so they benefit even more from data augmentation techniques.

**Normalizing Data Sets** Normalization is a scaling technique in Machine Learning applied during data preparation to change the values of numeric columns in the dataset to use a common scale. It is not necessary for all datasets in a model. It is required only when features of machine learning models have different ranges. Mathematically, we can calculate normalization with the below formula:  $X_n = (X - X_{\text{minimum}}) / (X_{\text{maximum}} - X_{\text{minimum}})$

- o  $X_n$  = Value of Normalization
- o  $X_{\text{maximum}}$  = Maximum value of a feature
- o  $X_{\text{minimum}}$  = Minimum value of a feature

**Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science** Normalization techniques in Machine Learning Although there are so many feature normalization techniques in Machine Learning, few of them are most frequently used. These are as follows:

- Min-Max Scaling: This technique is also referred to as scaling. The Min-Max scaling method helps the dataset to shift and rescale the values of their attributes, so they end up ranging between 0 and 1.
- Standardization Scaling: Standardization scaling is also known as Z-score normalization, in which values are centered around the mean with a unit standard deviation, which means the attribute becomes zero and the resultant distribution has a unit standard deviation. Mathematically, we can calculate the standardization by subtracting the feature value from the mean and dividing it by standard deviation. Hence, standardization can be expressed as follows:

**Relation Between AI , ML , DL and DS**

AI: The Dream of Intelligent Machines

Artificial Intelligence, or AI, is a field of computer science that aims to create intelligent machines capable of performing tasks that typically require human intelligence. It encompasses a wide range of techniques that enable computers to mimic human cognitive abilities like learning, problem-solving, reasoning, and decisionmaking.

ML: The Essence of AI

Machine Learning is a subset of AI that focuses on designing algorithms and statistical models that allow machines to learn from data without being explicitly programmed. In other words, instead of hand-coding specific instructions, we let the machine learn patterns and relationships from the data to make predictions or decisions.

DL: Unleashing the Power of Neural Networks

Deep Learning is a subfield of ML that has gained tremendous popularity in recent years. It is inspired by the structure and function of the human brain and involves training artificial neural networks with vast amounts of data. Deep learning has revolutionized AI by achieving remarkable breakthroughs in image recognition, natural language processing, and much more.

**Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science**

DS: The Art of Extracting Insights from Data

Data Science is a multidisciplinary field that combines AI, ML, and other techniques to extract valuable insights and knowledge from data. It involves data collection, cleaning, analysis, and visualization to uncover hidden patterns and trends. Data scientists play a crucial role in guiding decision-making processes in various industries.

The Interplay between AI, ML, DL, and DS: AI serves as the overarching concept that aims to create intelligent systems. Within AI, ML is a powerful tool that enables machines to learn from data and make decisions. DL, as a subset of ML, leverages neural networks to achieve deeper levels of understanding and is particularly effective in complex tasks like image and speech recognition. Data Science acts as the bridge that connects AI, ML, and DL together. Without quality data, machine learning models, and deep learning algorithms would be ineffective. Data science plays a pivotal role in curating, processing, and preparing data for the learning algorithms to work their magic.

business function. Basic Concepts of MIS – To the managers, Management Information System is an implementation of the organizational systems and procedures. To a programmer it is nothing but file structures and file processing. However, it involves much more complexity. The three components of MIS provide a more complete and focused definition, where System suggests integration and holistic view, Information stands for processed data, and Management is the ultimate user, the decision makers. Management information system can thus be analyzed as follows –

AD Management

Management covers the planning, control, and administration of the operations of a concern. The top management handles planning; the middle management concentrates on controlling; and the lower management is concerned with actual administration. Information Information, in MIS, means the processed data that helps the management in planning, controlling and operations. Data means all the facts arising out of the operations of the concern. Data is processed i.e. recorded, summarized, compared and finally presented to the management in the form of MIS report. System Data is processed into information with the help of a system. A system is made up of inputs, processing, output and feedback or control. Thus MIS means a system for processing data in order to give proper information to the management for performing its functions.

AD Definition Management Information System or 'MIS' is a planned system of collecting, storing, and disseminating data in the form of information needed to carry out the functions of management.

Objectives of MIS

The goals of an MIS are to implement the organizational structure and dynamics of the enterprise for the purpose of managing the organization in a better way and capturing the potential of the information system for competitive advantage.

Following are the basic objectives of an MIS –

- Capturing Data – Capturing contextual data, or operational information that will contribute in decision making from various internal and external sources of organization.
- Processing Data – The captured data is processed into information needed for planning, organizing, coordinating, directing and controlling functionalities at strategic, tactical and operational level. Processing data means – o making calculations with the data o sorting data o classifying data and o summarizing data
- Information Storage – Information or processed data need to be stored for future use.
- Information Retrieval – The system should be able to retrieve this information from the storage as and when required by various users.
- Information Propagation – Information or the finished product of the MIS should be circulated to its users periodically using the organizational network.

Characteristics of MIS

Following are the characteristics of an MIS –

- It should be based on a long-term planning.
- It should provide a holistic view of the dynamics and the structure of the organization.
- It should work as a complete and comprehensive system covering all interconnecting sub-systems within the organization.
- It should be planned in a top-down way, as the decision makers or the management should actively take part and provide clear direction at the development stage of the MIS.
- It should be based on need of strategic, operational and tactical information of managers of an organization.
- It should also take care of exceptional situations by reporting such situations.
- It should be able to make forecasts and estimates, and generate advanced information, thus providing a competitive advantage. Decision makers can take actions on the basis of such predictions.
- It should create linkage between all sub-systems within the organization, so that the decision makers can take the right decision based on an integrated view.
- It should allow easy flow of information through various sub-systems, thus avoiding redundancy and duplicity of data. It should simplify the operations with as much practicability as possible.
- Although the MIS is an integrated, complete system, it should be made in such a flexible way that it could be easily split into smaller sub-systems as and when required.
- A central database is the backbone of a well-built MIS.

Data v/s Information

Data can be described as unprocessed facts and figures. Plain collected data as raw facts cannot help in decision-making. However, data is the raw material that is organized, structured, and interpreted to create useful information systems. Data is defined as 'groups of non-random symbols in the form of text, images, voice representing quantities, action and objects'. Information is interpreted data; created from organized, structured, and processed data in a particular context. According to Davis and Olson – "Information is a data that has been processed into a form that is meaningful to recipient and is of real or perceived value in the current or the prospective action or decision of recipient."

Information/Data Collection Techniques

The most popular data collection techniques include –

- Surveys – A questionnaire is prepared to collect the data from the field.
- Secondary data sources or archival data: Data is collected through old records, magazines, company website etc.
- Objective measures or tests – An experimental test is conducted on the subject and the data is collected.
- Interviews – Data is collected by the system analyst by following a rigid procedure and collecting the answers to a set of pre-conceived questions through personal interviews.

MIS - Information Need & Objective

Information processing beyond doubt is the dominant industry of the present century. Following factors

states few common factors that reflect on the needs and objectives of the information processing –

- Increasing impact of information processing for organizational decision making.
- Dependency of services sector including banking, financial organization, health care, entertainment, tourism and travel, education and numerous others on information.
- Changing employment scene world over, shifting base from manual agricultural to machinebased manufacturing and other industry related jobs.
- Information revolution and the overall development scenario.
- Growth of IT industry and its strategic importance.
- Strong growth of information services fuelled by increasing competition and reduced product life cycle.
- Need for sustainable development and quality life.
- Improvement in communication and transportation brought in by use of information processing.
- Use of information processing in reduction of energy consumption, reduction in pollution and a better ecological balance in future.
- Use of information processing in land record managements, legal delivery system, educational institutions, natural resource planning, customer relation management and so on.

Organization Structures, Organizational structure (OS) is the systematic arrangement of human resources in an organization so as to achieve common business objectives. It outlines the roles and responsibilities of every member of the organization so that work and information flow seamlessly, ensuring the smooth functioning of an organization. An organizational structure is a system that outlines how certain activities are directed in order to achieve the goals of an organization. These activities can include rules, roles, and responsibilities. The organizational structure also determines how information flows between levels within the company. For example, in a centralized structure, decisions flow from the top down, while in a decentralized structure, decision-making power is distributed among various levels of the organization. Having an organizational structure in place allows companies to remain efficient and focused. Understanding an Organizational Structure Businesses of all shapes and sizes use organizational structures heavily. They define a specific hierarchy within an organization. A successful organizational structure defines each employee's job and how it fits within the overall system. Put simply, the organizational structure lays out who does what so the company can meet its objectives. This structuring provides a company with a visual representation of how it is shaped and how it can best move forward in achieving its goals. Organizational structures are normally illustrated in some sort of chart or diagram like a pyramid, where the most powerful members of the organization sit at the top, while those with the least amount of power are at the bottom. Not having a formal structure in place may prove difficult for certain organizations. For instance, employees may have difficulty knowing to whom they should report. That can lead to uncertainty as to who is responsible for what in the organization. Having a structure in place can help with efficiency and provide clarity for everyone at every level. That also means each and every department can be more productive, as they are likely to be more focused on energy and time.

**Centralized vs. Decentralized Organizational Structures**

An organizational structure is either centralized or decentralized. Traditionally, organizations have been structured with centralized leadership and a defined chain of command. The military is an organization famous for its highly centralized structure, with a long and specific hierarchy of superiors and subordinates. In a centralized organizational system, there are very clear responsibilities for each role, with subordinate roles defaulting to the guidance of their superiors. There has been a rise in decentralized organizations, as is the case with many technology startups. This allows companies to remain fast, agile, and adaptable, with almost every employee receiving a high level of personal agency. For example, Johnson & Johnson is a company that's known for its decentralized structure. As a large company with over 200 business units and brands that function in sometimes very different industries, each operates autonomously. Even in decentralized companies, there are still usually built-in hierarchies (such as the chief operating officer operating at a higher level than an entry-level associate). However, teams are empowered to make their own decisions and come to the best conclusion without necessarily getting "approval" from up top.

**Types of Organizational Structures**

**Functional Structure**

Four types of common organizational structures are implemented in the real world. The first and most common is a functional structure. This is also referred to as a bureaucratic organizational structure and breaks up a company based on the specialization of its workforce. Most small-to-medium-sized businesses implement a functional structure. Dividing the firm into departments consisting of marketing, sales, and operations is the act of using a bureaucratic organizational structure.

**Divisional or Multidivisional Structure**

The second type is common among large companies with many business units. Called the divisional or multidivisional (M-Form) structure, a company that uses this method structures its leadership team based on the products, projects, or subsidiaries they operate. A good example of this structure is Johnson & Johnson. With thousands of products and lines of business, the company structures itself so each business unit operates as its own company with its own president. Divisions may also be designated geographically in addition to specialization. For instance, a global corporation may have a North American Division and a

European Division. Team-Based Similar to divisional or functional structures, team-based organizations segregate into close-knit teams of employees that serve particular goals and functions, but where each team is a unit that contains both leaders and workers. Flat (Flatarchy) Structure Flatarchy, also known as a horizontal structure, is relatively newer, and is used among many startups. As the name alludes, it flattens the hierarchy and chain of command and gives its employees a lot of autonomy. Companies that use this type of structure have a high speed of implementation. Matrix Structure Firms can also have a matrix structure. It is also the most confusing and the least used. This structure matrixes employees across different superiors, divisions, or departments. An employee working for a matrixed company, for example, may have duties in both sales and customer service. Circular Structure Circular structures are hierarchical, but they are said to be circular as it places higher-level employees and managers at the center of the organization with concentric rings expanding outward, which contain lower-level employees and staff. This way of organizing is intended to encourage open communication and collaboration among the different ranks. Network Structure The network structure organizes contractors and third-party vendors to carry out certain key functions. It features a relatively small headquarters with geographically-dispersed satellite offices, along with key functions outsourced to other firms and consultants. Business Process What is a business process? A business process is an activity or set of activities that accomplish a specific organizational goal. Business processes should have purposeful goals, be as specific as possible and produce consistent outcomes. A business process refers to a wide range of structured, often chained, activities or tasks conducted by people or equipment to produce a specific service or product for a particular user or consumer. Business processes are implemented to accomplish a predetermined organizational goal. Business processes occur at all organizational levels; some are visible to customers, while others are not. The term business process may also refer to the cumulative effects of all steps progressing toward a business goal. This sequence of steps can be most clearly depicted using a flowchart. A business process is also known as a business method. The three types of business processes are:

- Management Processes: The processes that govern the operation of a system.
- Operational Processes: The processes that constitute the core business of the organization and create the primary value stream.
- Supporting Processes: The processes that support the core processes. Examples include accounting and technical support. Examples of business processes include:

- Invoicing
- Shipping products
- Receiving orders
- Updating personnel data
- Determining marketing and other budgets

Role of MIS A management information system (MIS) plays an important role in business organizations. There are many MIS and some of the important MIS are discussed below:

1. Decision making
2. Coordination among the department
3. Finding out Problems
4. Comparison of Business Performance
5. Strategies for an Organization

Role of MIS Decision making Management Information System (MIS) plays a significant role in the decision-making process of any organization. In any organization, a decision is made on the basis of relevant information which can be retrieved from the MIS.

Coordination among the department Management Information System satisfy multiple need of an organization across the different functional department. Finding out Problems As we know that MIS provides relevant information about every aspect of activities. Hence, if any mistake is made by the management then MIS, information will help in finding out the solution to that problem. Comparison of Business Performance MIS store all past data and information in its Database. That why the management information system is very useful to compare business organization performance. Strategies for an Organization Today each business is running in a competitive market. An MIS supports the organization to evolve appropriate strategies for the business to assent in a competitive environment.

**IMPORTANCE OF MIS** It goes without saying that all managerial functions are performed through decision-making; for taking rational decision, timely and reliable information is essential and is procured through a logical and wellstructured method of information collecting, processing and disseminating to decision makers. Such a method in the field of management is widely known as MIS. In today's world of ever increasing complexities of business as well as business organization, in order to service and grow , must have a properly planned, analyzed, designed and maintained MIS so that it provides timely, reliable and useful information to enable the management to take speedy and rational decisions. MIS has assumed all the more important role in today's environment because a manager has to take decisions under two main challenges: First, because of the liberalization and globalization, in which organizations are required to compete not locally but globally, a manager has to take quick decisions, otherwise his business will be taken away by his competitors. This has further enhanced the necessity for such a system. Second, in this information age wherein information is doubling up every two or three years, a manager has to process a large voluminous data; failing which he may end up taking a strong decision that may prove to be very costly to the company.

**Scope of MIS** There are many opportunities for students after completing the degree in

MIS. They are listed below.

- Information Systems Manager
- Business Intelligence Analyst
- Network Administrator
- Web Developer
- IT Consultant
- Systems Developer
- Technical Support Specialist
- Business Application Developer

**IMPACT OF THE MANAGEMENT INFORMATION SYSTEM** MIS plays a very important role in the organization; it creates an impact on the organization's functions, performance and productivity.

→ The impact of MIS on the functions is in its management with a good MIS supports the management of marketing, finance, production and personnel becomes more efficient. The tracking and monitoring of the functional targets becomes easy. The functional managers are informed about the progress, achievements and shortfalls in the activity and the targets.

→ The manager is kept alert by providing certain information indicating and probable trends in the various aspects of business. This helps in forecasting and long-term perspective planning. The manager's attention is bought to a situation which is expected in nature, inducing him to take an action or a decision in the matter.

→ Disciplined information reporting system creates structure database and a knowledge base for all the people in the organization. The information is available in such a form that it can be used straight away by blending and analysis, saving the manager's valuable time.

→ The MIS creates another impact in the organization which relates to the understanding of the business itself. The MIS begins with the definition of data, entity and its attributes. It uses a dictionary of data, entity and attributes, respectively, designed for information generation in the organization. Since all the information systems use the dictionary, there is common understanding of terms and terminology in the organization bringing clarity in the communication and a similar understanding of an event in the organization.

→ The MIS calls for a systematization of the business operations for an effective system design. This leads to streamlining of the operations which complicates the system design. It improves the administration of the business by bringing a discipline in its operations as everybody is required to follow and use systems and procedures. This process brings a high degree of professionalism in the business operations.

→ The goals and objectives of the MIS are the products of business goals and objectives. It helps indirectly to pull the entire organization in one direction towards the corporate goals and objectives by providing the relevant information to the organization.

→ A well designed system with a focus on the manager makes an impact on the managerial efficiency. The fund of information motivates an enlightened manager to use a variety of tools of the management. It helps him to resort to such exercises as experimentation and modeling.

→ The use of computers enables him to use the tools and techniques which are impossible to use manually. The ready-made packages make this task simple. The impact is on the managerial ability to perform. It improves decision-making ability considerably high.

→ Since, the MIS work on the basic system such as transaction processing and database, the drudgery of the clerical work is transferred to the computerized system, relieving the human mind for better work. It will be observed that lot of manpower is engaged in this activity in the organization. Seventy (70) percent of the time is spent in recording, searching, processing and communicating. This MIS has a direct impact on this overhead. It creates information-based working culture in the organization.

**Advantages of MIS:**

- Improves quality of an organization or information content by providing relevant information for sound decision making.
- MIS change large amount of data into summarize form and thereby avoid confusion which may be an answer when an information officer are flooded with detailed fact.
- MIS facilitates integration of specialized activities by keeping each department aware of problem and requirements of other departments.
- MIS serves as a link between managerial planning and control. It improves the ability of management to evaluate and improve performance.

**Disadvantages:**

- Too rigid and difficult to adapt.
- Resistance in sharing internal information between departments can reduce the effectiveness.
- Hard to quantify benefit to justify implementation of MIS.
- Quality of output of an MIS is directly proportional to quality of input and processes.

**MIS: A support to management- Structure/Architecture of MIS**The chain of superior-subordinate relationships is known as the Levels of Management. The three levels of management are Top Level Management, Middle-Level Management, and Operational Level Management. Management is a group activity, which means that every organization has a number of individuals placed at different positions and are provided with different responsibilities according to their skills, education, etc. For the fulfillment of the responsibilities given to the members of an organization, they are also provided with the required authority. Based on the amount and extent of responsibility and authority given to these members, a chain of superior-subordinate relationships is formed. This chain of superior-subordinate relationships is known as the Levels of Management. There are three levels of management; viz., Top Level Management, Middle Level Management, and Operational Level Management.

**Three Levels of Management**

1. Top Level Management
- The senior most executives of the organization are found at the top level of management. The top level of an organization's management consists of the

Board of Directors, Managing Director, Chairman, Chief Executive Officer, Chief Operating Officer, Vice-President, President, General Manager, and other Senior Executives. The managers at the top level of management of an organization are responsible for its survival and welfare. These managers perform stressful and complex work that demands long hours and commitment towards the company. Functions of the Top Level Management i) Determination of the objectives for the organization: The managers at the top level management formulates the goals or objectives for an organization along with the strategies to achieve those goals. ii) Framing of plans and policies: For the achievement of the pre-determined goals or objectives of an organization, it is essential to formulate proper strategies, plans and policies within the organization. The top level managers are responsible for the formulation of these plans and policies. iii) Coordination and control of the performance: Based on the overall pre-determined objectives of the organization, the top level managers coordinate and control different activities of different departments of the organization. iv) Analysis of the business environment: Business environment of an organization plays a crucial role in its success and survival. The managers at the top level of management of an organization carefully analyze the business environment and its implication and make necessary decisions for better results. v) Setting up an organizational framework: For the success and survival of an organization, it is essential to form a proper framework or structure within the company. The top level managers are responsible for the determination of the organizational framework for the proper and successful execution of its plans and policies. vi) Assembling of the resources: Achievement of the organizational goals requires different resources of materials, machines, men, money and materials. It is the duty of the managers at the top level management to arrange these resources.

2. Middle Level Management The next level of management is the Middle Level, which serves as a link between the Top Level Management and the Lower Level Management. The middle level management is superior to the lower or operational level management and subordinate to the top level management. The middle level of an organization's management consists of different functional department heads, such as Departmental Managers including Production, Purchase, Finance, Personnel, Marketing Managers, and other executive officers for different departments such as plant superintendent, etc. The employees or members of the middle level management are responsible to the top level management for their performance. Functions of the Middle Level Management i) Interpretation of the policies framed by the Top Level Management: As the middle level management acts as a subordinate to the top level management, the managers at this level have to clearly interpret the plans and policies framed by the managers at the top level management to the managers at the lower or operational level management. ii) Selection of suitable operative and supervisory personnel: To perform any function properly, an organization needs the required personnel. It is the duty of the Middle Level Managers to make sure that the organization has sufficient personnel with them to perform the functions and duties better. For the fulfillment of this duty, the middle level managers recruit and select suitable employees for different departments based on the applicant's skills, etc., and the firm's requirements. iii) Assigning of duties and responsibilities to the Lower Level Management: The middle level managers acts as superior to the operational level managers. These managers have to assign respective duties and responsibilities to the lower level managers and coordinate with them regarding the activities of different work units. iv) Motivating employees to get desired objectives: An organization can effectively and efficiently achieve its desired goals only when its employees are motivated enough to work towards the betterment of the organization. Therefore, the managers at the middle level management motivate the employees towards the achievement of the organizational goals and improvement of their performance. v) Cooperating with the entire organization: As middle level management serves as a link between the top level management and the lower level management, the managers at this level have to cooperate with every other department for the smooth functioning of the organization.

3. Lower Level Management The last level of management is the lower level management and is also known as the Supervisory or Operational Level Management. The managers at the lower level of management play a crucial role in the proper management of an organization, as they directly interact with the actual work force and interpret the instructions of the middle level managers to them. The responsibility and authority of the lower level managers depend upon the plans and policies formed by the top level management. The lower level management consists of foremen, supervisors, section officers, superintendents, and other managers who have direct control over the operative employees of the organization.

Functions of the Lower Level Management i) Issuing of orders and instructions: The managers at the operational level management issue orders to the workers and supervisors and instructs them on their roles, responsibilities, and authority. Besides, these managers also control the functioning of the workers. ii) Preparation of plan for activities: The lower level

managers plan the day-to-day activities of the organization. Besides, these managers also assign work to the subordinates, guide them for the same, and take corrective measures wherever and whenever necessary.

iii) Assigning and assisting in work: The job or responsibility of the lower level managers includes assigning work to the subordinates and assisting them with the work. They do so by explaining the work procedure to the employees and solving their problems for better performance.

iv) Representing workers' grievances: As the managers at the lower level management are in direct contact with the managers at the middle level management, they listen to the grievances of the workers and report those issues to the middle level managers.

v) Ensuring a safe and proper work environment: The lower level managers are responsible for providing the work force with a safe and proper work environment. They also have to maintain proper discipline and a good atmosphere within the organization, as it motivates the employees to work towards the accomplishment of the organizational goals.

vi) Helping the middle level management: The managers at the operational level management helps the middle level managers in selecting, training, placing, and promoting the workers of an organization as they can give a direct insight as to what is required for the achievement of the organizational goals and about the performance of the workers.

vii) Encourage initiative of employees: The best way to motivate employees and make them feel an important part of the organization is by encouraging them to take initiative. The lower level managers do so by welcoming their suggestions and ideas and by rewarding them for the good ones.

Classification of MIS based on information characteristics

Information can be classified in a number of ways, and you will learn two of the most important ways to classify information.

Classification by Characteristic Based on Anthony's classification of Management, information used in business for decision-making is generally categorized into three types –

- Strategic Information – Strategic information is concerned with long term policy decisions that defines the objectives of a business and checks how well these objectives are met. For example, acquiring a new plant, a new product, diversification of business etc, comes under strategic information.
- Tactical Information – Tactical information is concerned with the information needed for exercising control over business resources, like budgeting, quality control, service level, inventory level, productivity level etc.
- Operational Information – Operational information is concerned with plant/business level information and is used to ensure proper conduction of specific operational tasks as planned/intended. Various operator specific, machine specific and shift specific jobs for quality control checks comes under this category.

Classification by Application In terms of applications, information can be categorized as –

- Planning Information – These are the information needed for establishing standard norms and specifications in an organization. This information is used in strategic, tactical, and operation planning of any activity. Examples of such information are time standards, design standards.
- Control Information – This information is needed for establishing control over all business activities through feedback mechanism. This information is used for controlling attainment, nature and utilization of important processes in a system. When such information reflects a deviation from the established standards, the system should induce a decision or an action leading to control.
- Knowledge Information – Knowledge is defined as "information about information". Knowledge information is acquired through experience and learning, and collected from archival data and research studies.
- Organizational Information – Organizational information deals with an organization's environment, culture in the light of its objectives. Karl Weick's Organizational Information Theory emphasizes that an organization reduces its equivocality or uncertainty by collecting, managing and using these information prudently. This information is used by everybody in the organization; examples of such information are employee and payroll information.
- Functional/Operational Information – This is operation specific information. For example, daily schedules in a manufacturing plant that refers to the detailed assignment of jobs to machines or machines to operators. In a service oriented business, it would be the duty roster of various personnel. This information is mostly internal to the organization.
- Database Information – Database information construes large quantities of information that has multiple usage and application. Such information is stored, retrieved and managed to create databases. For example, material specification or supplier information is stored for multiple users.

Application of MIS- MIS application in business MIS application in business falls into several different categories that provide information on all forms of functioning within an organization. Executives and departments within an organization could obtain any of the following forms of data:

- Business Intelligence System: In BI, all levels of management and executives can print data and graphs showing information or trends relating to growth, costs, strategic control, efficiency, risk and performance.
- Executive Information System: An EI system provides the same information as a BI system, but with greater attention to detail and more confidential information, designed to help top-level executives make choices that impact the entire organization.
- Marketing Information System: MIS systems provide data about past marketing campaigns so that

marketing executives can determine what works, what does not work and what they need to change in order to achieve the desired results. • Transaction Processing System: TPS handles sales transactions and makes it possible for customers to sort search results by size, color or price. This system can also track trends related to sales and search results. • Customer Relationship Management System: Keeping up with customers is key to overall success, and CRMS helps companies know when and how to follow up with customers in order to encourage an ongoing sales relationship with them. • Sales Force Automation System: Gone are the days when sales teams must do everything manually. SFA systems automate much of what must be done for orders and to obtain customer information. • Human Resource Management System: HRM systems track how much employees are paid, when and how they are performing. Companies can use this information to help improve performance or the bottom line. • Knowledge Management System: Customers with questions want answers right away and knowledge management systems allow them to access frequently asked questions or troubleshoot on their own timetable. • Financial Accounting System: Financial accounting systems help to track accounts receivable and accounts payable, in order to best manage the cash flow of a company. • Supply Chain Management System: SCM systems record and manage the supply of finances, goods and data from the point of origin domestically or abroad, all the way to its destination in the hands of a customer. Business functions are the activities performed by a business, and the activities demonstrate the purpose of the business. Well-structured business functions ensure a company is successful for its customers, employees, investors, and other stakeholders. The activities, measures, or processes undertaken by a business are specific to its functions and support the smooth operation of any organization and maximize its success. The term “function” describes the organizational component and the kinds of tasks it carries out. Business functions explain the procedures the business goes through, engage and design. They are the systems, tasks, and processes that make up a company’s daily operations. These can range from finance, sales, and marketing to human resources. The daily business operations are attributed to various functions. They assist in assessing the company’s performance and generating new plans for its expansion. Marketing, finance, human resources, and production are a company’s primary functions that ensure the success and failures of the business. Types Business functions are divided into two broad categories: core functions and support functions. While support functions don’t directly provide funds to the company, core functions are all directly fundyielding. It is intended to assist the fundamental operations. Combined, both functions increase the organization’s efficiency and result in positive feedback. Once the functions and their differences are clearly understood, it is easy to identify the process that needs to be optimized to meet the defined goal.

1. Core Functions: The enterprise’s core business functions constitute its main activity or operations. In other words, the revenue-generating function. However, it may also comprise additional secondary activities if the company views them as core activities. Examples include:

- Production of goods & services
- Finance
- Marketing

2. Support Functions: The support business functions are additional or supporting tasks that it performs to enable or facilitate its production activity, which is one of its core business functions. The outputs of support functions are not directly aimed at the market or other external audiences. A business’s various tasks are categorized as business functions under support functions. These responsibilities are divided into sales, administration, production, finance, accounting, and marketing. Examples include:

- Public relations
- Quality control

Examples Let us look at business function examples to understand the concept better:

- Public Relations: A public relations or communications department manages all facets of public relations, internal branding, corporate communications, client servicing, and crisis management for a business or brand. To represent a business, they prepare press releases, connect with reporters and editors, serve as spokespersons in response to difficulties or concerns, and create speeches or policy proposals for CEOs and top executives.
- Research and Development (R&D): Businesses or organizations engaged in research and development (R & D) drive innovation. To assist businesses in developing new goods, services, or revenue streams that will increase profits, the R&D team conducts market research, industry comparisons, trend detection, product creation, and business experimentation. Firms and departments involved in research and development have extensive experience in analysis and a thorough awareness of the market circumstances in a particular industry or area.
- Sales and Marketing: To generate brand awareness and revenue, sales and marketing teams interact with possible investors, clients, customers, or sponsors to grow their organization. It involves selecting the audience, determining fair prices and promotions, and designing and conducting successful marketing and advertising campaigns.
- Human Resource: Companies and departments in human resources concentrate on tasks related to employees, such as finding top talent, conducting background checks on applicants, hiring, outlining benefits, handling performance management and employee relations, creating corporate policies, and

promoting organizational culture. Along with resolving disputes and looking into claims or allegations, HR businesses or departments also ensure that employment rules and regulations are followed. \*\*\*\*\*

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Name: Machine Learning Subject Code: AD 502 Subject Notes Syllabus: Classification algorithm: - Logistic Regression, Decision Tree Classification, K-Nearest Neighbors (K-NN), Support Vector Machine, Naive Bayes (Gaussian, Multinomial, Bernoulli). Performance Measures: Confusion Matrix, Classification Accuracy, Classification Report: Precisions, Recall, F1 and Suppor.

Course

Outcome (CO3): To understand a range of machine learning algorithms along with their strengths and weaknesses. Unit-III Classification Algorithm The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, Yes or No, 0 or 1, Spam or Not Spam, cat or dog, etc. Classes can be called as targets/labels or categories. Unlike regression, the output variable of Classification is a category, not a value, such as "Green or Blue", "fruit or animal", etc. Since the Classification algorithm is a Supervised learning technique, hence it takes labeled input data, which means it contains input with the corresponding output. In classification algorithm, a discrete output function(y) is mapped to input variable(x) The best example of an ML classification algorithm is Email Spam Detector. The main goal of the Classification algorithm is to identify the category of a given dataset, and these algorithms are mainly used to predict the output for the categorical data. Classification algorithms can be better understood using the below diagram. In the below diagram, there are two classes, class A and Class B. These classes have features that are similar to each other and dissimilar to other classes. Figure-3.1 Classification Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science The algorithm which implements the classification on a dataset is known as a classifier. There are two types of Classifications: Binary Classifier: If the classification problem has only two possible outcomes, then it is called as Binary Classifier. Examples: YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc. Multi-class Classifier: If a classification problem has more than two outcomes, then it is called as Multi-class Classifier. Example: Classifications of types of crops, Classification of types of music. Logistic Regression Algorithm Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or false, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1. Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems. In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1). The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc. Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets. Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function: Figure-3.2 Logistic Regression Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Logistic Function (Sigmoid Function): • The sigmoid function is a mathematical function used to map the predicted values to probabilities. • It maps any real value into another value within a range of 0 and 1. • The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function. • In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0. Logistic Regression Equation: The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below: We know the equation of the straight line can be written as: Decision Tree Classification • Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree structured

classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. • In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. • The decisions or the test are performed on the basis of features of the given dataset. • It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. • It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. • In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. • A decision tree simply asks a question, and based on the answer (Yes/No), it further splits the tree into subtrees. Below diagram explains the general structure of a decision tree: Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Figure-3.3 Decision Tree Classification Decision Tree Terminologies • Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets. • Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node. • Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions. • Branch/Sub Tree: A tree formed by splitting the tree. • Pruning: Pruning is the process of removing the unwanted branches from the tree. • Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes. How does the Decision Tree algorithm Work? • Step-1: Begin the tree with the root node, say S, which contains the complete dataset. • Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM). • Step-3: Divide the S into subsets that contain possible values for the best attributes. • Step-4: Generate the decision tree node, which contains the best attribute. • Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and call the final node as a leaf node.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science K-Nearest Neighbors (K-NN)- • K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. • K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. • K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well-suited category by using K-NN algorithm. • K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. • K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. • It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. • KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data. Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category. Figure-3.4.1 K-Nearest Neighbors (K-NN)- Why do we need a K-NN Algorithm? Suppose there are two categories, i.e., Category A and Category B, and we have a new data point  $x_1$ , so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram: Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Figure-3.4.2 K-Nearest Neighbors (K-NN)- The K-NN working can be explained on the basis of the below algorithm: • Step-1: Select the number K of the neighbors • Step-2: Calculate the Euclidean distance of K number of neighbors • Step-3: Take the K nearest neighbors as per the calculated Euclidean distance. • Step-4: Among these k neighbors, count the number of the data points in each category. • Step-5: Assign the new data points to that category for which the number of the neighbor is maximum. • Step-6: Our model is ready. Support Vector Machine • Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. • The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. • SVM chooses the extreme points/vectors that

help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane: Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science

Figure-3.5.1 Support Vector Machine Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram: Figure-3.5.2 Support Vector Machine SVM algorithm can be used for Face detection, image classification, text categorization, etc. Types of SVM • Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science • Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier. Naïve Bayes Classifier Algorithm • Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. • It is mainly used in text classification that includes a high-dimensional training dataset. • Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. • It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. • Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles. The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

• Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

• Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem. Working of Naïve Bayes' Classifier: Working of Naïve Bayes' Classifier can be understood with the help of the below example: Suppose we have a dataset of weather conditions and corresponding target variable "Play". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables. 2. Generate Likelihood table by finding the probabilities of given features.

3. Now, use Bayes theorem to calculate the posterior probability. Problem: If the weather is sunny, then the Player should play or not? Solution: To solve this, first consider the below dataset: Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Frequency table for the Weather Conditions: Weather Yes No Overcast 5 0 Rainy 2 2

Sunny 3 2 Total 10 5 Likelihood table weather condition: Weather No Yes Overcast 0 5 5/14= 0.35 Rainy 2 2 4/14=0.29 Sunny 2 3 5/14=0.35 All 4/14=0.29 10/14=0.71 Applying Bayes'theorem:  $P(\text{Yes}|\text{Sunny}) = P(\text{Sunny}|\text{Yes}) * P(\text{Yes}) / P(\text{Sunny})$

Outlook Play 0 Rainy Yes 1 Sunny Yes 2 Overcast Yes 3 Overcast Yes 4 Sunny No 5 Rainy Yes 6 Sunny Yes 7 Overcast Yes 8 Rainy No 9 Sunny No 10 Sunny Yes 11 Rainy No 12 Overcast Yes 13 Overcast Yes Chameli Devi Group of Institutions

Department of Artificial Intelligence & Data Science  $P(\text{Sunny}|\text{Yes}) = 3/10 = 0.3$   $P(\text{Sunny}) = 0.35$   $P(\text{Yes}) = 0.71$  So  $P(\text{Yes}|\text{Sunny}) = 0.3 * 0.71 / 0.35 = 0.60$

$P(\text{No}|\text{Sunny}) = P(\text{Sunny}|\text{No}) * P(\text{No}) / P(\text{Sunny})$   $P(\text{Sunny}|\text{NO}) = 2/4 = 0.5$   $P(\text{No}) = 0.29$   $P(\text{Sunny}) = 0.35$  So  $P(\text{No}|\text{Sunny}) = 0.5 * 0.29 / 0.35 = 0.41$  So as we can see from the above calculation that  $P(\text{Yes}|\text{Sunny}) > P(\text{No}|\text{Sunny})$  Hence

on a Sunny day, Player can play the game. Performance Measures Evaluating the performance of a Machine learning model is one of the important steps while building an effective ML model. To evaluate the performance or quality of the model, different metrics are used, and these metrics are known as performance metrics or evaluation metrics. These performance metrics help us understand how well our model has performed for the given data. In this way, we can improve the model's performance by tuning the hyper-parameters. Each ML model aims to generalize well on unseen/new data, and performance metrics help determine how well the model generalizes on the new dataset. In machine learning, each task or problem is divided into classification and Regression. Not all metrics can be used for all types of problems; hence, it is important to know and understand which metrics should be used. Different evaluation metrics are used for both Regression and Classification tasks. Performance Metrics for Classification In a classification

problem, the category or classes of data is identified based on training data. The model learns from the given dataset and then classifies the new data into classes or groups based on the training. It predicts class labels as the output, such as Yes or No, 0 or 1, Spam or Not Spam, etc. To evaluate the performance of a classification model, different metrics are used, and some of them are as follows:

- Accuracy
- Confusion Matrix
- Precision

Chameli Devi Group of Institutions

Department of Artificial Intelligence & Data Science • Recall • F-Score • AUC(Area Under the Curve)-ROC Confusion Matrix

A confusion matrix is a tabular representation of prediction outcomes of any binary classifier, which is used to describe the performance of the classification model on a set of test data when true values are known. The confusion matrix is simple to implement, but the terminologies used in this matrix might be confusing for beginners. A typical confusion matrix for a binary classifier looks like the below image(However, it can be extended to use for classifiers with more than two classes). We can determine the following from the above matrix:

- In the matrix, columns are for the prediction values, and rows specify the Actual values. Here Actual and prediction give two possible classes, Yes or No. So, if we are predicting the presence of a disease in a patient, the Prediction column with Yes means, Patient has the disease, and for NO, the Patient doesn't have the disease.
- In this example, the total number of predictions are 165, out of which 110 time predicted yes, whereas 55 times predicted No.
- However, in reality, 60 cases in which patients don't have the disease, whereas 105 cases in which patients have the disease.

In general, the table is divided into four terminologies, which are as follows:

1. True Positive(TP): In this case, the prediction outcome is true, and it is true in reality, also.
2. True Negative(TN): in this case, the prediction outcome is false, and it is false in reality, also.
3. False Positive(FP): In this case, prediction outcomes are true, but they are false in actuality.
4. False Negative(FN): In this case, predictions are false, and they are true in actuality.

Classification Accuracy The accuracy metric is one of the simplest Classification metrics to implement, and it can be determined as the number of correct predictions to the total number of predictions. It can be formulated as: To implement an accuracy metric, we can compare ground truth and predicted values in a loop, or we can also use the scikit-learn module for this. Precisions The precision metric is used to overcome the limitation of Accuracy. The precision determines the proportion of positive prediction that was actually correct. It can be calculated as the True Positive or predictions that are actually true to the total positive predictions (True Positive and False Positive). Recall or Sensitivity It is also similar to the Precision metric; however, it aims to calculate the proportion of actual positive that was identified incorrectly. It can be calculated as True Positive or predictions that are actually true to the total number of positives, either correctly predicted as positive or incorrectly predicted as negative (true Positive and false negative). The formula for calculating Recall is given below:

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science

F1 and Support F-score or F1 Score is a metric to evaluate a binary classification model on the basis of predictions that are made for the positive class. It is calculated with the help of Precision and Recall. It is a type of single score that represents both Precision and Recall. So, the F1 Score can be calculated as the harmonic mean of both precision and Recall, assigning equal weight to each of them. The formula for calculating the F1 score is given below:

When to use F-Score? As F-score make use of both precision and recall, so it should be used if both of them are important for evaluation, but one (precision or recall) is slightly more important to consider than the other. For example, when False negatives are comparatively more important than false positives, or vice versa.

AUC(Area Under the Curve)-ROC ROC curve In Machine Learning, only developing an ML model is not sufficient as we also need to see whether it is performing well or not. It means that after building an ML model, we need to evaluate and validate how good or bad it is, and for such cases, we use different Evaluation Metrics. AUC-ROC curve is such an evaluation metric that is used to visualize the performance of a classification model. It is one of the popular and important metrics for evaluating the performance of the classification model. AUC-ROC curve is a performance measurement metric of a classification model at different threshold values.

Firstly, let's understand ROC (Receiver Operating Characteristic curve) curve. ROC or Receiver Operating Characteristic curve represents a probability graph to show the performance of a classification model at different threshold levels. The curve is plotted between two parameters, which are:

- True Positive Rate or TPR
- False Positive Rate or FPR

In the curve, TPR is plotted on Y-axis, whereas FPR is on the X-axis.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science

TPR: TPR or True Positive rate is a synonym for Recall, which can be calculated as: FPR or False Positive Rate can be calculated as: Here, TP: True Positive FP: False Positive TN: True Negative FN: False Negative Now, to efficiently calculate the values at any threshold level, we need a method, which is AUC. AUC: Area Under the ROC curve

AUC is known for Area Under the ROC curve. As its name suggests, AUC calculates the two-dimensional area under the

entire ROC curve ranging from (0,0) to (1,1), as shown below image: Figure-3.6 ROC curve Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science In the ROC curve, AUC computes the performance of the binary classifier across different thresholds and provides an aggregate measure. The value of AUC ranges from 0 to 1, which means an excellent model will have AUC near 1, and hence it will show a good measure of Separability.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System UNIT-II Resources and Components of Information Systems, Managing Information Resources in an Organization, IT infrastructure and upcoming technologies, Integration and automation of Business Functions and developing Business Models. Fundamentals of Data Processing. Resources and Components of Information Systems, Resources of Information System - An Information System is generally integrated and co-ordinate a network of components, which combine together to convert data into information. The information system provides access of information when it is required. The information system is a set of the component which generally helps a system. Resources of Information System: There are 5 resources of information system which are given below: 1. Hardware: The system components which can physically touch – the system unit (tower, desktop, laptop), internal devices and peripheral devices (keyboards and monitors) – are called hardware and it is important to remember that basic definition: The hardware are the parts of the computer that are tangible and can be touched. Peripheral devices are provided in many other ways, but think of them as hardware that surrounds the system unit. These peripherals devices may be connected by wired or wireless technology to the system unit. Generally peripherals devices communicate with the interior components of the system unit via installed software. The software itself is intangible and can't be touched physically. 2. Software: We know that, the hardware needs to know what to do, and that is the role of software. The software may be divided into two types: first system software and second application software. Primary piece of system software is the operating system, such as Windows or IOS, which manages the hardware's operation. Application software is performing for specific tasks, such as handling a spreadsheet, creating a document, or designing a Web page. 3. People: The human element is the most important component of information system and the people that are needed to run the system and the procedures they follow so that the knowledge in the huge databases and data warehouses can be turned into learning that can interpret what has happened in the past and guide future action. 4. Data: Data is one of the most important component which is generally store in form of information in a database system and a database is a place where data is collected and from which it can be retrieved by querying it using one or more specific criteria. All types of data store in warehouse without knowing whatever form that an organization needs. The databases and data warehouses have assumed even greater importance in information systems with the emergence of "big data," a term for the truly massive amounts of data that can be collected and analyzed. 5. Network: The network is defined as a system in which more than the system is connected through a transmission media. It provides an interface to receive a piece of information or send an information. It is also one of the best resources in the information system. Components of Information System An Information system is a combination of hardware and software and telecommunication networks that people build to collect, create and distribute useful data, typically in an organization. It defines the flow of information within the system. The objective of an information system is to provide appropriate information to the user, to gather the data, process the data and communicate information to the user of the system. Components of the information system are as follows: 1. Computer Hardware: Physical equipment used for input, output and processing. The hardware structure depends upon the type and size of the organization. It consists of an input and an output device, operating system, processor, and media devices. This also includes computer peripheral devices. 2. Computer Software: The programs/ application program used to control and coordinate the hardware components. It is used for analysing and processing of the data. These programs include a set of instruction used for processing information. Software is further classified into 3 types: 1. System Software 2. Application Software 3. Procedures 3. Databases: Data are the raw facts and figures that are unorganized that are later processed to generate information. Softwares are used for organizing and serving data to the user, managing physical storage of media and virtual resources. As the hardware can't work without software the same as software needs data for processing. Data are managed using Database management system. Database software is used for efficient access for required data, and to manage knowledge bases. 4. Network: • Networks resources refer to the telecommunication networks like the intranet, extranet and the internet. • These resources facilitate the flow of information in the

organization. • Networks consists of both the physical devices such as networks cards, routers, hubs and cables and software such as operating systems, web servers, data servers and application servers. • Telecommunications networks consist of computers, communications processors, and other devices interconnected by communications media and controlled by software. • Networks include communication media, and Network Support.

5. Human Resources: It is associated with the manpower required to run and manage the system. People are the end user of the information system, end-user use information produced for their own purpose, the main purpose of the information system is to benefit the end user. The end user can be accountants, engineers, salespersons, customers, clerks, or managers etc. People are also responsible to develop and operate information systems. They include systems analysts, computer operators, programmers, and other clerical IS personnel, and managerial techniques.

Managing Information Resources in an Organization, Information Resources Management (IRM) is the planning, budgeting, organizing, directing, training, and control associated with an organization's information. The term encompasses both information itself and the related resources, such as personnel, equipment, funds, and technology.

IT infrastructure and upcoming technologies, AI will be at the heart of the new IT infrastructure. From our survey, it is clear that cloud computing will simplify today's complex state of infrastructure management (see Figure 3, next page). In fact, 61% of organizations consider cloud to be a key enabler of driving business priorities. Emerging digital technologies such as automation, containerization, artificial intelligence, IoT, edge and ambient computing are the triggers for innovation and new business models. Expect an even more dramatic shift as 5G adoption grows, making networks more reliable and secure. When deployed in the right combinations, these technologies have the power to enhance capacity and reduce maintenance and operating costs. The underlying implication of this goes beyond enabling people and processes with technology. IT infrastructure has a new role to play in improving user experience, supporting organizational goals, meeting regulatory requirements, and extracting value from investments that can go straight to the bottom line.

Technologies such as hybrid cloud, IoT and edge computing will change IT infrastructure provisioning and management. The collection and analysis of real-time data using IoT, for example, will ensure that just the right maintenance is undertaken at the right time and organizations have constant insight into the remain value of their assets. Over the next two or three years, a vast amount of enterprise infrastructure will be deployed at the edge. By some estimates, about 50% of new infrastructure will be at the edge. Much of this will have processing capability. Disasters like COVID-19 have helped to hasten the cycles of adoption for edge infrastructure. Sensors at the edge—say in automobiles, oil rig equipment, farm equipment, manufacturing equipment, utility meters, cobots(robot), scanners, headsets, wearable's—will drive the rise of next-generation IT assets. However, the truth is that most organizations are unprepared—or cannot fully leverage these digital technologies. To prepare, modernization of the underlying infrastructure is an urgent need.

IT Infrastructure Trends in 2023

IT Infrastructure is the backbone of today's digital business milieu. It comprises various hardware, software solutions, and network connections that empower organizations and ensure the smooth running of applications prerequisite for business functioning. The aim of IT infrastructure is to improve communication between devices, tablets, laptops, desktop PCs, servers, and cloud storage. If an IT infrastructure is reliable, relevant, and secure, it can help enterprises meet their goals and gain a competitive edge in the market. Alternatively, if it's not setup properly, businesses can face productivity, connectivity, and security issues.

2023 is shaping to be a big year for the IT infrastructure, considering the many changes and shifts it will incur centering around increasing business efficiency, sustainability, and cost optimization. Here are the top IT infrastructure trends that will dominate the year 2023:

1. AI & Automation will be Leveraged AI and Automation in IT infrastructure will take center stage in the upcoming year owing to their unparalleled efficiency and highly productive outcomes. Artificial Intelligence is a machine programmed in such a way that it can effectively resolve convoluted problems and deliver results in real-time without bothering humans. IT companies can benefit significantly by embracing AI-based infrastructure. They can perform tasks from eradicating security threats to automating data capture and streamlining compliance. At the same time, AI algorithms can easily recognize failures in storage, power, network connections, etc. This early prediction can help avoid an enterprise's system crash and allow them to deliver productive results faster.
2. Sustainable Technology will be in Demand Sustainable technology is an umbrella term that describes technology that has positive environmental, social, and governance (ESG) outcomes for the enterprise and its customers. Its innovation involves selecting and working with the right tools, hardware, and software to help companies deliver maximum results using minimum resources. Furthermore, these technologies aim to reduce environmental and ecological threats to create a sustainable product for customers.
3. Increase of Investment in SASE Secure Access Service Edge, or

SASE, is a network architecture that combines software-defined wide area networking (SD-WAN) and security functions into a unified cloud service. On that ground, SASE enables employees to safely connect to internal applications from anywhere and allows organizations better control over data and traffic that comes and goes from their internal network. Moreover, by filtering URLs, DNS queries, and other incoming and outgoing network traffic, SASE helps prevent data exfiltration, malware-based threats and other attacks on corporate data.

4. Attention to Platform Engineering Platform engineering is another growing IT Infrastructure Trends meant to aid software developers in dealing with the complexity of developing an application. As the industry has started evolving because of technological advancements, it's becoming difficult for developers to come up with a competent, strong, and secure application meeting the vivid demands of clients. It is designed and managed by a diligent product team for software engineering companies to accelerate the delivery of applications. It refines developer experience and productivity by bestowing self-service capabilities with automated infrastructure operations.

5. Just in Time Infrastructure Just-in-time infrastructure is a rising trend to help enterprises meet the rising demands of consumers. Enterprises today are under extreme pressure to create and deliver new digital products or services quickly to meet the needs of customers and internal users. To that end, attention is given to accelerating development and deployment cycles. Still, those efforts won't yield any expected results when the infrastructure is under-prepared or unavailable to support those efforts.

6. Industry Cloud Platforms Industry cloud platforms are an alternative to various cloud offerings purchased by companies as they provide a pre-integration solution that concurs with specific vertical market needs. It is a combination of traditional cloud services customized to fit a specific industry's needs to create value within the bounds of the market it is utilized rather than expanding it to another. Industry clouds also assist enterprises in meeting regulations specific to their geographic location.

7. Edge-Computing will Emerge as a Central Focus Edge computing is another trend that will dominate IT infrastructure in the coming years. Gartner has outlined that by 2025 more than 50% of data will be processed and created outside the cloud or data center. Edge computing is related to processing data closer to where it is being made, increasing processing speed and volume generated, resulting in high actionable results in real time.

8. High Demand for Skilled Manpower with Business Acumen

As digitalization continues to flourish, there is a profuse demand for various skills within the IT industry. Yet, a limited pool of talent is available for high-demand effective skillset – competence in the cloud, analytics, and automation. As a result, a new trend in 2023 is also focusing on hiring new I&O team members with a more business mindset and a technology-enriched background. Hence, they can address the business benefits of an application and how it fulfills an organization's goals. This also ensures that more effective and strategic business decisions are made.

Integration and automation of Business Functions and developing Business Models. Business integration seeks to use automation and orchestration to connect and streamline business processes and workflows. This uses IT to speed up the business, save money and bring more consistency with fewer errors to business operations. Business integration is a strategy whose goal is to synchronize IT and business cultures and objectives and align technology with business strategy and goals. Business integration is a reflection of how IT is being utilized as a function of business. Business integration has many implications for the role of the corporate CIO (Chief Information Officer), one of which is that the CIO will be taking on additional responsibilities such as business process management. In the past, the CIO was mainly responsible for IT processes. As technology increasingly becomes an embedded business function, IT decisions and leadership will fall under the domain of business leaders instead of technology experts. Computers, storage, networks and other IT resources have long been treated as a separate layer or a technological endeavor used to digitize the traditional human processes of running a business. For example, rather than writing and handling an order on paper, the order was entered and handled using computers. For many decades, IT basically displaced or augmented traditional human processes, but wasn't seen as bringing much more to the business. Thus, IT was considered an unavoidable cost center or business expense, as computers and storage replaced paper documents and file cabinets. But IT had no direct effect on any aspect of the business or the products and services the business delivered.

- Alignment. IT isn't a corporate science experiment. Technology isn't procured and deployed for its own sake. IT serves the business and business integration aligns IT with business goals.
- Services. IT isn't a room full of expensive hardware. Business integration seeks to transform those assets into business services. For example, a data center server may instead be regarded as an application platform where a business leader can deploy a new software application.
- Acceleration. IT is least efficient when it shadows traditional human actions. Business integration seeks to use automation and orchestration to connect and streamline business processes and workflows. This uses IT to speed up the business, save money and bring more consistency with fewer errors to business operations.
- Innovation. IT can

enable business leaders and employees to do things that are more creative and competitive. Business integration seeks to use IT to enable new opportunities for business. As one example, a business might employ internet of things devices and artificial intelligence to enhance product transportation or other supply chain activity. Benefits of business integration Business integration can provide a range of benefits to the business, including the following:

- Optimization. Business integration can reduce costs and operational bottlenecks by creating streamlined and optimized processes or workflows.
- Cloud utilization. There are countless options for cloud utilization and businesses integration can enhance operations by moving some infrastructure and workloads into the cloud, including IaaS, PaaS and SaaS cloud resources.
- Preserve legacy systems. Many businesses face compelling reasons to use older legacy workloads. Business integration can help to make legacy workloads interoperable with other newer workloads and workflows, enabling legacy systems to continue working while replacement systems are developed and refined.
- Innovation. Business integration can help a business drive digital transformation and create new assets and services, enabling the business to innovate and compete in ways never before possible. One example is creating a private cloud to facilitate a strong software development team.
- Agility. Business integration can help to make the business more responsive and agile, enabling employees and customers to access data, place and track orders, or request service as part of a well-established workflow.
- Consistency. Traditional manual business processes are prone to errors and oversights that can damage customer satisfaction and even violate compliance requirements. Business integration can bring orchestration and automation that helps eliminate such problems and delivers repeatable, auditable work.

Disadvantages of business integration While business integration is an essential need with benefits for most modern digital organizations, there are also several potential disadvantages that require consideration such as the following:

- Requires understanding. Business integration doesn't happen automatically. Digitizing, orchestrating and automating complex business processes and workflows demands a deep understanding of the business and its security, compliance and operational needs. Otherwise, it's easy to create and orchestrate errors and inefficiency into the processes.
- Less flexible. While business integration can accelerate the pace of everyday business, the use of orchestration and automation demand guardrails to ensure that events occur in predictable, repeatable patterns. There's less accommodation for unforeseen or special circumstances.
- Change. Any business integration initiative must include ample provisions for change as business needs and legal or regulatory requirements shift over time. This usually involves regular assessments of the business integration environment and a mechanism to approve and implement changes as required.

**B2B integration** The meaning of business integration can extend beyond internal resources, services, processes and workflows. The notion of business integration increasingly involves the integration of two or more independent business entities -- sometimes referred to as business-to-business (B2B) or supply chain integration. In simple terms, B2B integration is the use of IT to connect systems, data and processes that enable one business to interoperate or integrate with another to create a digital ecosystem. As one simple example, company A places an order with company B. Company B completes and ships the order to company A using the services of company C. Company B updates company A on the status of the order, and company A can track the location and status of the delivery through company C. Such interactions require a new level of business integration capable of exchanging data and communications using APIs, integration software such as supply chain management platforms, and other tools. It's worth noting that the term business integration is also used in discussing mergers and acquisitions. In this context, businesses are being blended together -- or integrated -- but this level of integration relies mainly on human negotiation and contractual agreement rather than IT as the primary mechanism for success.

**Automation of Business Functions and developing Business Models** Business automation tools help companies and their customers by automating repetitive, day-to-day tasks. They free up employees to focus on more strategic projects and provide an auditable trail of data that teams can use to make more informed decisions and apply consistent controls. Companies of all sizes can apply business automation to myriad tasks, projects and processes. The key benefits usually include time and cost savings, elimination of errors and setting up controls to ensure that policies are followed.

**How Do Business Automation Tools Work?** Business automation tools use technology to take the manual labor out of day-to-day business processes. From human resources to sales to accounting, nearly every corner of a business's operations can benefit from business automation. Using business automation, companies can both eliminate the need for manual labor while improving and simplifying the individual steps that make up different processes. For example, by automating the preliminary job candidate selection process, companies can save significant man hours that would have otherwise been spent giving all received applications an initial review.

**Types of Business Automation** By making certain business processes "automatic," business automation

eliminates repetitive tasks, reduces hours wasted on redundant tasks, and helps improve overall productivity. For these and other reasons, a growing number of organizations are infusing more and more business automation into their operations. Here are four types of business automation, how they work and when they should be used.

**Marketing Automation** Marketing is an important business activity that can be both laborious and costly, making it ripe for simplification through automation. Through marketing automation tools (which usually take the form of software), companies can generate highly qualified leads that are ready for sales engagement. These tools also provide a framework for teams to target, build, execute and measure the success of marketing campaigns—taking the complexity out of lead qualification and conversion. Marketing automation is useful for companies of all sizes. For example, a smaller company may use the software to develop, generate and send out monthly emails to convey relevant content or offers to a client distribution list. This process can significantly reduce the number of hours spent on customer “touches” over the course of a year. By automating online marketing and lead generation campaigns, companies can reduce the cost of developing and running these campaigns. This, in turn, helps to create a higher measurable return on investment (ROI) for each of those campaigns.

**Accounting and Bookkeeping Automation** By automating their accounting and bookkeeping functions, companies can save considerable time on accounts receivable (AR), accounts payable (AP), billing, collections, credit card applications, data backup and other financial processes that have to be managed on a daily or weekly basis. They can also apply automation to core processes like closing the books, general ledger (GL) management and bank account management. By removing manual elements from the accounting team’s work—and handling the number-crunching and transactional work—automation makes a complex process more manageable. Take accounts payable management, for example. About 55% of companies still handle their AP processes manually. Using an automated system for this specific area of business finance management saves money and time: Data capture is automated, invoices are automatically matched to documents, and approvals are electronically routed. It also reduces data errors and helps prevent fraud through a system of “touchless” controls that happen behind the scenes. For companies of all sizes, accounting is a time-consuming process that includes many manual steps. By automating some or all of those steps, companies can free up time for important tasks like analysis, strategy and collaboration among team members.

**Process Automation** Business process automation (BPA) goes beyond basic automation and incorporates integration of applications to help companies improve value and efficiency. A subset of BPA, robotic process automation (RPA) focuses on automating routine tasks, while BPA helps companies get more out of their automation investments. BPA does this by aggregating data across multiple sources to develop analysis that would be difficult to attain manually. Companies are using BPA for functions like:

- ♣ Automated order entry ♣ Email automation ♣ Automated batch processing ♣ Automated file transfers ♣ Automated report generation and distribution
- HR Automation Hiring new employees is a multi-step process that starts with an online job ad or recruitment effort and ends when the employee is officially on boarded. Many steps in this process can be automated. A human resources management system (HRMS) is a valuable tool in doing so. As part of a broader set of functionalities, these systems automate the candidate management process. This relates specifically to automated employment offers sent directly to candidates, which helps share roles to fill to both the outside world and current employees who may wish to apply for internal jobs or make referrals. Other HR tasks that software can manage include:
- ♣ Employee record retention and retrieval.
- ♣ Reviews of job applications submitted online.
- ♣ Distribution and signing of work contracts, confidentiality agreements, waivers and other newemployee documentation.
- ♣ Employee tax form management.
- ♣ Benefits enrollment eligibility.
- ♣ Training requirements (e.g., when an employee moves into a new position).

**Fundamentals of Data Processing:** Data processing, manipulation of data by a computer. It includes the conversion of raw data to machine-readable form, flow of data through the CPU and memory to output devices, and formatting or transformation of output. Any use of computers to perform defined operations on data can be included under data processing. Data processing occurs when data is collected and translated into usable information. Usually performed by a data scientist or team of data scientists, it is important for data processing to be done correctly as not to negatively affect the end product or data output. Data processing starts with data in its raw form and converts it into a more readable format (graphs, documents, etc.), giving it the form and context necessary to be interpreted by computers and utilized by employees throughout an organization.

**Six stages of data processing**

1. Data collection Collecting data is the first step in data processing. Data is pulled from available sources, including data lakes and data warehouses. It is important that the data sources available are trustworthy and well-built so the data collected (and later used as information) is of the highest possible quality.
2. Data preparation Once the data is collected, it then enters the data

preparation stage. Data preparation, often referred to as “pre-processing” is the stage at which raw data is cleaned up and organized for the following stage of data processing. During preparation, raw data is diligently checked for any errors. The purpose of this step is to eliminate bad data (redundant, incomplete, or incorrect data) and begin to create high-quality data for the best business intelligence.

3. Data input The clean data is then entered into its destination (perhaps a CRM like Sales force or a data warehouse like (Redshift), and translated into a language that it can understand. Data input is the first stage in which raw data begins to take the form of usable information.

4. Processing During this stage, the data inputted to the computer in the previous stage is actually processed for interpretation. Processing is done using machine learning algorithms, though the process itself may vary slightly depending on the source of data being processed (data lakes, social networks, connected devices etc.) and its intended use (examining advertising patterns, medical diagnosis from connected devices, determining customer needs, etc.).

5. Data output/interpretation The output/interpretation stage is the stage at which data is finally usable to non-data scientists. It is translated, readable, and often in the form of graphs, videos, images, plain text, etc.). Members of the company or institution can now begin to self-serve the data for their own data analytics projects.

6. Data storage The final stage of data processing is storage. After all of the data is processed, it is then stored for future use. While some information may be put to use immediately, much of it will serve a purpose later on. Plus, properly stored data is a necessity for compliance with data protection legislation like GDPR. When data is properly stored, it can be quickly and easily accessed by members of the organization when needed.

**Data Processing Cycle**

The data processing cycle consists of a series of steps where raw data (input) is fed into a system to produce actionable insights (output). Each step is taken in a specific order, but the entire process is repeated in a cyclic manner. The first data processing cycle's output can be stored and fed as the input for the next cycle, as the illustration below shows us.

**Figure- Data processing cycle (source)**

Type

Uses Batch Processing Data is collected and processed in batches. Used for large amounts of data. Eg: payroll system

Real-time Processing Data is processed within seconds when the input is given. Used for small amounts of data. Eg: withdrawing money from ATM

Online Processing Data is automatically fed into the CPU as soon as it becomes available. Used for continuous processing of data. Eg: barcode scanning

Multiprocessing Data is broken down into frames and processed using two or more CPUs within a single computer system. Also known as parallel processing. Eg: weather forecasting

Time-sharing Allocates computer resources and data in time slots to several users simultaneously.

**Data Processing Methods**

There are three main data processing methods - manual, mechanical and electronic.

**Manual Data Processing**

This data processing method is handled manually. The entire process of data collection, filtering, sorting, calculation, and other logical operations are all done with human intervention and without the use of any other electronic device or automation software. It is a low-cost method and requires little to no tools, but produces high errors, high labor costs, and lots of time and tedium.

**Mechanical Data Processing**

Data is processed mechanically through the use of devices and machines. These can include simple devices such as calculators, typewriters, printing press, etc. Simple data processing operations can be achieved with this method. It has much lesser errors than manual data processing, but the increase of data has made this method more complex and difficult.

**Electronic Data Processing**

Data is processed with modern technologies using data processing software and programs. A set of instructions is given to the software to process the data and yield output. This method is the most expensive but provides the fastest processing speeds with the highest reliability and accuracy of output.

**Examples of Data Processing**

Data processing occurs in our daily lives whether we may be aware of it or not. Here are some real-life examples of data processing:

- A stock trading software that converts millions of stock data into a simple graph
- An e-commerce company uses the search history of customers to recommend similar products
- A digital marketing company uses demographic data of people to strategize location-specific campaigns
- A self-driving car uses real-time data from sensors to detect if there are pedestrians and other cars on the road

\*\*\*\*\*

Analysis of the existing system, Analysis of new requirements; System Development; System Implementation; Factors responsible for success and failure of Information Systems. Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality Softwares. The SDLC aims to produce high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates. • SDLC is the acronym of Software Development Life Cycle. • It is also called as Software Development Process. • SDLC is a framework defining tasks performed at each step in the software development process. • ISO/IEC12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software. What is SDLC? SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process. The following figure is a graphical representation of the various stages of a typical SDLC. A typical Software Development Life Cycle consists of the following stages – Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System Stage 1: Planning and Requirement Analysis Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas. Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks. Stage 2: Defining Requirements Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle. Stage 3: Designing the Product Architecture SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification. This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product. A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS. Stage 4: Building or Developing the Product In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle. Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed. Stage 5: Testing the Product This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS. Stage 6: Deployment in the Market and Maintenance Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UATUser acceptance testing). Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base. SDLC Models There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models". Each process model follows a Series of steps unique to its type to ensure success in the process of software development. Following are the most important and popular SDLC models followed in the industry – • Waterfall

Model • Iterative Model • Spiral Model • V-Model • Big Bang Model Other related methodologies are Agile Model, RAD Model (Rapid Application Development) and Prototyping Models. SDLC - Waterfall Model The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linearsequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System The Waterfall model is the earliest SDLC approach that was used for software development. The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap. AD Waterfall Model - Design Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially. The following illustration is a representation of the different phases of the Waterfall Model. The sequential phases in Waterfall model are –

- Requirement Gathering and analysis – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- System Design – The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System
- Implementation – with inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- Integration and Testing – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- Deployment of system – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- Maintenance – There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment. All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap. Waterfall Model - Application Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are –
- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Waterfall Model - Advantages The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order. Some of the major advantages of the Waterfall Model are as follows –

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Waterfall Model - Disadvantages The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not welldocumented or thought upon in the concept stage. The major disadvantages of the Waterfall Model are as follows –

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot

accommodate changing requirements. • Adjusting scope during the life cycle can end a project. • Integration is done as a "big-bang" at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

SDLC - Iterative Model Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed. An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model. Iterative Model - Design Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental). The following illustration is a representation of the Iterative and Incremental model – Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time." This process may be described as an "evolutionary acquisition" or "incremental build" approach." In this incremental model, the whole requirement is divided into various builds. During each iteration, the development module goes through the requirements, design, implementation and testing phases. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is ready as per the requirement. The key to a successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification & testing of each version of the software against those requirements Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System within each cycle of the model. As the software evolves through successive cycles, tests must be repeated and extended to verify each version of the software. Iterative Model - Application Like other SDLC models, Iterative and incremental development has some specific applications in the software industry. This model is most often used in the following scenarios –

- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- There is a time to the market constraint.
- A new technology is being used and is being learnt by the development team while working on the project.
- Resources with needed skill sets are not available and are planned to be used on contract basis for specific iterations.
- There are some high-risk features and goals which may change in the future.

Iterative Model - Pros and Cons The advantage of this model is that there is a working model of the system at a very early stage of development, which makes it easier to find functional or design flaws. Finding issues at an early stage of development enables to take corrective measures in a limited budget. The disadvantage with this SDLC model is that it is applicable only to large and bulky software development projects. This is because it is hard to break a small software system into further small serviceable increments/modules. The advantages of the Iterative and Incremental SDLC Model are as follows –

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System

- Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
- Easier to manage risk - High risk part is done first.
- With every increment, operational product is delivered.
- Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
- Risk analysis is better.
- It supports changing requirements.
- Initial Operating time is less.
- Better suited for large and mission-critical projects.
- During the life cycle, software is produced early which facilitates customer evaluation and feedback.

The disadvantages of the Iterative and Incremental SDLC Model are as follows –

- More resources may be required.
- Although cost of change is lesser, but it is not very suitable for changing requirements.
- More management attention is required.
- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
- Defining increments may require definition of the complete system.
- Not suitable for smaller projects.
- Management

complexity is more. • End of project may not be known which is a risk. • Highly skilled resources are required for risk analysis. • Projects progress is highly dependent upon the risk analysis phase.

**SDLC - Spiral Model**

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral.

**Spiral Model - Design**

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

**Identification** This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase. This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral, the product is deployed in the identified market.

**Design** The Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and the final design in the subsequent spirals.

**Construct or Build** The Construct phase refers to production of the actual software product at every spiral. In the baseline spiral, when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback. Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to the customer for feedback.

**Evaluation and Risk Analysis** Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback. The following illustration is a representation of the Spiral Model, listing the activities in each phase.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System

Based on the customer evaluation, the software development process enters the next iteration and subsequently follows the linear approach to implement the feedback suggested by the customer. The process of iterations along the spiral continues throughout the life of the software.

**Spiral Model Application** The Spiral Model is widely used in the software industry as it is in sync with the natural development process of any product, i.e. learning with maturity which involves minimum risk for the customer as well as the development firms. The following pointers explain the typical uses of a Spiral Model –

- When there is a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which is usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.

• Significant changes are expected in the product during the development cycle.

**Spiral Model - Pros and Cons**

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System

The advantage of spiral lifecycle model is that it allows elements of the product to be added in, when they become available or known. This assures that there is no conflict with previous requirements and design. This method is consistent with approaches that have multiple software builds and releases which allows making an orderly transition to a maintenance activity. Another positive aspect of this method is that the spiral model forces an early user involvement in the system development effort. On the other side, it takes a very strict management to complete such products and there is a risk of running the spiral in an indefinite loop. So, the discipline of change and the extent of taking change requests are very important to develop and deploy the product successfully.

The advantages of the Spiral SDLC Model are as follows –

- Changing requirements can be accommodated.
- Allows extensive use of prototypes.
- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.

The disadvantages of the Spiral SDLC Model are as follows –

- Management is more complex.
- End of the project may not be known early.
- Not suitable for small or low risk projects and could be expensive for small projects.
- Process is complex
- Spiral may go on indefinitely.
- Large number of intermediate stages requires excessive documentation.

**SDLC - V-Model**

The V-model is an SDLC model where execution of processes happens in a sequential manner in a Vshape. It is also known as Verification and Validation model. The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding

development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase. V-Model - Design Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System Under the V-Model, the corresponding testing phase of the development phase is planned in parallel. So, there are Verification phases on one side of the 'V' and Validation phases on the other side. The Coding Phase joins the two sides of the V-Model. The following illustration depicts the different phases in a V-Model of the SDLC.

**V-Model - Verification Phases**

There are several Verification phases in the V-Model, each of these are explained in detail below.

**Business Requirement Analysis**

This is the first phase in the development cycle where the product requirements are understood from the customer's perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. This is a very important activity and needs to be managed well, as most of the customers are not sure about what exactly they need. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.

**System Design**

Once you have the clear and detailed product requirements, it is time to design the complete system. The system design will have the understanding and detailing the complete hardware and communication setup for the product under development. The system test plan is developed based on the system design. Doing this at an earlier stage leaves more time for the actual test execution later.

**Architectural Design**

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. The system design is broken down further into modules taking up different functionality. This is also referred to as High Level Design (HLD). The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With this information, integration tests can be designed and documented during this stage.

**Module Design**

In this phase, the detailed internal design for all the system modules is specified, referred to as Low Level Design (LLD). It is important that the design is compatible with the other modules in the system architecture and the other external systems. The unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage. These unit tests can be designed at this stage based on the internal module designs.

**Coding Phase**

The actual coding of the system modules designed in the design phase is taken up in the Coding phase. The best suitable programming language is decided based on the system and architectural requirements. The coding is performed based on the coding guidelines and standards. The code goes through numerous code reviews and is optimized for best performance before the final build is checked into the repository.

**Validation Phases**

The different Validation Phases in a V-Model are explained in detail below.

**Unit Testing**

Unit tests designed in the module design phase are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.

**Integration Testing**

Integration testing is associated with the architectural design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.

**System Testing**

System testing is directly associated with the system design phase. System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during this system test execution.

**Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System Acceptance Testing**

Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non-functional issues such as load and performance defects in the actual user environment.

**V- Model – Application**

V- Model application is almost the same as the waterfall model, as both the models are of sequential type. Requirements have to be very clear before the project starts, because it is usually expensive to go back and make changes. This model is used in the medical development field, as it is strictly a disciplined domain. The following pointers are some of the most suitable scenarios to use the V-Model application.

- Requirements are well defined, clearly documented and fixed.
- Product definition is stable.
- Technology is not dynamic and is well understood by the project team.
- There are no ambiguous or undefined requirements.
- The project is short.

**V-Model - Pros and Cons**

The advantage of the V-Model method is that it is very easy to understand and apply. The simplicity of this model also makes it easier to manage. The disadvantage is that the model is not flexible to changes and

just in case there is a requirement change, which is very common in today's dynamic world, it becomes very expensive to make the change. The advantages of the V-Model method are as follows – • This is a highly-disciplined model and Phases are completed one at a time. • Works well for smaller projects where requirements are very well understood. • Simple and easy to understand and use. • Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process. The disadvantages of the V-Model method are as follows – • High risk and uncertainty. • Not a good model for complex and object-oriented projects. • Poor model for long and ongoing projects. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System • Not suitable for the projects where requirements are at a moderate to high risk of changing. • Once an application is in the testing stage, it is difficult to go back and change a functionality. • No working software is produced until late during the life cycle. SDLC - Big Bang Model The Big Bang model is an SDLC model where we do not follow any specific process. The development just starts with the required money and efforts as the input, and the output is the software developed which may or may not be as per customer requirement. This Big Bang Model does not follow a process/procedure and there is a very little planning required. Even the customer is not sure about what exactly he wants and the requirements are implemented on the fly without much analysis. Usually this model is followed for small projects where the development teams are very small. Big Bang Model – Design and Application The Big Bang Model comprises of focusing all the possible resources in the software development and coding, with very little or no planning. The requirements are understood and implemented as they come. Any changes required may or may not need to revamp the complete software. This model is ideal for small projects with one or two developers working together and is also useful for academic or practice projects. It is an ideal model for the product where requirements are not well understood and the final release date is not given. Big Bang Model - Pros and Cons The advantage of this Big Bang Model is that it is very simple and requires very little or no planning. Easy to manage and no formal procedure are required. However, the Big Bang Model is a very high risk model and changes in the requirements or misunderstood requirements may even lead to complete reversal or scraping of the project. It is ideal for repetitive or small projects with minimum risks. The advantages of the Big Bang Model are as follows – • This is a very simple model • Little or no planning required • Easy to manage • Very few resources required • Gives flexibility to developers • It is a good learning aid for new comers or students.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System The disadvantages of the Big Bang Model are as follows – • Very High risk and uncertainty. • Not a good model for complex and object-oriented projects. • Poor model for long and ongoing projects. • Can turn out to be very expensive if requirements are misunderstood. SDLC - Agile Model Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like – • Planning • Requirements Analysis • Design • Coding • Unit Testing and • Acceptance Testing. At the end of the iteration, a working product is displayed to the customer and important stakeholders. What are the various factors that may lead to failure of an information system? The problems causing information system failure fall into multiple categories. The major problem areas are design, data, cost, and operations. Problems with an information system's design, data, cost, or operations can be evidence of a system failure. Factors responsible for success and failure of Information Systems:- What are the factors of success and failure in information system? Factor analysis identified three things as being closely related to successful information systems: the quality of the information product being supplied; the quality of systems personnel and services; and the knowledge and involvement of systems personnel in the business. What are the factors for successful information systems? The five most important factors based on the average ratings were: (1) accuracy of output, (2) reliability of output, (3) timeliness of output, (4) realization of user requirements, and (5) user's confidence in the systems. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System What are some of the factors that would influence the success or failure of an information systems project implementation? Factors including coordination and communication, information system analysis, project team ability and troubleshooting are mutually influence and affect to decision factor and project goals; decision factor and project goals affect to performance of information system projects. \*\*\*\*\*

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System UNIT-IV Information System Applications: Business Applications, Decision Support Systems (DSS)- Characteristics, Problem Analysis v/s Decision making, DSS applications in E enterprise, Knowledge Management System and Knowledge Based Expert System, Enterprise Model System and E-Business, ERP systems, E-Commerce, Ecommunication; Business Process Reengineering Information Systems Defined Information systems are collections of multiple information resources (e.g., software, hardware, computer system connections, the system housing, system users, and computer system information) to gather, process, store, and disseminate information. Tools such as laptops, databases, networks, and smartphones are examples of information systems. So you're employing an information system! Many people rely on various types of information systems to communicate with friends and family, bank or shop online, or look up information via a search engine. Companies and organizations employ information systems to communicate and work with their customers and suppliers, manage the organization, perform essential business operations, and roll out and maintain marketing campaigns. Six Major Types of Information Systems We have the transaction processing systems (TPS) at the operational level. Next are the office automation systems (OAS) and knowledge work systems (KWS), both working at the knowledge level. Next, the management level has the management information systems (MIS) and decision support systems (DSS), and we conclude with the executive support systems (ESS) at the strategic level. Let's explore the different types of information systems more in-depth.

1. Transaction Processing System (TPS) Transaction processing is essential to helping businesses perform daily operations. Transactions are defined as any activity or event that affects the company, and include things like deposits, withdrawals, shipping, billing customers, order entry, and placing orders. TPS supports these business transactions.

2. Office Automation System (OAS) OAS consists of computers, communication-related technology, and the personnel assigned to perform the official tasks. The OAS covers office transactions and supports official activity at every level in the organization. The official activities are subdivided into managerial and clerical activities. Office automation systems include the following applications:

- Email: The email application also covers file attachments such as audio, video, and documents.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System

- Voice Mail: This application records and stores phone messages into the system's memory and can be retrieved anytime.
- Word Processing: Word processing covers the creation of documents, including memos, reports, letters, and anything else that's printable electronically. The created text can be copied, edited, and stored via word processing commands, and checked for grammar and spelling, line and word counting, and headers and footers.

3. Knowledge Work System (KWS) The KWS is a specialized system that expedites knowledge creation and ensures that the business's technical skills and knowledge are correctly applied. The Knowledge Work System aids workers in creating and disseminating new information using graphics, communication, and document management tools. Here are some examples of KWS:

- Computer-Aided Design Systems (CAD): CAD systems automate design creation and revision via computers and graphics software, especially in the manufacturing and tooling processes.
- Financial Workstations: These systems pull and combine data from many different internal and external sources, covering research reports, market data, and management data. Financial workstations can rapidly analyze huge amounts of financial data and trading situations.
- Virtual Reality Systems: These systems take the CAD system to the next level, using interactive graphics utilities to create realistic computer-generated simulations. VR systems are typically found in scientific, educational, and business circles.

4. Management Information System (MIS) Middle managers handle much of the administrative chores for day-to-day routines and performance monitoring, ensuring that all the work is aligned with the organization's needs. That's why MIS is such a valuable tool. Management Information Systems are specially designed to help middle managers and supervisors make decisions, plan, and control the workflow. The MIS pulls transactional data from various Transactional Processing Systems, compiles the information, and presents it in reports and displays. Additionally, these reports can be produced monthly, quarterly, or annually, although MIS can have more immediate reports (e.g., hourly, daily).

5. Decision Support System (DSS) The DSS is a management-level, interactive computer-based information system that helps managers to make decisions. The Decision Support System specifically gives middle managers the information necessary to make informed, intelligent decisions.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System Decision Support Systems use different decision

models to analyze or summarize large pieces of data into an easy-to-use form that makes it easier for managers to compare and analyze information. Often, these summaries come in the form of charts and tables.

**6. Executive Support System (ESS)** The ESS is like the MIS but for executive-level decision-making. The decisions involve company-wide matters, so the stakes are higher. Consequently, they demand more insight and judgment. The ESS provides greater telecommunication, better computing capabilities, and more efficient display options than the DSS. Executives use ESS to make effective decisions through summarized internal data taken from DSS and MIS and external sources. In addition, executive support systems help monitor performances, track competitors, spot opportunities, and forecast future trends.

**Information System Applications: Business Applications** Currently, information system applications are used in various fields to help facilitate various human activities and the use of technology. With today's sophistication of technology, various kinds of things are easier to do. Information systems have been used in various fields, one of which is corporate business.

**Information System Applications in Daily Life** After knowing the meaning of an information system, the next thing that will be discussed is an example of an information system application. Conscious or not that there are many examples of the application of information systems in everyday life. Examples of these are applied in the fields of education, work, etc., among others.

**1. Information System Application: E-Commerce** First, there is E-Commerce which is included in information systems and applications in the economic and business fields. Every need for buying and selling goods and services that are carried out online through an application based on a website or mobile is one example of e-commerce. Examples of information system applications that use e-commerce such as Shopee, Lazada, TokoPedia, BukaLapak, and so on.

**2. Information System Application: E-Learning** E-learning or electronic learning is an information system that is applied in schools. The application of data that is processed and disseminated through technology.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A)

**Management Information System** There are many benefits from the presence of information system applications in the field of education, including convenience. The convenience offered not only saves time but also saves paper. Therefore this example of an information system is very useful and widely used in everyday life.

**3. Information System Application: Fleet Management System** The fleet management system is an information system application that will assist the process of monitoring the logistics fleet and delivery of goods so that the tracking process becomes more systematic and continues to be centralized. Usually, this system responds to the detected location by using the help of GPS.

**4. Information System Application: Integrated Service System for Students** This student service system was created to facilitate student activities on campus activities. One of the activities is to obtain results from academic activities in the form of tests or other activities. In addition, the application of this information system can also facilitate the registration of the desired courses each semester. Students do not need to come directly to campus to carry out these activities. These students are only enough to open the available applications.

**5. Information System Application: Online Booking** Next is the application of information systems in placing orders online. Reservations that can be made online can be in the form of transportation tickets, concert tickets, and hotels. Apart from these things, many other things can be ordered online, either on a website or an application. The way it works is also very easy, one only needs to choose what things they want to order online. Later that person will get evidence in the form of a successful booking. After the evidence is obtained, the person can immediately exchange the existing ticket.

**6. Information System Application: Office Automation System** The office automation system is the advantage of information system applications that can combine various kinds of information technology equipment on server devices for computer network needs. The main purpose is usually used to simplify the communication process so that it runs more effectively and efficiently.

**7. Information System Application: Transaction Processing System** The last example is the transaction processing system which is an information system for organizations and companies on certain routine business operations. The implementation process of this transaction processing system includes financial transaction activities, re-registration, and various other administrative activities.

**8. Information System Application: Video Call** Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A)

**Management Information System** This is an information system where the application is on the cellphone. The way it works is extraordinary, it's easy to just download an application that provides video call facilities. Then just register via telephone number. In this day and age, many applications provide video call facilities, even though these facilities are already built-in on mobile phones. Usually, this video call facility is integrated with a chat application that uses internet quota as a means. So to make this video call make sure you have an internet connection.

**9. Information System Application: Enterprise Resource Planning** This is an example of a company

management information system that not many ordinary people know about. This information system is widely used by several companies because it facilitates their activities. Many large companies have used this system. This system functions in terms of monitoring or supervision of company management. Furthermore, this system is also connected to the fields of work units in finance, marketing, operations, and others. Things that used to take a long time can now be done in a short time.

**10. Information System Application: Artificial Intelligence**

One of these information systems was previously used by companies. Artificial Intelligence or artificial intelligence greatly facilitates the transaction process in the company. Seeing the sophistication of A.I., it is not surprising that many have applied this to other places. Perhaps the most widely heard and most often heard is that AI is applied to mobile phones. The way this system works is to solve existing problems with the knowledge of the experts who have been included in it. So the system that is processed to produce this information still comes from humans as well.

**Business application**

**Business Applications of Information Systems**

Information systems perform three vital roles in business firms. Business applications of IS support an organization's business processes and operations, business decision-making, and strategic competitive advantage. Major application categories of information systems include operations support systems, such as transaction processing systems, process control systems, and enterprise collaboration systems, and management support systems, such as management information systems, decision support systems, and executive information systems. Other major categories are expert systems, knowledge management systems, strategic information systems, and functional business systems.

However, in the real world most application categories are combined into cross-functional information systems that provide Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System information and support for decision-making and also perform operational information processing activities.

An information system (IS) can be any organized combination of people, hardware, software, communications networks, and data resources that collect, transforms, and disseminate information in an organization.

**Information Technologies:** Business professionals rely on many types of information systems that use a variety of information technologies. For example:

- Types of IS- Manual (paper-and-pencil) information systems-
- Informal (word-of-mouth) information systems-
- Formal (written procedures) information systems-
- Computer-based information systems

Computer-based information systems (IS) use hardware, software, the Internet, and other telecommunications networks, computer-based data resource management techniques, and other forms of information technologies (IT) to transform data resources into a variety of information products for consumers and business professionals.

The role of Information System in an Organization

Information systems perform three vital roles in any type of organization. That is, they support an organization's:

- Business processes and operations
- Decision making by employees and managers
- Strategies for competitive advantage

— Analysing Royal Caribbean International We can learn a lot about the challenges of revitalizing and redirecting information technology in a company from the Real World Case of Royal Caribbean International. Take a few minutes to read it, and we will discuss it (See Royal Caribbean International: Renewing and Realigning IT with Business in Section IX).

**The Major Roles of IS:** Three major roles of the business applications of information systems include:

- Support Business Processes – involves dealing with information systems that support the business processes and operations in a business.
- Support Decision Making – help decision makers to make better decisions and attempt to gain a competitive advantage.
- Support Competitive Advantage – help decision makers to gain a strategic advantage over competitors requires innovative use of information technology.

Information System Implementation

New information technologies offer scholarly publishers an historic opportunity to increase speed and efficiency of production, add convenience for contributors, and enhance value for readers. The implementation of these systems, however, involves substantial risk. Information technology (IT) projects can and often do fall short of their objectives for a variety of reasons, including cost overruns, resistance from staff or external users, and failure of the technology to perform as expected.

**Decision Support Systems (DSS)**

Characteristics

Decision support systems (DSS) are interactive software-based systems intended to help managers in decision-making by accessing large volumes of information generated from various related information systems involved in organizational business processes, such as office automation system, transaction processing system, etc. DSS uses the summary information, exceptions, patterns, and trends using the analytical models. A decision support system helps in decision-making but does not necessarily give a decision itself. The decision makers compile useful information from raw data, documents, personal knowledge, and/or business models to identify and solve problems and make decisions.

**Programmed and Non-programmed Decisions** There are two types of decisions - programmed and non-programmed decisions. Programmed decisions are basically automated processes, general routine work, where -

- These decisions have been taken several times.
- These decisions follow some guidelines or rules. For example, selecting a reorder level for inventories is a programmed decision.
- Non-programmed decisions occur in unusual and non-addressed situations, so -
- It would be a new decision.
- There will not be any rules to follow.
- These decisions are made based on the available information.
- These decisions are based on the manager's discretion, instinct, perception and judgment. For example, investing in a new technology is a non-programmed decision.

Decision support systems generally involve non-programmed decisions. Therefore, there will be no exact report, content, or format for these systems. Reports are generated on the fly.

**Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System**

**Characteristics of a DSS**

- Support for decision-makers in semi-structured and unstructured problems.
- Support for managers at various managerial levels, ranging from top executive to line managers.
- Support for individuals and groups. Less structured problems often require the involvement of several individuals from different departments and organization level.
- Support for interdependent or sequential decisions.
- Support for intelligence, design, choice, and implementation.
- Support for variety of decision processes and styles.
- DSSs are adaptive over time.

**Benefits of DSS**

- Improves efficiency and speed of decision-making activities.
- Increases the control, competitiveness and capability of futuristic decision-making of the organization.
- Facilitates interpersonal communication.
- Encourages learning or training.
- Since it is mostly used in non-programmed decisions, it reveals new approaches and sets up new evidences for an unusual decision.
- Helps automate managerial processes.

**Components of a DSS** Following are the components of the Decision Support System –

- Database Management System (DBMS) – To solve a problem the necessary data may come from internal or external database. In an organization, internal data are generated by a system such as TPS and MIS. External data come from a variety of sources such as newspapers, online data services, databases (financial, marketing, human resources).
- Model Management System – It stores and accesses models that managers use to make decisions. Such models are used for designing manufacturing facility, analyzing the financial health of an organization, forecasting demand of a product or service, etc.
- Support Tools – Support tools like online help; pulls down menus, user interfaces, graphical analysis, error correction mechanism, facilitates the user interactions with the system.

**Classification of DSS** There are several ways to classify DSS. Hoi Apple and Whinstone classifies DSS as follows –

**Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System**

- Text Oriented DSS – It contains textually represented information that could have a bearing on decision. It allows documents to be electronically created, revised and viewed as needed.
- Database Oriented DSS – Database plays a major role here; it contains organized and highly structured data.
- Spreadsheet Oriented DSS – It contains information in spread sheets that allows create, view, modify procedural knowledge and also instructs the system to execute self-contained instructions. The most popular tool is Excel and Lotus 1-2-3.
- Solver Oriented DSS – It is based on a solver, which is an algorithm or procedure written for performing certain calculations and particular program type.
- Rules Oriented DSS – It follows certain procedures adopted as rules.
- Rules Oriented DSS – Procedures are adopted in rules oriented DSS. Export system is the example.
- Compound DSS – It is built by using two or more of the five structures explained above.

**D Types of DSS** Following are some typical DSSs –

- Status Inquiry System – It helps in taking operational, management level, or middle level management decisions, for example daily schedules of jobs to machines or machines to operators.
- Data Analysis System – It needs comparative analysis and makes use of formula or an algorithm, for example cash flow analysis, inventory analysis etc.
- Information Analysis System – In this system data is analyzed and the information report is generated. For example, sales analysis, accounts receivable systems, market analysis etc.
- Accounting System – It keeps track of accounting and finance related information, for example, final account, accounts receivables, accounts payables, etc. that keep track of the major aspects of the business.
- Model Based System – Simulation models or optimization models used for decision-making are used infrequently and creates general guidelines for operation or management.

**Problem-solving** involves identifying an issue, finding causes, asking questions and brainstorming solutions. Gathering facts helps make the solution more obvious. Decision-making is the process of choosing a solution based on your judgment, situation, facts, knowledge or a combination of available data.

**Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System Problem-solving vs.**

decision-making Problem-solving is an analytical method to identify potential solutions to a situation. It's a complex process and judgment calls, or decisions, may have to be made on the way. The primary goal is to find the best solution. Problem-solving involves identifying an issue, finding causes, asking questions and brainstorming solutions. Gathering facts helps make the solution more obvious. Decision-making is the process of choosing a solution based on your judgment, situation, facts, knowledge or a combination of available data. The goal is to avoid potential difficulties. Identifying opportunity is an important part of the decision-making process. Making decisions is often a part of problem-solving. Why problem-solving and decision-making are important Problem-solving and decision-making skills are both important because they can help you navigate a variety of situations that might come up at work. They complement one another and can resolve many of the same issues. Both problem-solving and decision-making involve critical thinking. Problem-solving and decision-making apply to all careers and industries. Because both can help companies by resolving complex situations and problems, employers typically value these skills in job candidates. They show you can think through various scenarios and make sound decisions that are good for the company. For example, a business may have multiple problems that all demand time and resources. A good manager or leader can decide which problems to prioritize. That includes making numerous decisions as part of the problem-solving process and then following through with the steps to fix the problem.

DSS applications in EnterpriseA decision support system (DSS) is a computerized program used to support determinations, judgments, and courses of action in an organization or a business. A DSS sifts through and analyzes massive amounts of data, compiling comprehensive information that can be used to solve problems and in decision-making. Specific uses (Applications) for DSSs in business There are many different ways managers can use DSS software to their advantage. Typically, business planners will build custom DSSs to evaluate specific operations. These include inventory management, in which DSS applications can provide guidance on establishing supply chain movement, and sales, in which DSS software helps managers predict how changes may affect results.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System To manage inventory DSSs are very helpful in evaluating inventory to help a business's cash flow and profitability by predicting demand for particular products and itemizing assets. To aid sales optimization and sales projections Decision support technology can also analyze sales data, make predictions and monitor existing revenue patterns. Planners can use the technology to tackle sales numbers using a variety of decision support resources. To optimize industry-specific systems Other uses for decision support systems include projecting the future of a business or to get a bird's-eye of a company's performance. These insights help owners and managers navigate difficult situations better, especially when they have reliable information to predict expenditures and revenues. Examples of DSSs We all use DSSs in our personal and business lives every day. For example, every time you use Google, you're using a highly sophisticated DSS that organizes a massive amount of information in a searchable, retrievable format. It can locate the specific images, videos and text files you need to help your business achieve more. GPS tracking is another type of DSS. As you can see in our Verizon Connect review, its software allows drivers to determine the best and quickest route between two points while monitoring traffic conditions and helping them avoid congestion. These are some other uses of DSS, including:

- Agriculture: Farmers use DSS tools for crop planning to help them determine the best times for planting, fertilization and harvesting.
- Medicine: Clinical DSS technology has many uses: maintaining research information about chemotherapy protocols, preventive and follow-up care, and monitoring medication orders. DSSs are also used with medical diagnosis software.
- Weather forecasting: Some states use DSSs to provide information about potential future hazards such as floods. To do this, they factor in real-time weather conditions, floodplain boundaries information and historic county flood data.
- Real estate: Real estate companies use DSSs to manage data on comparable home prices and acreage.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System

- Education: Universities and colleges use DSSs to know how many students they currently have enrolled. This helps them predict how many students will register for particular courses or whether the student population is sufficient to meet the university's costs.
- Knowledge Management System and Knowledge Based Expert System, Definition of KMSA knowledge management system comprises a range of practices used in an organization to identify, create, represent, distribute, and enable adoption to insight and experience. Such insights and experience comprise knowledge, either embodied in individual or embedded in organizational processes and practices. Purpose of KMS
- Improved performance
- Competitive advantage
- Innovation
- Sharing of knowledge
- Integration
- Continuous improvement by
- Driving strategy
- Starting new lines of business
- Solving problems faster
- Developing professional

skills o Recruit and retain talent Activities in Knowledge Management • Start with the business problem and the business value to be delivered first. • Identify what kind of strategy to pursue to deliver this value and address the KM problem. • Think about the system required from a people and process point of view. • Finally, think about what kind of technical infrastructure are required to support the people and processes. • Implement system and processes with appropriate change management and iterative staged release.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System

Level of Knowledge Management Knowledge Based Expert System

Knowledge-based systems are a form of artificial intelligence (AI) designed to capture the knowledge of human experts to support decision-making. An expert system is an example of a knowledge-based system because it relies on human expertise. One of the principal members of the AI group is Knowledge-Based Expert System (KBES). With the advancement of computing facilities and other resources, the focus has shifted to more demanding tasks (which require intelligence).

Meaning of Knowledge Based Expert System As society and industry become more knowledge oriented, they rely on different experts for solving problems and decision making. Here, KBES becomes a productive tool as it provides collective knowledge of one or more experts. It acts as an expert on demand, anytime, anywhere and also helps in saving money by getting cheaper expert knowledge and letting users' function at a higher level with consistency. KBES is a computer-based system which uses and produces knowledge from data, information and knowledge. These systems have the ability to understand the information that is being processed and can take a final decision based on it. This is different from the traditional computer systems that do not have any idea about the data/information which they are processing.

A knowledge based expert system is software or computer based information system which comprises knowledge as data that is processed by computer programs to draw inferences from knowledge base.

Features of Knowledge Based Expert System

The key features of the knowledge-based expert system in MIS are as described below:

- 1) High-Level Expertise: Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System

The high-level expertise provided to help in problem solving is one of the most valuable features of an expert system. This expertise is analogous to the best thinking by top experts in the specific field providing solutions which are creative, precise and efficient.

- 2) Predictive Modeling Power: Predictive modeling power is another useful feature of an expert system. It acts as a model of problem solving in the given domain and throws up answers for given problem situations and also reflects the change in them (given new situations explaining the reason for change). This helps the user to assess the potential effect of new facts or data and know about their relationship to the results.
- 3) Institutional Memory: Institutional memory refers to the body of knowledge which defines the ability of an expert system. When people in important position leave an organisation, their expertise gets reserved as institutional memory. This is significant and critical for vital military and government institutions where personnel transfers are frequent.
- 4) Ability to Provide a Training Facility: An expert system is also capable of providing training to important personnel. As expert systems already have the knowledge and reasoning ability, they can train various employees when required.

Components of Knowledge Based Expert System

A knowledge base, the Inference Engine (IE) (a search program) and user interface are included in Knowledge Based Expert System (KBES):

- 1) Inference Engine (IE): IE is used to understand the knowledge present in the knowledge base.
- 2) Knowledge Base: Knowledge base is the storehouse of different forms of knowledge.
- 3) User Interface: An appropriate user interface must be present which should have the natural language processing facility.

Enterprise Model System

An enterprise model is a representation of the structure, activities, processes, information, resources, people, behavior, goals, and constraints of a business, government, or other enterprises.

What Does Enterprise Modeling Mean? Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System

Enterprise modeling is a term for the modeling of various processes, infrastructures, asset groups, or other elements of a business or organization. Enterprise modeling helps leaders to visualize what is going on within a business and how to make changes. Early forms of enterprise modeling helped analysts to fix hardware and deal with other sorts of troubleshooting. With the growing complexity of business IT infrastructures, enterprise modeling is becoming increasingly useful. Various kinds of enterprise modeling help analysts or others to accomplish different tasks. Business process modeling gives professionals a bird's eye view of a particular business process in order to help with change management. Data modeling helps database administrators and others to more efficiently plan methodology for the use, storage and recall of data, one of a contemporary business's largest assets. Another kind enterprise modeling is function or activity modeling, which can help provide a visual demonstration of work flow. A major part of the utility of

enterprise modeling is related to the value of visuals in planning. By effectively representing structures, processes or hierarchies visually, planners can get much better information about a business and what changes would look like. Enterprise modeling software helps by automating a lot of what would otherwise have to be done manually to work up detailed and sophisticated models of complex business processes such as supply chains.

E-BusinessE-business (electronic business) is the conduct of online business processes on the web, internet, extranet or a combination thereof. These customer-, internal- and management-focused business processes include buying and selling goods and services, servicing customers, processing payments, managing production and supply chains, collaborating with business partners, sharing information, running automated employee services and recruiting employees. E-business is similar to e-commerce but encompasses much more than online purchasing transactions. Functions and services range from the development of intranets and extranets to the provision of eservices over the internet by application service providers. Enterprises conduct e-business to buy parts and supplies from other companies, collaborate on sales promotions and conduct joint research. E-business (electronic business) refers to the carrying out business activities online with the help of basic tools internet, intranet, and extranet. E-business also called online business which includes activities such as buying, selling, business deals, procurements, monetary transactions over the internet, customer relationship management, collaboration with suppliers, and other stakeholders.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System In simple words, electronic business means the transactions of business activities that take place online with the help of the internet. It is the modern way of doing business, handling customers, fulfilling their expectations, and uplifting companies' profits. Electronic business has changed the way the traditional businesses used to be where physical presence is vital for buyers' and sellers' interactions and for the delivery of goods and services. But, in e-business physical presence is not required. Buying and selling, order processing, and buyers and sellers interactions are done electronically using telecommunication networks. Characteristics of E-BusinessE-businesses have some unique characteristics which are as follows –

- Easy setup – It is easy to set up e-businesses. One just needs a website and digital banking-enabled payment gateway to engage in it. No geographic barriers – As e-businesses are operated online via the internet, there is a barrier to geographies in it. Customers from anywhere can buy anything from the business any time they wish. Cost efficient – E-businesses are a cost-efficient mode of business because they save enough money that would have been required if a physical location had to be opted for the business. Flexible timing of business – E-businesses can be done any time and from anywhere. There is no limit on timing as it can be operated 24 hours a day 365 days a year. Cheap marketing – E-businesses don't require elaborate marketing. The costs required are only spent on digital marketing which is cheaper than traditional modes of marketing and advertising. No interaction between buyer and seller – In an e-business, no interaction is needed between the seller and the buyer. Transactions and transfers occur online. The payment is done electronically and the order is shipped to the consumer's address by the seller. Delivery takes extra time – As the seller may be situated far from the buyer, an extra amount of time may be required to get the items delivered in e-businesses. The threat of transaction – The transaction threat is more prominent in the case of e-businesses as hackers may access banking data and steal money from the accounts digitally. Customers may be situated anywhere, they may buy anything and the process can occur at any time the consumer wishes.

Advantages of E-BusinessesSome of the most notable advantages are the following – Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System

- Less costly – E-businesses need only a website and software which cost less than the establishment of a traditional business. So, e-businesses are pocket friendly.
- Easy to organize – Online businesses can be set up at home with minimum requirements of internet, website, and a few software.
- Lack of geographic barriers – There is no barrier of geography to e-businesses. They can sell things worldwide. Customers can be located anywhere in the world. The only requirement is that the customers should have internet and digital payment modes available to them.
- Government subsidies – Governments promote and help e-businesses because they promote digitization. Digitization makes payments transparent and this helps governments keep track of payments easily.
- Flexible timing – There is no fixed working hour for e-businesses. They can be operated round the clock and consumers may reach the site anytime from anywhere. There is no need for anyone to stay available from the seller's end. The process gets done automatically without any errors.

Disadvantages of E-Businesses Following are the most notable disadvantages of e-businesses –

- Lack of interpersonal communication – unlike traditional businesses, e-businesses do not have oneto-one communication. This may lead to the inability to properly judge the quality of products. Moreover, as there is no personal connection, it is hard

for the business to develop trust with the consumer. Delivery time– In traditional business, we get the product delivered immediately after payment. However, in the case of e-business, it takes extra time for the product to reach the consumer. More risks – as hackers can easily get consumers' banking details online, e-businesses are considered riskier than traditional methods. ERP systemsERP is an integrated, real-time, cross-functional enterprise application, an enterprise-wide transaction framework that supports all the internal business processes of a company. It supports all core business processes such as sales order processing, inventory management and control, production and distribution planning, and finance. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System Why of ERP? ERP is very helpful in the following areas – • Business integration and automated data update • Linkage between all core business processes and easy flow of integration • Flexibility in business operations and more agility to the company • Better analysis and planning capabilities • Critical decision-making • Competitive advantage • Use of latest technologies Features of ERP The following diagram illustrates the features of ERP – Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System Scope of ERP • Finance – Financial accounting, Managerial accounting, treasury management, asset management, budget control, costing, and enterprise control. • Logistics – Production planning, material management, plant maintenance, project management, events management, etc. • Human resource – Personnel management, training and development, etc. • Supply Chain – Inventory control, purchase and order control, supplier scheduling, planning, etc. • Work flow – Integrate the entire organization with the flexible assignment of tasks and responsibility to locations, position, jobs, etc. Advantages of ERP • Reduction of lead time • Reduction of cycle time • Better customer satisfaction • Increased flexibility, quality, and efficiency • Improved information accuracy and decision making capability • Onetime shipment • Improved resource utilization • Improve supplier performance • Reduced quality costs Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System • Quick decision-making • Forecasting and optimization • Better transparency Disadvantage of ERP • Expense and time in implementation • Difficulty in integration with other system • Risk of implementation failure • Difficulty in implementation change • Risk in using one vendor E-CommerceThe terms e-commerce and e-business are often used interchangeably but they differ. E-commerce means buying and selling goods and services online with the help of the internet and making monetary transactions at the same time. E-commerce remains at this. But, e-business goes beyond just buying and selling over the internet and making payments as mentioned above it also includes managing customers, CRM, ERP, managing production control, processing payments, e-recruiting, e-collaboration, and many more. In fact, electronic commerce is the part of the electronic business – it can be best convinced as electronic commerce is the subset of electronic business whereas e-business is the superset of e-commerce. Ecommerce is a method of buying and selling goods and services online. The definition of ecommerce business can also include tactics like affiliate marketing. You can use ecommerce channels such as your own website, an established selling website like Amazon, or social media to drive online sales. Types of E-commerceDepending on the goods, services, and organization of an ecommerce company, the business can opt to operate several different ways. Here are several of the popular business models. Business-to-Consumer (B2C) B2C e-commerce companies sell directly to the product end-user. Instead of distributing goods to an intermediary, a B2C company performs transactions with the consumer that will ultimately use the good. This type of business model may be used to sell products (like your local sporting goods store's website) or services (such as a lawn care mobile app to reserve landscaping services). This is the most Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System common business model and is likely the concept most people think about when they hear the term ecommerce. Business-to-Business (B2B) Similar to B2C, an e-commerce business can directly sell goods to a user. However, instead of being a consumer, that user may be another company. B2B transactions often entail larger quantities, greater specifications, and longer lead times. The company placing the order may also have a need to set recurring goods if the purchase is for recurring manufacturing processes. Business-to-Government (B2G) Some entities specialize as government contractors providing goods or services to agencies or administrations. Similar to a B2B relationship, the business produces items of value and remits those items to an entity. B2G e-commerce companies must often meet government requests for proposal requirements, solicit bids for projects, and meet very specific product or service criteria. In addition, there may be joint government endeavors to solicit a single contract through a government-

wide acquisition contract. Consumer-to-Consumer (C2C) Established companies are the only entities that can sell things. E-commerce platforms such as digital marketplaces connect consumers with other consumers who can list their own products and execute their own sales. These C2C platforms may be auction-style listings (i.e. eBay auctions) or may warrant further discussion regarding the item or service being provided (i.e. Craigslist postings). Enabled by technology, C2C e-commerce platforms empower consumers to both buy and sell without the need for companies. Consumer-to-Business (C2B) Modern platforms have allowed consumers to more easily engage with companies and offer their services, especially related to short-term contracts, gigs, or freelance opportunities. For example, consider listings on up work. A consumer may solicit bids or interact with companies that need particular jobs done. In this way, the e-commerce platform connects businesses with freelancers to enable consumers greater power to achieve pricing, scheduling, and employment demands.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System Consumer-to-Government (C2G) Less of a traditional e-commerce relationship, consumers can interact with administrations, agencies, or governments through C2G partnerships. These partnerships are often not in the exchange of service but rather, the transaction of obligation. For example, uploading your federal tax return to the Internal Revenue Service (IRS) digital website is an e-commerce transaction regarding an exchange of information. Alternatively, you may pay your tuition to your university online or remit property tax assessments to your county assessor.

E-communication

Electronic communication is any form of communication that's broadcast, transmitted, stored or viewed using electronic media, such as computers, phones, email and video. An e-communication could be an email, SMS text message or another type of electronic or digital communication, such as a communication through internal accounts used to communicate with electors about other local authority services. An example of electronic communication

Email, instant messaging, websites, blogs, text messaging, voicemail and video messaging are a few examples of electronic communication. Electronic communication has changed the way businesses communicate with each other. Electronic communication can be very beneficial if used effectively.

Types of Electronic Communication

Electronic communication can be classified into different types like messaging, voice call, e-mail, social media, etc. We know that e-communication has changed due to the way public interact and communicate with each other for different purposes like personal or business. By using this, it is very simple to communicate with the world. E-Mail

E-Mail or electronic mail is the most used type of electronic communication. By using this communication, one can send a message to another person through a mail immediately. For that, we need to create an account to send an e-mail, media files, photos, documents, etc. This type of communication has replaced many conventional types of communication due to many benefits.

e-mail Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System So this type of communication is more suitable for different methods of communication. The benefits of this communication are ease of usage, completely free, etc. Additionally, this type of electronic communication doesn't affect the surroundings.

Messaging

This type of communication allows people to interact with others who are far away from us. This is possible only due to technology as well as usage of the internet. There are different types of messengers available like Skype, Windows Live, Gmail, etc. These messengers help in chatting or sending messages to our beloved ones or friends. There are many benefits by using this kind of communication like the message which we sent & the response are immediate. But in some cases, some files include nil although bug can stop the functioning of your computer by giving you lots of trouble.

Blogging

At present, blogging is the most preferable communication method. This is a type of online journaling, which can be updated daily, or many times a day. It covers all the information or a particular topic. By using such blogs, one can share, follow, or even post comments. This kind of communication is extremely suitable. This is the reason why people utilize blogs very often. Additionally, by using the internet, people can access, read & follow it worldwide.

blogging

Video Chat

This type of communication can be done by adding web cameras for video calling application. By using this application, one can communicate with others and also they can observe with whom they are speaking. The webcam can be connected to the computer externally and also we need to use applications like Skype, Hangouts, etc. There are many benefits to using video chatting. We can contact anybody immediately. We can communicate with more than one person at a time by using the feature like business conference feature. Also, we can share PPTs, data sheets online.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System

Social Networking

Social media is one kind of communication between people, which is used with their general advantage otherwise for relationships. In this, mostly Facebook, as well as LinkedIn, give places for

people to work together, sometimes in real-time. There is a Micro-blogging service namely Twitter, which allows the short message of more than 140 characters to be transmitted to a huge audience. Social\_media Not like text messages, it sends to simply tiny groups. The posts like Microblog are intended to be seen by all the followers and users can repost texts that they desire to share with their followers. Therefore, a microblog post can reach rapidly and a viral post is s message which reports widely. Telex This is a significant device for current electronic communication. This system uses a tele-printer to communicate from one position to another using a machine. It includes mainly two parts like keyboard transmitter as well as a receiver. telex Whenever a text is to be sent, then the user presses a push-button, and stays for the call tone, calls the number preferred & enters the massage on a tiny paper strip at the end of receiver end because it is entered within the creating office. This method is the quickest & most exact methods for exchanging written posts. Fax The Fax machine is a kind of communications and use of this is increasing gradually to transmit materials which are visual like illustrations, diagrams, picture, etc. Here, this machine can be connected using a telephonic. The transmitted document can be fed throughout the machine, after that it is scanned electronically & signals are broadcasted to the end of receiver wherever an equal document copy is replicated on a plain paper sheet using the receiving machine. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System fax This machine has made it achievable to send important documents copies which include testimonials, certificates, degrees, contracts, agreements from one location to another in a telephone call speed. Because of this reason, it is a commonly used technique for communication. Multimedia The multimedia is one kind of communication system and it is an excellent innovation to improve the communication system. This is a blend of several media which bring mutually to transmit messages. The multimedia mainly includes a photo, graphics, voice, music, animation, and message. Whenever all these media are located jointly otherwise computer screen then becomes multimedia. This can be used efficiently for marketing and advertising campaigns. This type of communication is extremely powerful. multimedia Thus, this is all about electronic communication principles which include immediate messaging, websites, social networking voicemail, e-mail, and text messaging. This communication has changed completely the way people communicate with each other. This can be used for personal, business, etc. By using this, it is extremely simple to communicate with the entire world. Here is a question for you, what are the strengths & weaknesses of the electronic system? Business Process ReengineeringBusiness Process Reengineering is the radical redesign of business processes to achieve dramatic improvements in productivity, cycle times, quality, and employee and customer satisfaction. Business Process Re-engineering (BPR) is the radical redesign of business processes to achieve dramatic improvements in critical aspects like quality, output, cost, service, and speed. Business process reengineering (BPR) aims at cutting down enterprise costs and process redundancies on a very huge scale. Five steps of Business Process Re-engineering (BPR) To keep business process re-engineering fair, transparent, and efficient, stakeholders need to get a better understanding of the key steps involved in it. Although the process can differ from one organization to another, these steps listed below succinctly summarize the process: Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System Below are the 5 Business Process Re-engineering Steps: 1. Map the current state of your business processes Gather data from all resources—both software tools and stakeholders. Understand how the process is performing currently. 2. Analyze them and find any process gaps or disconnects Identify all the errors and delays that hold up a free flow of the process. Make sure if all details are available in the respective steps for the stakeholders to make quick decisions. 3. Look for improvement opportunities and validate them Check if all the steps are absolutely necessary. If a step is there to solely inform the person, remove the step, and add an automated email trigger. 4. Design a cutting-edge future-state process map Create a new process that solves all the problems you have identified. Don't be afraid to design a totally new process that is sure to work well. Designate KPIs for every step of the process. 5. Implement future state changes and be mindful of dependencies Inform every stakeholder of the new process. Only proceed after everyone is on board and educated about how the new process works. Constantly monitor the KPIs. A real-life example of BPR Many companies like Ford Motors, GTE, and Bell Atlantic tried out BPR during the 1990s to reshuffle their operations. The reengineering process they adopted made a substantial difference to them, dramatically cutting down their expenses and making them more effective against increasing competition. \*\*\*\*\* Case Study video link - Case Study on Information Systems - <https://youtu.be/0lJgIKxAJQM>

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Name: Machine Learning Subject Code: AD 502 Subject Notes Syllabus: Ensemble Learning and Random Forest: Introduction to Ensemble Learning, Basic Ensemble Techniques (Max Voting, Averaging, Weighted Average), Voting Classifiers, Bagging and Pasting, Out-of-Bag Evaluation, Random Patches and Random Subspaces, Random Forests (Extra-Trees, Feature Importance), Boosting (AdaBoost, Gradient Boosting), Stacking.

#### Course

Outcome (CO4): Apply structured thinking to unstructured problems. Unit-IV Ensemble Learning Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model. Basic idea is to learn a set of classifiers (experts) and to allow them to vote.

Advantage : Improvement in predictive accuracy. Disadvantage : It is difficult to understand an ensemble of classifiers.

Diagram 4.1 Ensemble Learning Why do ensembles work? Ensembles overcome three problems – Statistical Problem – The Statistical Problem arises when the hypothesis space is too large for the amount of available data. Hence, there are many hypotheses with the same accuracy on the data and the learning algorithm chooses only one of them! There is a risk that the accuracy of the chosen hypothesis is low on unseen data! Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Computational Problem – The Computational Problem arises when the learning algorithm cannot guarantees finding the best hypothesis. Representational Problem – The Representational Problem arises when the hypothesis space does not contain any good approximation of the target classes. Main Challenge for Developing Ensemble Models? The main challenge is not to obtain highly accurate base models, but rather to obtain base models which make different kinds of errors. For example, if ensembles are used for classification, high accuracies can be accomplished if different base models misclassify different training examples, even if the base classifier accuracy is low.

Methods for Independently Constructing Ensembles – • Majority Vote • Bagging and Random Forest • Randomness Injection • Feature-Selection Ensembles • Error-Correcting Output Coding Methods for Coordinated Construction of Ensembles – • Boosting • Stacking Ensemble Techniques Max-voting Max-voting, which is generally used for classification problems, is one of the simplest ways of combining predictions from multiple machine learning algorithms. In max-voting, each base model makes a prediction and votes for each sample. Only the sample class with the highest votes is included in the final predictive class. Averaging Model averaging is an approach to ensemble learning where each ensemble member contributes an equal amount to the final prediction. In the case of regression, the ensemble prediction is calculated as the average of the member predictions Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science

Weighted Average In a weighted average, each data point value is multiplied by the assigned weight, which is then summed and divided by the number of data points. A weighted average can improve the data's accuracy. Stock investors use a weighted average to track the cost basis of shares bought at varying times. Voting Classifiers A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output. It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting. The idea is instead of creating separate dedicated models and finding the accuracy for each them, we create a single model which trains by these models and predicts output based on their combined majority of voting for each output class. Voting Classifier supports two types of votings. Hard Voting: In hard voting, the predicted output class is a class with the highest majority of votes i.e the class which had the highest probability of being predicted by each of the classifiers. Suppose three classifiers predicted the output class(A, A, B), so here the majority predicted A as output. Hence A will be the final prediction. Soft Voting: In soft voting, the output class is the prediction based on the average of probability given to that class. Suppose given some input to three models, the prediction probability for class A = (0.30, 0.47, 0.53) and B = (0.20, 0.32, 0.40). So the average for class A is 0.4333 and B is 0.3067, the winner is clearly class A because it had the highest probability averaged by each classifier.

Bagging and Pasting Bagging The Bagging is an assembling approach that tries to resolve overfitting for class or the regression problems. Bagging pursues to improve the accuracy and overall performance of gadget mastering algorithms. It does this by taking random subsets of an original dataset, with substitute, and fits either a classifier (for classification) or regressor (for regression) to each subset. Bagging is also known as Bootstrap aggregating. It is an ensemble learning approach that enhances the overall performance and accuracy of the gadget for learning algorithms. It is miles used to address bias-variance alternate-off increases and decreases the variance of a prediction version. The Bagging avoids overfitting of data and is used for each regression and classification of the class, in particular for the

decision tree algorithms. Implementation Steps of Bagging Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Step 1: Multiple subsets are made from the original information set with identical tuples, deciding on observations with replacement. Step 2: A base model is created on all subsets. Step 3: Every version is found in parallel with each training set and unbiased. Step 4: The very last predictions are determined by combining the forecasts from all models. Diagram 4.2 Bagging Application of the Bagging: 1. IT 2. Environment 3. Finance 4. Healthcare Advantages of Bagging : 1. Easier for implementation Python libraries, including scikit-examine (sklearn), make it easy to mix the predictions of base beginners or estimators to enhance model performance. Their documentation outlines the available modules you can leverage for your model optimization. 2. Variance reduction The Bagging can reduce the variance inside a getting to know set of rules which is especially helpful with excessive-dimensional facts, where missing values can result in better conflict, making it more liable to overfitting and stopping correct generalization to new datasets. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Pasting in ML Pasting is an ensemble technique similar to bagging except for the fact that in pasting sampling is done without replacement i.e. an observation can be present in only one subset. Since pasting limits diversity of models its performance with is suboptimal when compared to bagging, particularly in case of small datasets. However, pasting is preferred over bagging in case of so large datasets, owing to computational efficiency. Out-of-Bag Evaluation Out-of-bag (OOB) error, also called out-of-bag estimate, is a method of measuring the prediction error of random forests, boosted decision trees, and other machine learning models utilizing bootstrap aggregating (bagging). Bagging uses subsampling with replacement to create training samples for the model to learn from. OOB error is the mean prediction error on each training sample  $x_i$ , using only the trees that did not have  $x_i$  in their bootstrap sample Bootstrap aggregating allows one to define an out-of-bag estimate of the prediction performance improvement by evaluating predictions on those observations that were not used in the building of the next base learner. When bootstrap aggregating is performed, two independent sets are created. One set, the bootstrap sample, is the data chosen to be "in-the-bag" by sampling with replacement. The out-of-bag set is all data not chosen in the sampling process. When this process is repeated, such as when building a random forest, many bootstrap samples and OOB sets are created. The OOB sets can be aggregated into one dataset, but each sample is only considered out-of-bag for the trees that do not include it in their bootstrap sample. The picture below shows that for each bag sampled, the data is separated into two groups. Diagram 4.3 Out-of-Bag Evaluation Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science This example shows how bagging could be used in the context of diagnosing disease. A set of patients are the original dataset, but each model is trained only by the patients in its bag. The patients in each out-of-bag set can be used to test their respective models. The test would consider whether the model can accurately determine if the patient has the disease. With random Subspaces, estimators differentiate because of random subsets of the features. Again, such a solution is achievable by tuning the parameters of BaggingClassifier and BaggingRegressor, by setting max\_features to a number less than 1.0, representing the percentage of features to be chosen randomly for each model of the ensemble. Random Patches and Random Subspaces With random Subspaces, estimators differentiate because of random subsets of the features. Again, such a solution is achievable by tuning the parameters of BaggingClassifier and BaggingRegressor, by setting max\_features to a number less than 1.0, representing the percentage of features to be chosen randomly for each model of the ensemble. Instead, in Random Patches, estimators are built on subsets of both samples and features. Random Forests Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting. The below diagram explains the working of the Random Forest algorithm: Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Diagram -4.4 Random Forest Why use Random Forest? Below are some points that explain why we should use the Random Forest algorithm: • It takes less training time as compared to other algorithms. • It predicts output with high accuracy, even for the large dataset it runs efficiently. • It can also maintain accuracy when a large proportion of data is missing. Boosting Boosting is an ensemble learning method that combines a set of weak

learners into strong learners to minimize training errors. In boosting, a random sample of data is selected, fitted with a model, and then trained sequentially. That is, each model tries to compensate for the weaknesses of its predecessor. Each classifier's weak rules are combined with each iteration to form one strict prediction rule. Boosting is an efficient algorithm that converts a weak learner into a strong learner. They use the concept of the weak learner and strong learner conversation through the weighted average values and higher votes values for prediction. These algorithms use decision stamp and margin maximizing classification for processing. Diagram -4.5 Boosting Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science There are three types of Algorithms available such as AdaBoost or Adaptive boosting Algorithm, Gradient, and XG Boosting algorithm. These are the machine learning algorithms that follow the process of training for predicting and fine-tuning the result. Example Let's understand this concept with the help of the following example. Let's take the example of the email. How will you recognize your email, whether it is spam or not? You can recognize it by the following conditions:

- If an email contains lots of sources, that means it is spam.
- If an email contains only one file image, then it is spam.
- If an email contains the message "You Own a lottery of \$xxxxx," it is spam.
- If an email contains some known source, then it is not spam.
- If it contains the official domain like educba.com, etc., it is not spam.

The rules mentioned above are not that powerful to recognize spam or not; hence these rules are called weak learners. To convert weak learners to the strong learner, combine the prediction of the weak learner using the following methods.

1. Using average or weighted average.
2. Consider prediction has a higher vote.

Consider the 5 rules mentioned above; there are 3 votes for spam and 2 votes for not spam. Since there is high vote spam, we consider it spam. Types of Boosting

Boosting methods are focused on iteratively combining weak learners to build a strong learner that can predict more accurate outcomes. As a reminder, a weak learner classifies data slightly better than random guessing. This approach can provide robust prediction problem results, outperform neural networks, and support vector machines for tasks. Boosting algorithms can differ in how they create and aggregate weak learners during the sequential process. Three popular types of boosting methods include:

1. Adaptive boosting or AdaBoost: This method operates iteratively, identifying misclassified data points and adjusting their weights to minimize the training error. The model continues to optimize sequentially until it yields the strongest predictor. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science AdaBoost is implemented by combining several weak learners into a single strong learner. The weak learners in AdaBoost take into account a single input feature and draw out a single split decision tree called the decision stump. Each observation is weighted equally while drawing out the first decision stump. The results from the first decision stump are analyzed, and if any observations are wrongfully classified, they are assigned higher weights. A new decision stump is drawn by considering the higher-weight observations as more significant. Again if any observations are misclassified, they're given a higher weight, and this process continues until all the observations fall into the right class. AdaBoost can be used for both classification and regression-based problems. However, it is more commonly used for classification purposes.
2. Gradient Boosting: Gradient Boosting is also based on sequential ensemble learning. Here the base learners are generated sequentially so that the present base learner is always more effective than the previous one, i.e., and the overall model improves sequentially with each iteration. The difference in this boosting type is that the weights for misclassified outcomes are not incremented. Instead, the Gradient Boosting method tries to optimize the loss function of the previous learner by adding a new model that adds weak learners to reduce the loss function. The main idea here is to overcome the errors in the previous learner's predictions. This boosting has three main components:

- Loss function: The use of the loss function depends on the type of problem. The advantage of gradient boosting is that there is no need for a new boosting algorithm for each loss function.
- Weak learner: In gradient boosting, decision trees are used as a weak learners. A regression tree is used to give true values, which can combine to create correct predictions. Like in the AdaBoost algorithm, small trees with a single split are used, i.e., decision stump. Larger trees are used for large levels, e.g., 4-8.
- Additive Model: Trees are added one at a time in this model. Existing trees remain the same. During the addition of trees, gradient descent is used to minimize the loss function. Like AdaBoost, Gradient Boosting can also be used for classification and regression problems.

3. Extreme gradient boosting or XGBoost: XGBoost is an advanced gradient boosting method. XGBoost, developed by Tianqi Chen, falls under the Distributed Machine Learning Community (DMLC) category. The main aim of this algorithm is to increase the speed and efficiency of computation. The Gradient Descent Boosting algorithm computes the output slower since they sequentially analyze the data set. Therefore XGBoost is used to boost or extremely boost the model's performance. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science XGBoost is designed to focus on computational speed and model efficiency. The main features

provided by XGBoost are:

- Diagram -4.6 Extreme gradient boosting or XGBoost
- Parallel Processing: XG Boost provides Parallel Processing for tree construction which uses CPU cores while training.
- Cross-Validation: XG Boost enables users to run cross-validation of the boosting process at each iteration, making it easy to get the exact optimum number of boosting iterations in one run.
- Cache Optimization: It provides Cache Optimization of the algorithms for higher execution speed.
- Distributed Computing: For training large models, XG Boost allows Distributed Computing. Stacking Stacking is one of the most popular ensemble machine learning techniques used to predict multiple nodes to build a new model and improve model performance. Stacking enables us to train multiple models to solve similar problems, and based on their combined output, it builds a new model with improved performance. This ensemble technique works by applying input of combined multiple weak learners' predictions and Meta learners so that a better output prediction model can be achieved. In stacking, an algorithm takes the outputs of sub-models as input and attempts to learn how to best combine the input predictions to make a better output prediction. Stacking is also known as a stacked generalization and is an extended form of the Model Averaging Ensemble technique in which all sub-models equally participate as per their performance weights and build a new model with better predictions. This new model is stacked up on top of the others; this is the reason why it is named stacking. Architecture of Stacking The architecture of the stacking model is designed in such a way that it consists of two or more base/learner's models and a meta-model that combines the predictions of the base models. These base models are called level 0 models, and the meta-model is known as the level 1 model. So, the Stacking ensemble method includes Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science original (training) data, primary level models, primary level prediction, secondary level model, and final prediction. The basic architecture of stacking can be represented as shown below the image. Diagram -4.7 Stacking Architecture
- Original data: This data is divided into n-folds and is also considered test data or training data.
- Base models: These models are also referred to as level-0 models. These models use training data and provide compiled predictions (level-0) as an output.
- Level-0 Predictions: Each base model is triggered on some training data and provides different predictions, which are known as level-0 predictions.
- Meta Model: The architecture of the stacking model consists of one meta-model, which helps to best combine the predictions of the base models. The meta-model is also known as the level-1 model.
- Level-1 Prediction: The meta-model learns how to best combine the predictions of the base models and is trained on different predictions made by individual base models, i.e., data not used to train the base models are fed to the meta-model, predictions are made, and these predictions, along with the expected outputs, provide the input and output pairs of the training dataset used to fit the metamodel.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System UNIT-V Evaluation and Maintenance of MIS, Protecting the Information Systems-Security challenges in E-enterprises; Security threats, vulnerability, and safeguards, Controlling security threat and vulnerability, Technologies for Information System Control, Disaster Recovery Plans. Emerging trends and technologies with regard to Management Information Systems. \*\*\*\*\* Evaluation and Maintenance of MISImplementation, Evaluation and Maintenance of Information System Implementation The design of a management information system may seem to management to be an expensive project, the cost of getting the MIS on line satisfactorily may often be comparable to that of its design, and the implementation has been accomplished when the outputs of the MIS are continuously utilized by decision makers. Once the design has been completed, there are four basic methods for implementing the MIS. These areas

1. Install the system in a new operation or organization.
2. Cut off the old system and install the new This produces a time gap during which no system is in operation. Practically, installation requires one or two days for small companies or small systems.
3. Cut over by segments This method is also referred as " phasing in" the new system. Small parts or subsystems are substituted for the old. In the case of upgrading old systems, this may be a very desirable method.
4. Operate in parallel and cut over. The new system is installed and operated in parallel with the current system until it has been checked out, then only the current system is cut out. This method is expensive because of personal and related costs. Its big advantages are that the system is fairly well debugged when it becomes the essential information system.

Plan the implementation The three main phases in implementation take place in series. These are

1. The initial installation
2. The test of the system as a whole

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System 3. The evaluation,

maintenance and control of the system. Many implementation activities should be undertaken in parallel to reduce implementation time. Training of personnel and preparation of software may be in parallel with each other and with other implementation activities. The first step in the implementation procedure is to plan the implementation. Some analyst includes the planning of the implementation with the design of the system, the planning and the action to implement the plan should be bound closely together. Planning is the first step of management, not the last. The MIS design and the urgent need for the system at the time the design is completed will weigh heavily on the plan for implementation.

**Implementation Tasks** The major implementation tasks consists of

1. Planning the implementation activities
2. Acquiring and laying out facilities and offices
3. Organizing the personnel for implementation
4. Developing procedures for installation and testing
5. Developing the training program for operating personnel
6. Completing the system's software
7. Acquiring required hardware
8. Generating files
9. Designing forms
10. Testing the entire system
11. Completing cutover to the new system
12. Documenting the system
13. Evaluating the MIS
14. Providing system maintenance(debugging and improving)

Establish Relationships among tasks For small projects, the order of performance may simply be described in text form. A Gantt chart or network diagram makes visualization of the plan and schedule much clearer. For large projects, many concurrent and sequential activities are interrelated so that a network diagram must be employed in any good plan. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System Establish a Schedule Schedule is prepared by having the system designers estimate the times between the events in the program network. The critical path (longest time through the network) can be calculated. After specifying the starting date, the end date is established. Cost Schedule to Tasks and Time The cost for completing each task required to complete is established as part of the plan; then the rate of expenditures should be budgeted. Reporting and control of the work in progress may be obtained by weekly meetings. The financial personnel must make certain that report formats allow them to show cost and technical progress relationship as well as cost and time.

2. Acquiring and laying out facilities and offices For the installation of a new system to replace a current one may require a major revision of facilities as well as completely new office, computer room etc. The MIS project manager must prepare rough layouts and estimates of particular floor areas that feel to be needed. The manager then prepares cost estimates. Space planning must be done by the space to be occupied by people, the space occupied by equipment and the movement of people and equipment in the work progress. A large investment in good working conditions will repay its cost many times.
3. Organizing the personnel for implementation As the implementation tasks have been defined, management usually assigns a project manager to guide the implementation. The purpose of the MIS is to increase the amount and quality of their contributions, the system is their system. Top management must make the middle managers for their involvement in implementation, besides these, systems specialists, computer programmer; top management should make sure that each people who will operate the system should have active parts in the implementation.
4. Developing procedures for installation and testing After organizing the personnel for implementation the next task is to develop or prepare the procedures for implementation. As the project leader has the network plan for proceeding with the implementation, this leader calls the key people in the project to prepare more detailed procedures for system installation. Procedures for evaluating and selecting hardware must be spelled out. Procedures for phasing in parts of the MIS or operating the MIS in parallel must be developed.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System The major part of implementing the MIS is the testing of each segment of total system as it is installed.

5. Developing the training program for operating personnel A program is developed keeping in mind to impress management and support. After developing the program, it is necessary to train operating personnel in their new duties. They must have a thorough understanding of what the new MIS is like and what it is supposed to do. They must learn how it will operate. They are faced with many changes in their work and have to obtain acceptance of changes. As there are various levels of personnel and these people will be working with only a small part of the MIS, the seminars should be designed to provide them with an understanding of the complete system.
6. Completing the system's software As the software is developed internally or under contract, in both cases, the software development must take in mind the nature of the hardware required. As the system designers and programmers provide the flow diagrams and the block diagrams during the detailed design state. Some modification may be required, as the implementation stage progresses.
7. Acquiring required hardware This acquisition is usually the limiting factor in getting am MIS implementation. These tasks should be started during the design stage. The decision is to be needed, whether to buy or

lease the hardware. Capital expenditure analysis is only one of many factors involved in this decision. Others are prestige, usage etc. 8. Generating files In the implementation stage, the actual data must be obtained and recorded for the initial testing and operation of the system. This requires format of the data, storage form and format and remarks to indicate when the data have been stored. The collection of data used in routine operations is often called the master file.

Responsibility for file maintenance for each file item should also be assigned. The development of files or databases belongs to information system designers and storage and retrieval experts. The translation of specifications for files into computer programs is a function of computer specialists. 9. Designing forms For controlling the marketing, a salesperson has to fill out the forms summarizing the day's activities. The form ensures the right information to be supplied for computer storage. Forms are required not just for input and output but also for transmitting data at intermediate stages.

10. Testing the entire system As the total system is installed, tests should be performed with the test specifications and procedure. A Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System test during installation stage consists of component tests, subsystem tests and total system acceptance tests. Components may be equipment (that can be new or old), new software programs, new data collection methods, work procedures, reporting formats. Difficulties that occur during component tests may lead to design changes. As more components are installed, subsystems may be tested. There is a difference between the testing of component and the testing of a system. System tests require verification of multiple inputs, complex logic systems, and timing aspects of many parts. 11. Completing cutover to the new system Cutover is a point at which the new component replaces the old component to the new system replaces the old system. This involves old forms, old files and old equipment being retried. The debugging proves associated with the cutover to the new system may extend for several months. 12. Documenting the system Documentation of the MIS means preparation of written descriptions of the scope, purpose, information flow components, and operating procedures of the system.

Documentation is a necessity for troubleshooting, for replacement of subsystems, for interfacing with other systems, for training new operating personnel and also for evaluating and upgrading the system. 13. Evaluating the system After the MIS has been operating smoothly for a short period of time, an evaluation of each step in the design and of the final system performance should be made. Evaluation should not be delayed beyond the time when the system's analysts have completed most of the debugging. The longer the delay, the more difficult it will be for designer to remember important details. The evaluation should be made by the customer as well as by the designers. 14. Providing system maintenance Control and maintenance of the system are the responsibilities of the line managers. Control of the systems means the operation of the system as it was designed to operate. Sometimes, well-intentioned people or operators may make unauthorized changes to improve the system, changes that are not approved or documented. Maintenance is closely related to control. Maintenance is that ongoing activity that keeps the MIS at the highest levels of effectiveness and efficiency within cost constraints. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System Maintenance is directed towards reducing errors due to design, reducing errors due to environmental changes and improving the system's scope and services. Protecting the Information SystemsThe protection of information systems against unauthorized access to or modification of

information, whether in storage, processing or transit, and against the denial of service to authorized users, including those measures necessary to detect, document, and counter such threats. An information system (IS) is a collection of hardware, software, data, and people that work together to collect, process, store, and disseminate information. An IS can be used for a variety of purposes, such as supporting business operations, decision making, and communication.

Information security refers to the protection of information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction. It aims to protect the confidentiality, integrity, and availability of information and information systems. • Information systems are vulnerable to a variety of security threats, such as hackers, viruses, and natural disasters. As such, it is important for organizations to implement appropriate security measures to protect their information systems. • There are several different security measures that organizations can implement to protect their information systems, such as:

- Firewalls: Firewalls are used to restrict access to an organization's network and to protect against unauthorized access.
- Intrusion detection systems: These systems are used to detect and alert organizations to potential security breaches.
- Encryption: Encryption is used to protect sensitive information by converting it into unreadable code.
- Access controls: Access controls are used to restrict access to information and information systems to authorized individuals only.
- Security policies: Organizations can implement

security policies to ensure that their employees understand their security responsibilities and adhere to them. • Security Auditing: Regularly monitoring the system for possible malicious activities and vulnerabilities. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System • By implementing these security measures, organizations can protect their information systems from unauthorized access and use, and ensure that their sensitive information is kept confidential and secure. The Information System is an integrated set of the component for collecting, storing, processing and communicating information. Business firm and other organization on the information system to manage their operation in the marketplace supply service and augment personals lives. Security challenges in E-enterprises; Security Challenge: The number of smart phone devices capable of offering internet technology and experience rivaling desktop computer standards is growing at a fast pace. Security and privacy concern for mobile devices rival or go beyond similar concern for a laptop computer as mobile device are even more mobile by nature and are less likely to be managed by an organization. Ensure Security: In order to ensure security, it is necessary to provide at least the following services, which are given below.

- 1. Authorization: It is act of determining whether an (authenticate) entity has the right to execute action.
- 2. Audit: An auditing service providing a history of action that can be used to determine what (if anything) went wrong and what caused it to go wrong.
- 3. Physical authentication: Some form of authentication such as an object (a key or a smart card ) or a personal characteristic like a fingerprint, retinal pattern, hand geometry.
- 4. Data Confidentiality: It protects against disclosure of any data while in transit and is provided by encryption of data.

ADVANTAGES OR DISADVANTAGES: Advantages of implementing information system and security include:

- Protection of sensitive information: By implementing security measures, organizations can protect their sensitive information from unauthorized access, use, disclosure, disruption, modification, or destruction.
- Compliance: Implementing information security can help organizations meet compliance requirements, such as HIPAA, PCI-DSS, and SOX.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System

- Risk management: By implementing security measures, organizations can better manage the risks associated with their information systems.
- Business continuity: By protecting information systems from natural disasters, power outages and other disruptions, organizations can ensure that their business operations can continue uninterrupted.
- Cost savings: Implementing security measures can help organizations avoid costly data breaches and other security incidents.

Disadvantages of implementing information system and security include:

- Cost: Implementing security measures can be costly, as it may require additional resources, such as security experts, to manage the process.
- Time-consuming: Implementing security measures can be time-consuming, especially for organizations that have not previously used this framework.
- Complexity: Implementing security measures can be complex, especially for organizations that have a lot of data and systems to protect.
- Inflexibility: Security measures can be inflexible, making it difficult for organizations to respond quickly to changing security needs.
- Limited Adaptability: Security measures are predefined, which is not adaptable to new technologies, it may require updating or revising to accommodate new technology.

Security threats, vulnerability, and safeguards, Vulnerability vs threat vs risk These terms are frequently used together, but they do explain three separate components of cyber security. In short, we can see them as a spectrum:

- First, vulnerability exposes your organization to threats.
- A threat is a malicious or negative event that takes advantage of a vulnerability.
- Finally, the risk is the potential for loss and damage when the threat does occur.

What is a vulnerability? A vulnerability is a weakness, flaw or other shortcoming in a system (infrastructure, database or software), but it can also exist in a process, a set of controls, or simply just the way that something has been implemented or deployed. There are different types of vulnerabilities:

- Technical vulnerabilities, like bugs in code or an error in some hardware or software.
- Human vulnerabilities, such as employees falling for phishing, smashing or other common attacks.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System

Some vulnerability is routine: you release something and quickly follow up with a patch for it. The issue with the weakness is when it is unknown or undiscovered to your team. If it's left as-is, this weakness could be vulnerable to some attack or threat. For example, vulnerability is leaving your door unlocked overnight. It alone isn't a problem, but if a certain person comes along and enters that door, some bad, bad things might happen. Here, the more vulnerabilities you have, the greater potential for threats and the higher your risk.

What is a threat? In cyber security, the most common definition of a threat is this: Anything that could exploit a vulnerability, which could affect the confidentiality, integrity or availability of your systems, data, people and more.

(Confidentiality, integrity and availability, sometimes known as the CIA triad, is another fundamental concept of cyber security.) A more advanced definition of threat is when an adversary or attacker has the opportunity, capability and intent to bring a negative impact upon your operations, assets, workforce and/or customers. Examples of this can include malware, ransomware, phishing attacks and more — and the types of threats out there will continue to evolve. For example, your organization may have no vulnerabilities to exploit due to a solid patch management program or strong network segmentation policies that prevent access to critical systems. Chances are likely, however, that you do have vulnerabilities, so let's consider the risk factor. What is a risk? Risk is the probability of a negative (harmful) event occurring as well as the potential of scale of that harm. Your organizational risk fluctuates over time, sometimes even on a daily basis, due to both internal and external factors. This is where cyber security teams can begin to measure that risk: 1. Estimate how often an adversary or attacker is likely to attempt to exploit a vulnerability to cause the desired harm. 2. Gauge how well your existing systems, controls and processes can standup to those attempts. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System 3. Determine the value of the impact or harm the adversary may cause if the adversary is indeed successful. One way of describing risk was consequence X likelihood, but as security teams have advanced their processes and intelligence, we see that you have to also account for the safeguards you've already put in place. Risk = threat x vulnerability This is another way of looking at risk, albeit a bit simplified: Vulnerability x Threat = Risk We can sum up this calculation with the concepts from above: that a single vulnerability multiplied by the potential threat (frequency, existing safeguards, and potential value loss) can give you an estimate of the risk involved. In order for organizations to begin risk mitigation and risk management, you first need to understand your vulnerabilities and the threats to those vulnerabilities. This is no small task. 11 Practical ways to keep your IT systems safe and secure Here are some practical steps you and your staff can take to improve your data security. 1. Back up your data You should back up your data regularly. If you're using an external storage device, keep it somewhere other than your main workplace – encrypt it, and lock it away if possible. That way, if there's a break-in, fire or flood, you'll minimize the risk of losing all your data. Check your back-up. You don't want to find out it's not worked when you need it most. Make sure your back-up isn't connected to your live data source, so that any malicious activity doesn't reach it. 2. Use strong passwords and multi-factor authentication Make sure you use strong passwords on smartphones, laptops, tablets, email accounts and any other devices or accounts where personal information is stored. They must be difficult to guess. The National Cyber Security Centre (NCSC) recommends using three random words. Where possible, you should consider using multi-factor authentication. Multi-factor authentication is a security measure to make sure the right person is accessing the data. It requires at least two separate forms of identification before access is granted. For example, you use a password and a one-time code which is sent by text message. 3. Be aware of your surroundings For example, if you're on a train or in a shared workspace, other people may be able to see your screen. A privacy screen might help you. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System 4. Be wary of suspicious emails You and your staff need to know how to spot suspicious emails. Look out for signs such as bad grammar, demands for you to act urgently and requests for payment. New technologies mean that email attacks are becoming more sophisticated. A phishing email could appear to come from a source you recognize. If you're not sure, speak to the sender. NCSC provide useful training materials to help you and your staff recognize suspicious emails. 5. Install anti-virus and malware protection And keep it up-to-date. You must make sure the devices you and your employees use at home, or when you're working away, are secure. Anti-virus software can help protect your device against malware sent through a phishing attack. 6. Protect your device when it's unattended Lock your screen when you're temporarily away from your desk to prevent someone else accessing your computer. If you do need to leave your device for longer, put it in a secure place, out of sight. 7. Make sure your Wi-Fi connection is secure Using public Wi-Fi, or an insecure connection, could put personal data at risk. You should make sure you always use a secure connection when connecting to the internet. If you're using a public network, consider using a secure Virtual Private Network (VPN). 8. Limit access to those who need it Different workers may need to use different types of information. Put access controls in place to make sure people can only see the information they need. For example, payroll or HR may need to see workers' personal information, but your sales staff won't. If someone leaves your company, or if they're absent for a long period of time, suspend their access to your systems. 9. Take care when sharing your screen Sharing your screen in a virtual meeting may show your device to others exactly as you see it, including any open tabs or documents. Before sharing your screen, you should close anything

you don't need and make sure your notifications and pop-up alerts are switched off. 10. Don't keep data for longer than you need it Getting rid of data you no longer need will free up storage space. This also means you have less personal information at risk if you suffer a cyber-attack or personal data breach. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System 11. Dispose of old IT equipment and records securely You must make sure no personal data is left on computers, laptops, smartphones or any other devices, before you dispose of them. You could consider using deletion software, or hire a specialist to wipe the data. Controlling security threat and vulnerability All computer systems have vulnerabilities, some simple, some complex. If a cyber-attacker tries hard enough, they will find a way to exploit vulnerability. Any attempt an organization makes to stop security threats is called a control. Most of the hard work of cyber security is selecting the right controls, and then making sure the controls are actually working. Ready for another list of three? A control can be preventative, detective or corrective. Preventative controls are things like passwords or multi-factor authentication. Detective controls are systems that flag malware or phishing attempts and the like. And corrective controls manage the aftermath of an attack using tools like incident response, forensic analysis or restoring data from backups. If implementing cybersecurity controls sounds complicated, don't worry. In the world of cybersecurity, most organizations run frameworks or prescriptive processes and controls to manage cyber risk. Control frameworks are like a box of chocolates; instead of picking and choosing each individual control, frameworks tailor controls to an organization's size and activity. Note that different industries and regulatory bodies either require or suggest frameworks your organization should implement. Six practical cybersecurity controls No matter how an organization determines what controls it needs, whether via risk assessments or adopting a framework, there are some smaller essential controls that almost every organization will use.

1. Update operating systems - When a vulnerability is found in software, the manufacturer will work out how to fix the vulnerability and provide an updated version of the software. Keeping your systems up to date will protect against recently identified vulnerabilities.

2. Whitelist applications - Whitelisting means that a computer is configured to only run the software that the organization explicitly permits. This is quite a hard control to manage, but it makes it very difficult for a cyber-attacker.

3. Harden the computer's defense - Make sure that all configurable settings in the operating system and applications are configured for security. Another recommendation is to regularly de-install parts of the operating system and applications that will never be used.

4. Limit administrative access - One of the easiest cybersecurity controls that's recommended by every framework is to limit the number of people within the organization who have administrative access to Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System systems. Reducing the number of accounts that have such access means there are fewer accounts for an attacker to target.

5. Implement multi-factor authentication - Require a user to provide something else in addition to a username and a password to log in. This could be something unique only to the user—such as a fingerprint—or a physical product that the user has, such as a smartcard or a mobile device.

6. Create safe back-ups - If an attacker gains access to a system and either tampers with, erases or encrypts data with the intent of securing a ransom from the organization, a backup helps the organization restore the data and recover its operations without paying the ransom.

Technologies for Information System Control- (What are information system controls in MIS?)

Information systems security control is comprised of the processes and practices of technologies designed to protect networks, computers, programs and data from unwanted, and most importantly, deliberate intrusions. Elements of information systems security control include:

- Identifying isolated and networked systems
- Application security
- Information security, including hard copy
- Network security (network and isolated)
- Mitigating insider vulnerabilities
- Incident response
- Training

What Are Information Security Controls?

Information security controls play an indispensable role in safeguarding valuable information assets, assuring the CIA (Central Intelligence Agency). In addition, they help to manage and mitigate security risks, including those posed by malware and other potential threats. We can group information security controls into a few broad categories.

- Access controls restrict access to information and information systems based on user privileges, authentication mechanisms like passwords, and authorization rules.
- Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System
- Encryption transforms information into an unreadable format unless you have the decryption key, assuring the information remains

protected even if unauthorized access occurs. It involves the use of cryptographic algorithms and keys.

- Firewalls control the flow of incoming and outgoing network traffic using a predetermined set of rules.
- Intrusion detection and prevention systems monitor network traffic and system activities to detect and stop unauthorized access or malicious activities. They can generate alerts or take actions to block or mitigate threats.
- Malware protection controls include antivirus software, anti-malware solutions, and other technologies. They detect, prevent, and remove malicious software (malware) such as viruses, worms, and Trojans.
- Secure configuration controls harden operating systems, applications, and devices by disabling unnecessary services and removing default passwords.
- Security incident and event management solutions collect and analyze log data from various sources to identify and respond to security incidents, detect anomalies in real-time environments, and support forensic investigations.
- Secure coding practices emphasize the importance of writing secure code to minimize vulnerabilities and prevent common coding errors that attackers could exploit.

Disaster Recovery Plan A disaster recovery plan (DRP) is a documented, structured approach that describes how an organization can quickly resume work after an unplanned incident. A DRP is an essential part of a business continuity plan (BCP). A disaster recovery plan (DRP), disaster recovery implementation plan, or IT disaster recovery plan is a recorded policy and/or process that is designed to assist an organization in executing recovery processes in response to a disaster to protect business IT infrastructure and more generally promote recovery. The purpose of a disaster recovery plan is to comprehensively explain the consistent actions that must be taken before, during, and after a natural or man-made disaster so that the entire team can take those actions. A disaster recovery plan should address both man-made disasters that are intentional, such as fallout from terrorism or hacking, or accidental, such as an equipment failure. Organizations of all sizes generate and manage massive amounts of data, much of it mission critical. The impact of corruption or data loss from human error, hardware failure, malware, or hacking can be Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System substantial. Therefore, it is essential to create a disaster recovery plan for the restoration of business data from a data backup image. It is most effective to develop an information technology (IT) disaster recovery plan in conjunction with the business continuity plan (BCP). A business continuity plan is a complete organizational plan that consists of five components: 1. Business resumption plan 2. Occupant emergency plan 3. Continuity of operations plan 4. Incident management plan (IMP) 5. Disaster recovery plan Generally, components one through three do not touch upon IT infrastructure at all. The incident management plan typically establishes procedures and a structure to address cyber attacks against IT systems during normal times, so it does not deal with the IT infrastructure during disaster recovery. For this reason, the disaster recovery plan is the only component of the BCP of interest to IT. Among the first steps in developing such a strategy is business impact analysis, during which the team should develop IT priorities and recovery time objectives. The team should time technology recovery strategies for restoring applications, hardware, and data to meet business recovery needs. Every situation is unique and there is no single correct way to develop a disaster recovery plan. However, there are three principle goals of disaster recovery that form the core of most DRPs:

- prevention, including proper backups, generators, and surge protectors
- detection of new potential threats, a natural byproduct of routine inspections
- correction, which might include holding a “lessons learned” brainstorming session and securing proper insurance policies

Disaster recovery plan - Although specific disaster recovery plan formats may vary, the structure of a disaster recovery plan should include several features: Goals A statement of goals will outline what the organization wants to achieve during or after a disaster, including the recovery time objective (RTO) and the recovery point objective (RPO). The recovery point objective refers to how much data (in terms of the most recent changes) the company is willing to lose after a disaster occurs. For example, an RPO might be to lose no more than one hour of data, which means data backups must occur at least every hour to meet this objective. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System Recovery time objective or RTO refers to the acceptable downtime after an outage before business processes and systems must be restored to operation. For example, the business must be able to return to operations within 4 hours in order to avoid unacceptable impacts to business continuity. Personnel Every disaster recovery plan must detail the personnel who are responsible for the execution of the DR plan, and make provisions for individual people becoming unavailable. IT inventory An updated IT inventory must list the details about all hardware and software assets, as well as any cloud services necessary for the company’s operation, including whether or not they are business critical, and whether they are owned, leased, or used as a service. Backup procedures The DRP must set forth how each data resource is backed up

exactly where, on which devices and in which folders, and how the team should recover each resource from backup. Disaster recovery procedures These specific procedures, distinct from backup procedures, should detail all emergency responses, including last-minute backups, mitigation procedures, limitation of damages, and eradication of cybersecurity threats. Disaster recovery sites Any robust disaster recovery plan should designate a hot disaster recovery site. Located remotely, all data can be frequently backed up to or replicated at a hot disaster recovery site — an alternative data center holding all critical systems. This way, when disaster strikes, operations can be instantly switched over to the hot site. Restoration procedures Finally, follow best practices to ensure a disaster recovery plan includes detailed restoration procedures for recovering from a loss of full systems operations. In other words, every detail to get each aspect of the business back online should be in the plan, even if you start with a disaster recovery plan template. Here are some procedures to consider at each step. Include not just objectives such as the results of risk analysis and RPOs, RTOs, and SLAs, but also a structured approach for meeting these goals. The DRP must address each type of downtime and disaster with a step-by-step plan, including data loss, flooding, natural disasters, power outages, ransomware, server failure, site-wide outages, and other issues. Be sure to enrich any IT disaster recovery plan template with these critical details.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System Create a list of IT staff including contact information, roles, and responsibilities.

Ensure each team member is familiar with the company disaster recovery plan before it is needed so that individual team members have the necessary access levels and passwords to meet their responsibilities. Always designate alternates for any emergency, even if you think your team can't be affected. Address business continuity planning and disaster recovery by providing details about mission-critical applications in your DRP. Include accountable parties for both troubleshooting any issues and ensuring operations are running smoothly. If your organization will use cloud backup services or disaster recovery services, vendor name and contact information, and a list of authorized employees who can request support during a disaster should be in the plan; ideally the vendor and organizational contacts should know of each other. Media communication best practices are also part of a robust disaster recovery and business continuity plan. A designated public relations contact and media plan are particularly useful to high profile organizations, enterprises, and users who need 24/7 availability, such as government agencies or healthcare providers. Look for disaster recovery plan examples in your industry or vertical for specific best practices and language. Emerging trends and technologies with regard to MISWhat are the emerging technologies in management information system? The Latest Trend of Information Technologies are: Cloud Computing, Internet of Things (IoT), Big data, Cyber security, Context-Rich Systems, Increased automation, Continued mobile pervasiveness, Web-Scale IT, 3D Printing. Major trends in technology that impact communication within organizations include; voice over Internet protocol, Web 2.0, and virtual networks. Unlike virtual networks, face-to-face (F2F) meetings provide limited support for global projects and establishments.

1. What are technology trends?  
Technology trends refer to the prevailing developments, innovations, and advancements in the world of technology. These trends often shape the direction of industries, businesses, and society as a whole, influencing how we interact, work, and live.

2. Why are technology trends important?  
Following technology trends is crucial for individuals and businesses alike because it allows them to stay competitive and relevant in a rapidly evolving digital landscape. By keeping abreast of emerging Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System technologies, one can make informed decisions about adopting new tools, improving processes, and leveraging opportunities for growth.

3. How do you keep up with technology trends?  
You can stay updated with technology trends by following reputable technology news sources, subscribing to industry newsletters, attending conferences and webinars, participating in online communities, and engaging in continuous learning and skill development.

Technology trends in 2023 – 2024- The most important technology trends of 2023 are:

- Artificial intelligence (AI): AI is becoming increasingly sophisticated and is being used in a wide variety of applications, from healthcare to customer service to manufacturing.
- Machine learning (ML): ML is a subset of AI that allows computers to learn from data without being explicitly programmed. ML is being used in a wide variety of applications, such as fraud detection, recommendation engines, and self-driving cars.
- Blockchain: Blockchain is a distributed ledger technology that allows for secure and transparent transactions. Blockchain is being used in a wide variety of applications, such as financial services, supply chain management, and voting.
- 5G: 5G is the next generation of cellular network technology. 5G offers significantly faster speeds and lower latency than 4G, making it ideal for applications such as augmented reality, virtual reality, and self-driving cars.
- The Internet of Things (IoT): The IoT refers to the network of

physical devices that are connected to the internet. The IoT is growing rapidly and is being used in a wide variety of applications, such as smart homes, smart cities, and industrial automation. \*\*\*\*\* Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes V-Semester AD 504 (A) Management Information System

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Name: Machine Learning Subject Code: AD 502 Subject Notes Syllabus: Dimensionality Reduction: The Curse of Dimensionality, Main Approaches for Dimensionality Reduction (Projection, Manifold Learning) PCA: Preserving the Variance, Principal Components, Projecting Down to d Dimensions, Explained Variance Ratio, Choosing the Right Number of Dimensions, PCA for Compression, Randomized PCA, Incremental PCA. Kernel PCA: Selecting a Kernel and Tuning Hyper parameters.

Course

Outcome (CO5): To apply the algorithms to a real-world problem, optimize the models learned and report on the expected accuracy that can be achieved by applying the models. Unit-V Dimensionality Reduction The number of input features, variables, or columns present in a given dataset is known as dimensionality, and the process to reduce these features is called dimensionality reduction. A dataset contains a huge number of input features in various cases, which makes the predictive modeling task more complicated. Because it is very difficult to visualize or make predictions for the training dataset with a high number of features, for such cases, dimensionality reduction techniques are required to use. Dimensionality reduction technique can be defined as, "It is a way of converting the higher dimensions dataset into lesser dimensions dataset ensuring that it provides similar information." These techniques are widely used in machine learning for obtaining a better fit predictive model while solving the classification and regression problems. It is commonly used in the fields that deal with high-dimensional data, such as speech recognition, signal processing, bioinformatics, etc. It can also be used for data visualization, noise reduction, cluster analysis, etc. Diagram 4.1 Dimensionality Reduction Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Curse of Dimensionality Handling the high-dimensional data is very difficult in practice, commonly known as the curse of dimensionality. If the dimensionality of the input dataset increases, any machine learning algorithm and model becomes more complex. As the number of features increases, the number of samples also gets increased proportionally, and the chance of overfitting also increases. If the machine learning model is trained on highdimensional data, it becomes overfitted and results in poor performance. Hence, it is often required to reduce the number of features, which can be done with dimensionality reduction.

Approaches of Dimension Reduction There are two ways to apply the dimension reduction technique, which are given below: Feature Selection Feature selection is the process of selecting the subset of the relevant features and leaving out the irrelevant features present in a dataset to build a model of high accuracy. In other words, it is a way of selecting the optimal features from the input dataset. Three methods are used for the feature selection: 1. Filters Methods In this method, the dataset is filtered, and a subset that contains only the relevant features is taken. Some common techniques of filters method are: ♣ Correlation ♣ Chi-Square Test ♣ ANOVA ♣ Information Gain, etc. 2. Wrappers Methods The wrapper method has the same goal as the filter method, but it takes a machine learning model for its evaluation. In this method, some features are fed to the ML model, and evaluate the performance. The performance decides whether to add those features or remove to increase the accuracy of the model. This method is more accurate than the filtering method but complex to work. Some common techniques of wrapper methods are: ♣ Forward Selection ♣ Backward Selection ♣ Bi-directional Elimination Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science 3.

Embedded Methods: Embedded methods check the different training iterations of the machine learning model and evaluate the importance of each feature. Some common techniques of Embedded methods are: ♣ LASSO ♣ Elastic Net ♣ Ridge Regression, etc. Feature Extraction: Feature extraction is the process of transforming the space containing many dimensions into space with fewer dimensions. This approach is useful when we want to keep the whole information but use fewer resources while processing the information. • Principal Component Analysis • Linear Discriminant Analysis • Kernel PCA • Quadratic Discriminant Analysis Projection Perspective is a machine learning technique used to reduce the dimensionality of data. There are several methods commonly used for this, such as • Principal Component Analysis (PCA) – PCA identifies the directions along which the data varies the most and projects the data onto these components. • Linear Discriminant Analysis (LDA) – LDA is used for supervised dimensionality reduction. • T-Distributed

Stochastic Neighbor Embedding (t-SNE) – t-SNE is used for visualizing clusters or groups of data points. • Autoencoders – Autoencoders are neural network architectures that can be used for unsupervised dimensionality reduction. • Random Projections – Random projections are a simple and computationally efficient method for dimensionality reduction. The Basics of Projection Perspective • Explain the concept of projection and how it relates to machine learning. • Describe the mathematical representation of the projection perspective. • Introduce the concept of feature space and target space.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Projection Techniques • Orthogonal Projection – Discuss how orthogonal projection projects data onto a lower-dimensional subspace. • Principal Component Analysis (PCA) – Explain how PCA employs projection perspective to reduce dimensionality while retaining the most important information. • Linear Discriminant Analysis (LDA) – Describe how LDA utilizes projection perspective for feature extraction and classification. • t-SNE – Briefly discuss how t-SNE uses projection perspective to visualize high-dimensional data in a lower-dimensional space. Manifold Manifolds describe a vast number of geometric surfaces. To be a manifold, there's one important rule that needs to be satisfied. The best way to understand this property is through example. Manifolds exist in any dimension, but for the sake of simplicity, let's think about a three-dimensional space. Suppose there is a small ant walking along a manifold in three dimensions. This manifold could be curvy, twisty, or even have holes in it. Now here's the rule: From the point of view of the ant, everywhere it walks should look like a flat plane. Does this rule sound familiar? If you're looking for an application I think this is one that all of us can relate to; we live on a manifold! A sphere is one of the simplest examples of a manifold in three dimensions. Principal Component Analysis Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the Principal Components. It is one of the popular tools that is used for exploratory data analysis and predictive modeling. It is a technique to draw strong patterns from the given dataset by reducing the variances. PCA generally tries to find the lower-dimensional surface to project the high-dimensional data. PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are image processing, movie recommendation system, optimizing the power allocation in various communication channels. It is a feature extraction technique, so it contains the important variables and drops the least important variable. The PCA algorithm is based on some mathematical concepts such as:

- Variance and Covariance
- Eigenvalues and Eigen factors

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science

Some common terms used in PCA algorithm:

- Dimensionality: It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- Correlation: It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.
- Orthogonal: It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
- Eigenvectors: If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.
- Covariance Matrix: A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

Principal Components in PCA the transformed new features or the output of PCA are the Principal Components. The number of these PCs are either equal to or less than the original features present in the dataset. Some properties of these principal components are given below:

- The principal component must be the linear combination of the original features.
- These components are orthogonal, i.e., the correlation between a pair of variables is zero.
- The importance of each component decreases when going to 1 to n, it means the 1 PC has the most importance, and n PC will have the least importance.

Steps for PCA algorithm

Getting the dataset

Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.

Representing data into a structure

Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.

Standardizing the data

In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance. If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name

the matrix as Z. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Calculating the Covariance of Z To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z. Calculating the Eigen Values and Eigen Vectors Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues. Sorting the Eigen Vectors In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P\*. Calculating the new features Or Principal Components Here we will calculate the new features. To do this, we will multiply the P\* matrix to the Z. In the resultant matrix Z\*, each observation is the linear combination of original features. Each column of the Z\* matrix is independent of each other. Remove less or unimportant features from the new dataset. The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out. Backward EliminationBackward elimination is a feature selection technique while building a machine learning model. It is used to remove those features that do not have a significant effect on the dependent variable or prediction of output. Steps of Backward Elimination Below are some main steps which are used to apply backward elimination process: Step-1: Firstly, We need to select a significance level to stay in the model. (SL=0.05) Step-2: Fit the complete model with all possible predictors/independent variables. Step-3: Choose the predictor which has the highest P-value, such that. If P-value >SL, go to step 4. Else Finish, and Our model is ready. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Step-4: Remove that predictor. Step-5: Rebuild and fit the model with the remaining variables.

Chameli Devi Group of Institutions Department of Artificial Intelligence and Data Science AD 603 (A) Data Mining and Warehousing B. Tech VI Semester Unit -II .....

Syllabus: Unit-II: OLAP Systems: Basic concepts, OLAP queries, Types of OLAP servers, OLAP Operations etc. Data Warehouse Hardware and Operational Design: Security, Backup and Recovery.

..... OLAP System: Basic ConceptsOLAP (Online Analytical Processing) is the technology support the multidimensional view of data for many Business Intelligence (BI) applications. OLAP provides fast, steady and proficient access, powerful technology for data discovery, including capabilities to handle complex queries, analytical calculations, and predictive “what if” scenario planning. OLAP is a category of software technology that enables analysts, managers and executives to gain insight into data through fast, consistent, interactive access in a wide variety of possible views of information that has been transformed from raw data to reflect the real dimensionality of the enterprise as understood by the user. OLAP enables end-users to perform ad hoc analysis of data in multiple dimensions, thereby providing the insight and understanding they need for better decision making. Characteristics of OLAP SystemThe need for more intensive decision support prompted the introduction of a new generation of tools. Generally used to analyze the information where huge amount of historical data is stored. Those new tools, called online analytical processing (OLAP), create an advanced data analysis environment that supports decision making, business modeling, and operations research. Its four main characteristics are: 1. Multidimensional data analysis techniques 2. Advanced database support 3. Easy to use end user interfaces 4. Support for client/server architecture. 1. Multidimensional Data Analysis Techniques: Multidimensional analysis is inherently representative of an actual business model. The most distinctive characteristic of modern OLAP tools is their capacity for multidimensional analysis (for example actual vs budget). In multidimensional analysis, data are processed and viewed as part of a multidimensional structure. This type of data analysis is particularly attractive to business decision makers because they tend to view business data as data that are related to other business data. 2. Advanced Database Support: → For efficient decision support, OLAP tools must have advanced data access features. Access too many different kinds of DBMSs, flat files, and internal and external data sources. → Access to aggregated data warehouse data as well as to the detail data found in operational databases. → Advanced data navigation features such as drill-down and roll-up. → Rapid and consistent query response times. → The ability to map end-user requests, expressed in either business or model terms, to the appropriate data source and then to the proper data access language (usually SQL). → Support for very large databases. As already

explained the data warehouse can easily and quickly grow to multiple gigabytes and even terabytes.

3. Easy-to-Use End-User Interface: Advanced OLAP features become more useful when access to them is kept simple. OLAP tools have equipped their sophisticated data extraction and analysis tools with easy-to-use graphical interfaces. Many of the interface features are “borrowed” from previous generations of data analysis tools that are already familiar to end users. This familiarity makes OLAP easily accepted and readily used.

4. Client/Server Architecture: Conform the system to the principals of Client/server architecture to provide a framework within which new systems can be designed, developed, and implemented. The client/server environment enables an OLAP system to be divided into several components that define its architecture. Those components can then be placed on the same computer, or they can be distributed among several computers. Thus, OLAP is designed to meet ease-of-use requirements while keeping the system flexible.

V Motivation for using OLAP

- I) Understanding and improving sales: For an enterprise that has many products and uses a number of channels for selling the products, OLAP can assist in finding the most popular products and the most popular channels. In some cases it may be possible to find the most profitable customers.
- II) Understanding and reducing costs of doing business: Improving sales is one aspect of improving a business, the other aspect is to analyze costs and to control them as much as possible without affecting sales. OLAP can assist in analyzing the costs associated with sales.

Guidelines for OLAP Implementation

Following are a number of guidelines for successful implementation of OLAP. The guidelines are, somewhat similar to those presented for data warehouse implementation.

- 1. Vision: The OLAP team must, in consultation with the users, develop a clear vision for the OLAP system. This vision including the business objectives should be clearly defined, understood, and shared by the stakeholders.
- 2. Senior management support: The OLAP project should be fully supported by the senior managers and multidimensional view of data. Since a data warehouse may have been developed already, this should not be difficult.
- 3. Selecting an OLAP tool: The OLAP team should familiarize themselves with the ROLAP and MOLAP tools available in the market. Since tools are quite different, careful planning may be required in selecting a tool that is appropriate for the enterprise. In some situations, a combination of ROLAP and MOLAP may be most effective.
- 4. Corporate strategy: The OLAP strategy should fit in with the enterprise strategy and business objectives. A good fit will result in the OLAP tools being used more widely.
- 5. Focus on the users: The OLAP project should be focused on the users. Users should, in consultation with the technical professional, decide what tasks will be done first and what will be done later. Attempts should be made to provide each user with a tool suitable for that person's skill level and information needs. A good GUI user interface should be provided to non-technical users.
- 6. Joint management: The OLAP project must be managed by both the IT and business professionals. Many other people should be involved in supplying ideas. An appropriate committee structure may be necessary to channel these ideas.
- 7. Review and adapt: As noted in last chapter, organizations evolve and so must the OLAP systems. Regular reviews of the project may be required to ensure that the project is meeting the current needs of the enterprise.

OLTP vs. OLAP

Fig. Difference between OLAP and OLTP

OLAP Queries

OLAP queries are complex queries that:

- Touch large amounts of data
- Discover patterns and trends in the data
- Typically expensive queries that take long time
- Also called decision-support queries

Example-I: Query Syntax: `SELECT ...GROUP BY ROLLUP ( grouping_Column_reference_list);`

Example-II: `SELECT Time, Location, product, sum(revenue) AS Profit FROM sales GROUP BY ROLLUP (Time, Location, product);`

The Query calculates the standard aggregate values specified in the GROUPBY clause. Then, it creates progressively higher-level subtotals, moving from right to left through the list of grouping columns. Finally, it creates a grand total.

OLAP Servers

Online Analytical Processing Server (OLAP) is based on the multidimensional data model. It allows managers, and analysts to get an insight of the information through fast, consistent, and interactive access to information.

Types of OLAP Servers

We have four types of OLAP servers

- Relational OLAP (ROLAP)
- Multidimensional OLAP (MOLAP)
- Hybrid OLAP (HOLAP)
- Specialized SQL Servers

Relational OLAP: ROLAP servers are placed between relational back-end server and client front-end tools. To store and manage warehouse data, ROLAP uses relational or extended-relational DBMS. ROLAP includes the following

- Implementation of aggregation navigation logic
- Optimization for each DBMS back end
- Additional tools and services
- Can handle large amounts of data
- Performance can be slow

Multidimensional OLAP: MOLAP uses array-based multidimensional storage engines for multidimensional views of data.

- Multidimensional data stores
- The storage utilization may be low if the data set is sparse.
- MOLAP server uses two levels of data storage representation to handle dense and sparse datasets.

Hybrid OLAP: Hybrid OLAP technologies attempt to combine the advantages of MOLAP and ROLAP. It offers higher scalability of ROLAP and faster computation of MOLAP.

HOLAP servers allow storing the large data volumes of detailed information. The aggregations are stored separately in MOLAP store. Specialized SQL Servers.

Specialized SQL servers provide advanced query language and query processing support for SQL queries over star and snowflake schemas in a read-only environment.

**OLAP Operations:** Four types of analytical operations in OLAP are

1. Roll-up
2. Drill-down
3. Slice and dice
4. Pivot (rotate)

**Roll-up:** Roll-up is also known as "consolidation" or "aggregation." The Roll-up operation can be performed in 2 ways:

1. Reducing dimensions
2. Climbing up concept hierarchy.

Concept hierarchy is a system of grouping things based on their order or level. Consider the following diagram: Fig: Roll-up operation

- In this example, cities New Jersey and Los Angeles are rolled up into country USA.
- The sales figure of New Jersey and Los Angeles are 440 and 1560 respectively. They become 2000 after roll-up
- In this aggregation process, data is location hierarchy moves up from city to the country.
- In the roll-up process at least one or more dimensions need to be removed. In this example, Quarter dimension is removed.

**2) Drill-down:** In drill-down data is fragmented into smaller parts. It is the opposite of the rollup process. It can be done via

- Moving down the concept hierarchy
- Increasing a dimension

**Figure: Drill-down operation** Consider the diagram above:

- Quarter Q1 is drilled down to months January, February, and March. Corresponding sales are also registers.
- In this example, dimension months are added.

**Slice:** Here, one dimension is selected, and a new sub-cube is created. Following diagram explain how slice operation performed: Figure 2.4: Slice operation

- Dimension Time is Sliced with Q1 as the filter.
- A new cube is created altogether.

**Dice:** This operation is similar to a slice. The difference in dice is to select 2 or more dimensions that result in the creation of a sub-cube. Figure 2.5: Dice Pivot: In Pivot, rotate the data axes to provide a substitute presentation of data. In the following example, the pivot is based on item types. Figure 2.6: Pivot operation

**Hardware and operational design**

**Server Hardware:** Two main hardware architectures → Symmetric Multi-Processing (SMP) → Massively Parallel Processing (MPP)

- SMP machine is a set of tightly coupled CPUs that share memory and disk.
- MPP machine is a set of loosely coupled CPUs, each of which has its own memory and disk.

**Symmetric Multi-Processing (SMP):** An SMP machine is a set of CPUs that share memory and disk. This is sometimes called a shared-everything environment the CPUs in an SMP machine are all equal a process can run on any CPU in the machine, run on different CPUs at different times.

**Scalability of SMP machines:** Length of the communication bus connecting the CPUs is a natural limit. As the bus gets longer the inter-process or communication speeds become slower, each extra CPU imposes an extra, band with load on the bus, increases memory contention, and so on.

**Massively Parallel Processing (MPP):** Made up of many loosely coupled nodes linked together by a high-speed connection. Each node has its own memory, and the disks are not shared. Most MPP system allow a disk to be dual connected between two nodes.

**Shared Nothing Environment**

- a) Nothing is shared directly. Many different models of MPP → Nodes with two CPUs -One CPU is dedicated to handling I/O -Nodes that are genuine SMP machines → Single CPU node -Some will be dedicated to handling I/O and others to processing
- b) Level of shared nothing varies from vendor to vendor

**Example:** IBM SP/2 is a. fully shared nothing system Virtual Shared Disk (VSD) → An extra layer of software to be added to allow disks to be shared across nodes → System will suffer overheads if an I/O is issued and data has to be shipped from node to node.

**Distributed Lock Manager** → MPP machines require this to maintain the integrity of the distributed resources → Track which node's memory holds the current copy of each piece of information → Getting the data rapidly from node to node

**Network Hardware Network Architecture** → Sufficient bandwidth to supply the data feed and user requirements . Impact to design → User access via WAN → impacts the design of Query Manager → Source system data transfer → Data extractions

**Example:** Problem of getting the data from the source systems. It may not get the data to the warehouse system early enough to allow it to be loaded, transformed, processed and backed up within the overnight time window.

**Guideline** → Ensure that the network architecture and bandwidth are capable of supporting the data transfer and any data extractions in an acceptable time. → The transfer of data to be loaded must be complete quickly enough to allow the rest of the overnight processing to complete.

**Client Hardware Client Management** → Those responsible for client machine management will need to know the requirements for that machine to access the data warehouse system. → Details such as the network protocols supported on the server, and the server's Internet address, will need to be supplied. → If multiple access paths to the server system exist this information needs to be relayed to those responsible for the client systems. → During node fail over users may need to access a different machine address.

**Client Tools** → The tool should not be allowed to affect the basic design of the warehouse itself. → Multiple tools will be used against the data warehouse. → Should be thoroughly tested and trialed to ensure that they are suitable for the users.

→ Testing of the tools should ideally be performed in parallel with the data warehouse design: → Usability issues to be exposed, → Drive out any requirements that the tool will place on the data warehouse

**Disk Technology: RAID Technology**

**Redundant Array of Inexpensive Disks** → The purpose of RAID technology is to provide resilience against disk failure, so that the loss of an individual disk does not mean loss of data. → Striping is a technique in which data is spread across multiple disks. → RAID levels 0, 1 and 5 are commercially viable and thus widely available **Table: RAID Level with Descriptions**

**What is Data Warehouse Security?** Data warehouses pull data from many different sources, and warehouses have many moving parts. Security issues arise every time data moves from one place to another. Data warehouse security entails taking the necessary steps to protect information so that only authorized personnel can access it. Data warehouse security should involve the following:

- Strict user access control so that users of the warehouse can only access data they need to do their jobs.
- Taking steps to secure networks on which data is held.
- Carefully moving data and considering the security implications involved in all data movement.

**Physical Security Practices**

1. Restricting and controlling physical access to data warehouses has been made easy thanks to tech innovations like biometric readers, anti-tailgating systems and other physical access control mechanisms. These might look excessive and expenditure overhead, but they play a crucial role in ensuring the integrity and safety of the precious enterprise data.
2. Imparting information about security protocols and ensuring all the personnel in the proximity of the data warehouse religiously obey and adhere to these rules is one of the keys to success. It's understandable that an employee can be used by intruders to gain access, but if the employee in question is ardently following the specified guidelines it makes a world of difference.

**Software-Based Security Measures**

**Data Encryption:** Data encryption is one of the primary safeguards against data theft. All data should be encrypted using algorithms like AES (advanced encryption standard) or FIPS 140-2 certified software for data encryption, whether it's in the transactional database or the data warehouse. Some proponents would argue that data encryption adversely affects the performance and data access speed of data centers, but considering the alternative, it is preferred.

**Data Segmenting and Partitioning:** Data encryption although an added security measure can be quite cumbersome if applied without segmenting and partitioning. Segmenting and partitioning entail classifying or splitting data into sensitive and non-sensitive information. After going through partitioning the data should be accordingly encrypted and put into separate tables ready for consumption.

**Securing On-The-Move:** Data Securing data in a single place and transit are two different ball games. Here data in transit means the one which is being relayed from transactional databases in real time to the data warehouse. These transactional databases can be anywhere geographically, therefore using protective protocols, such as SSL or TSL is highly recommended. Cloud-based data warehouses nowadays provide a secure and impenetrable tunnel between the database and the cloud storage which should be leveraged.

**Trusted Witness Server:** As mentioned earlier, hackers and intruders nowadays have become as skilled and sophisticated as the security measures they are up against. Implementing a trusted witness server is akin to hiring a watchdog that avidly and quite tenaciously keeps vigil on your data access points. It can detect an unwarranted and suspicious attempt at accessing data and generate an alert immediately. This allows the people responsible for the data warehouse security to stop the intruders dead in their tracks.

**Backup and recovery of Data Warehouse** Backup and recovery are among the most important tasks for an administrator, and data warehouses are no different. However, because of the sheer size of the database, data warehouses introduce new challenges for an administrator in the backup and recovery area. Data warehouses are unique in that the data can come from a myriad of resources and it is transformed before finally being inserted into the database; but mostly because it can be very large. Managing the recovery of a large data warehouse can be a daunting task and traditional OLTP backup and recovery strategies may not meet the needs of a data warehouse.

**Strategies and Best Practices for Backup and Recovery**

Devising a backup and recovery strategy can be a daunting task. And when you have hundreds of gigabytes of data that must be protected and recovered in the case of a failure, the strategy can be very complex. The following best practices can help you implement your warehouse's backup and recovery strategy:

- Best Practice A: Use ARCHIVELOG Mode
- Best Practice B: Use RMAN
- Best Practice C: Use Read-Only Table spaces
- Best Practice D: Plan for NOLOGGING Operations
- Best Practice E: Not All Table spaces are Equally Important

**Best Practice-A:** Use ARCHIVELOG Mode Archived redo logs are crucial for recovery when no data can be lost, because they constitute a record of changes to the database. Oracle can be run in either of two modes:

- ARCHIVELOG --Oracle archives the filled online redo log files before reusing them in the cycle.
- NOARCHIVELOG --Oracle does not archive the filled online redo log files before reusing them in the cycle.

Running the database in "ARCHIVELOG" mode has the following benefits:

- The database can be completely recovered from both instance and media failure.
- The user can perform backups while the database is open and available for use.
- Oracle supports multiplexed archive logs to avoid any possible single point of failure on the archive logs
- The user has more recovery options, such as the ability to perform table space-point-in-time

recovery (TSPITR). → Archived redo logs can be transmitted and applied to the physical standby database, which is an exact replica of the primary database. → The database can be completely recovered from both instance and media failure. Running the database in “NOARCHIVELOG” mode has the following consequences: → The user can only back up the database while it is completely closed after a clean shutdown. → Typically, the only media recovery option is to restore the whole database, which causes the loss of all transactions since the last backup. Best Practice-B: Use RMAN There are many reasons to adopt RMAN. Some of the reasons to integrate RMAN into your backup and recovery strategy are that it offers: → Extensive reporting → Incremental backups → Downtime free backups → Backup and restore validation → Backup and restore optimization → Easily integrates with media managers → Block media recovery → Archive log validation and management → Corrupt block detection Best Practice C: Use Read-Only Table spaces Best Practice D: Plan for NOLOGGING Operation Best Practice E: Not All Table spaces are Equally Important

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Subject Notes AD604[C] Compiler Design UNIT- I: Introduction to Compiling & Lexical Analysis Introduction of Compiler, Major Data Structure in Compiler, BOOT Strapping & Porting, Compiler Structure: Analysis-Synthesis Model of Compilation, Various Phases of a Compiler, Lexical Analysis: Input Buffering, Specification & Recognition of Tokens, LEX. 1. Introduction of Compiler A Compiler is a translator from one language, the input or source language, to another language, the output or target language. Often, but not always, the target language is an assembler language or the machine language for a computer processor. Note that using a compiler requires a two-step process to run a program. 1. Execute the compiler (and possibly an assembler) to translate the source program into a machine language program. 2. Execute the resulting machine language program, supplying appropriate input. Compiler is a translator program that translates a program written in (HLL) the source program and translates it into an equivalent program in (MLL) the target program. As an important part of a compiler is error showing to the programmer. Executing a program written in HLL programming language is basically of two parts. The source program must first be compiled translated into an object program. Then the results object program is loaded into a memory executed. 1.1 Language Processing System We have learnt that any computer system is made of hardware and software. The hardware understands a language, which humans cannot understand. So we write programs in high-level language, which is easier for us to understand and remember. These programs are then fed into a series of tools and OS components to get the desired code that can be used by the machine. This is known as Language Processing System. The high-level language is converted into binary language in various phases. A compiler is a program that converts high-level language to assembly language. Similarly, an assembler is a program that converts the assembly language to machine-level language. Let us first understand how a program, using C compiler, is executed on a host machine.

- User writes a program in C language (high-level language).
- The C compiler compiles the program and translates it to assembly program (low-level language).
- An assembler then translates the assembly program into machine code (object).
- A linker tool is used to link all the parts of the program together for execution (executable machine code).
- A loader loads all of them into memory and then the program is executed.

Before diving straight into the concepts of compilers, we should understand a few other tools that work closely with compilers. Preprocessors Preprocessors are normally fairly simple as in the C language, providing primarily the ability to include files and expand macros. There are exceptions, however. IBM's PL/I, another Algol-like language had quite an extensive preprocessor, which made available at preprocessor time, much of the PL/I language itself (e.g., loops and I believe procedure calls). Assemblers Assembly code is a mnemonic version of machine code in which names, rather than binary values, are used for machine instructions, and memory addresses. Some processors have fairly regular operations and as a result assembly code for them can be fairly natural and not-too-hard to understand. Other processors, in particular Intel's x86 line, have let us charitably say more interesting instructions with certain registers used for certain things. Linkers The linker has another input, namely libraries, but to the linker the libraries look like other programs compiled and assembled. The two primary tasks of the linker are

- Relocating relative addresses.
- Resolving external references (such as the procedure xor() above).

Loaders After the linker has done its work, the resulting “executable file” can be loaded by the operating system into central memory. The details are OS dependent. With early single-user operating systems all programs would be loaded into a fixed address (say 0) and the loader simply copies the file to memory. Today it is much more complicated since (parts of) many programs reside in memory at the same time. Hence the compiler/assembler/linker cannot know the real location for an identifier. Indeed, this real location

can change. Compiler A Compiler is a translator from one language, the input or source language, to another language, the output or target language. Often, but not always, the target language is an assembler language or the machine language for a computer processor. Executing a program written in HLL programming language is basically of two parts. The source program must first be compiled translated into an object program. Then the results object program is loaded into a memory executed. Assembler Programmers found it difficult to write or read programs in machine language. They begin to use a mnemonic (symbols) for each machine instruction, which they would subsequently translate into machine language. Such a mnemonic machine language is now called an assembly language. Programs known as assembler were written to automate the translation of assembly language into machine language. The input to an assembler program is called source program, the output is a machine language translation (object program). Interpreter An interpreter is a program that appears to execute a source program as if it were machine language. Execution in Interpreter Languages such as BASIC, SNOBOL, LISP can be translated using interpreters. JAVA also uses interpreter. The process of interpretation can be carried out in following phases. 1. Lexical analysis 2. Syntax analysis 3. Semantic analysis 4. Direct Execution Advantages: Modification of user program can be easily made and implemented as execution proceeds. Type of object that denotes various may change dynamically. Debugging a program and finding errors is simplified task for a program used for interpretation. The interpreter for the language makes it machine independent. Disadvantages: The execution of the program is slower. Memory consumption is more. 2. Major Data Structure in Compiler Symbol Tables Symbol Tables are organized for fast lookup. Items are typically entered once and then looked up several times. Hash Tables and Balanced Binary Search Trees are commonly used. Each record contains a "name" (symbol) and information describing it. Simple Hash Table Hasher translates "name" into an integer in a fixed range- the hash value. Hash Value indexes into an array of lists. Entry with that symbol is in that list or is not stored at all. Items with same hash value = bucket. Balanced Binary Search Tree Binary search trees work if they are kept balanced. Can achieve logarithmic lookup time. Algorithms are somewhat complex. Red-black trees and AVL trees are used. No leaf is much farther from root than any other. Parse Tree The structure of a modern computer language is tree-like. Trees represent recursion well. A grammatical structure is a node with its parts as child nodes. Interior nodes are non-terminals. The tokens of the language are leaves. 3. Bootstrapping& Porting Bootstrapping is a technique that is widely used in compiler development. It has four main uses:

- It enables new programming languages and compilers to be developed starting from existing ones.
- It enables new features to be added to a programming language and its compiler.
- It also allows new optimizations to be added to compilers.
- It allows languages and compilers to be transferred between processors with different instruction sets

A compiler is characterized by three languages:

- Source Language
- Target Language
- Implementation Language

Figure 1.1: T-Diagram Notation: Represents a compiler for Source S, Target T, implemented in I. The T-diagram shown above is also used to depict the same compiler. To create a new language, L, for machine A: 1. Create , a compiler for a subset, S, of the desired language, L, using language A, which runs on machine A. (Language A may be assembly language.) 2. Create , a compiler for language L written in a subset of L. 3. Compile using to obtain , a compiler for language L, which runs on machine A and produces code for machine A. Figure 1.2: Bootstrapping of Compiler The process illustrated by the T-diagrams is called bootstrapping and can be summarized by the equation: To produce a compiler for a different machine B: 1. Convert into (by hand, if necessary). Recall that language S is a subset of language L. 2. Compile to produce , a cross-compiler for L which runs on machine A and produces code for machine B. 3. Compile with the cross-compiler to produce , a compiler for language L which runs on machine B. Figure 1.3: Porting of Compiler 4. Structure of The Compiler: Analysis and Synthesis Model of Compilation A compiler can broadly be divided into two phases based on the way they compile. Analysis Model Known as the front-end of the compiler, the analysis phase of the compiler reads the source program, divides it into core parts and then checks for lexical, grammar and syntax errors. The analysis phase generates an intermediate representation of the source program and symbol table, which should be fed to the Synthesis phase as input. Figure 1.4: Analysis and Synthesis phase of Compiler Synthesis Model Known as the back-end of the compiler, the synthesis phase generates the target program with the help of intermediate source code representation and symbol table. A compiler can have many phases and passes.

- Pass: A pass refers to the traversal of a compiler through the entire program.
- Phase: A phase of a compiler is a distinguishable stage, which takes input from the previous stage, processes and yields output that can be used as input for the next stage. A pass can have more than one phase.

4.1 Various Phase of Compiler Phases of a compiler: A compiler operates in phases. A phase is a logically interrelated operation that takes source program in one representation and produces output in another representation. Compilation process is partitioned

into no-of-sub processes called ‘phases’.The phases of a compiler are shown in below.

**Lexical Analysis:-** Lexical Analysis or Scanners reads the source program one character at a time, carving the source program into a sequence of character units called tokens.

**Syntax Analysis:-** The second stage of translation is called Syntax analysis or parsing. In this phase expressions, statements, declarations etc... are identified by using the results of lexical analysis. Syntax analysis is aided by using techniques based on formal grammar of the programming language.

**Front end Source Code Machine Code Errors IR Back end**

**Figure 1.5: Phases of Compiler Semantic Analysis**

There is more to a front end than simply syntax. The compiler needs semantic information, e.g., the types (integer, real, pointer to array of integers, etc.) of the objects involved. This enables checking for semantic errors and inserting type conversion where necessary.

**Intermediate Code Generations:-** An intermediate representation of the final machine language code is produced. This phase bridges the analysis and synthesis phases of translation.

**Code Optimization:-** This is optional phase described to improve the intermediate code so that the output runs faster and takes less space.

**Source Program LexicalAnalyzer Syntax Analyzer Semantic Analyzer Intermediate CodeGenerator Code Optimizer Code Generator Target Program SymbolTableManager ErrorHandler Code Generation:-**

The last phase of translation is code generation. A number of optimizations to reduce the length of machine language program are carried out during this phase. The output of the code generator is the machine language program of the specified computer.

**Symbol-Table Management** The symbol table stores information about program variables that will be used across phases. Typically, this includes type information and storage location. A possible point of confusion: the storage location does not give the location where the compiler has stored the variable. Instead, it gives the location where the compiled program will store the variable.

**Error Handlers** It is invoked when a flaw error in the source program is detected. The output of LA is a stream of tokens, which is passed to the next phase, the syntax analyzer or parser. The SA groups the tokens together into syntactic structure called as expression. Expression may further be combined to form statements. The syntactic structure can be regarded as a tree whose leaves are the token called as parse trees.

## 5. LEXICAL ANALYSIS

To identify the tokens we need some method of describing the possible tokens that can appear in the input stream. For this purpose we introduce regular expression, a notation that can be used to describe essentially all the tokens of programming language.

o Secondly , having decided what the tokens are, we need some mechanism to recognize these in the input stream. This is done by the token recognizers, which are designed using transition diagrams and finite automata.

**Figure 1.6: Lexical Analysis phase Role Of Lexical Analyzer**

The LA is the first phase of a compiler. Its main task is to read the input character and produce as output a sequence of tokens that the parser uses for syntax analysis. Upon receiving a ‘get next token’ command from the parser, the lexical analyzer reads the input character until it can identify the next token. The LA returns to the parser representation for the token it has found. The representation will be an integer code, if the token is a simple construct such as parenthesis, comma or colon. LA may also perform certain secondary tasks as the user interface. One such task is stripping out from the source program the commands and white spaces in the form of blank, tab and new line characters. Another is correlating error message from the compiler with the source program.

**Token** Token is a sequence of characters that can be treated as a single logical entity. Typical tokens are,

- 1) Identifiers
- 2) keywords
- 3) operators
- 4) special symbols
- 5) constants

**Source Program Lexical Analysis Parser Symbol table Token Get next Token Pattern**

A set of strings in the input for which the same token is produced as output. This set of strings is described by a rule called a pattern associated with the token.

**Lexeme** A lexeme is a sequence of characters in the source program that is matched by the pattern for a token.

### 5.1 Lexical Errors

Lexical errors are the errors thrown by your lex when unable to continue. Which means that there’s no way to recognize a lexeme as a valid token for you lex?

Syntax errors, on the other side, will be thrown by your scanner when a given set of already recognized valid tokens don’t match any of the right sides of your grammar rules. Simple panic-mode error handling system requires that we return to a high-level parsing function when a parsing or lexical error is detected. Error-recovery actions are:

- a. Delete one character from the remaining input.
- b. Insert a missing character in to the remaining input.
- c. Replace a character by another character.
- d. Transpose two adjacent characters.

### 5.2 Lexical Analysis: Input Buffer

The lexical analyzer scans the characters of the source program one at a time to discover tokens. Often, however, many characters beyond the next token many have to be examined before the next token itself can be determined. For this and other reasons, it is desirable for the lexical analyzer to read its input from an input buffer. Fig. 1.8 shows a buffer divided into two halves of, say 100 characters each. One pointer marks the beginning of the token being discovered. A look ahead pointer scans ahead of the beginning point, until the token is discovered .we view the position of each pointer as being between the character last read and the character next to be read. In practice each buffering scheme adopts one convention either a pointer is at the symbol last

read or the symbol it is ready to read. The distance which the look ahead pointer may have to travel past the actual token may be large. For example, in a PL/I program we may see: DECLARE (ARG1, ARG2... ARG n) Without knowing whether DECLARE is a keyword or an array name until we see the character that follows the right parenthesis. In either case, the token itself ends at the second E. If the look ahead pointer travels beyond the buffer half in which it began, the other half must be loaded with the next characters from the source file. Since the buffer shown in figure is of limited size there is an implied constraint on how much look ahead can be used before the next token is discovered. In the above example, if the look ahead traveled to the left half and all the way through the left half to the middle, we could not reload the right half, because we would lose characters that had not yet been grouped into tokens. While we can make the buffer larger if we chose or use another buffering scheme, we cannot ignore the fact that overhead is limited. Buffer Pairs A buffer is divided into two N-character halves, as shown below Figure 1.7: An input buffer in two halves Each buffer is of the same size N, and N is usually the number of characters on one block. E.g., 1024 or 4096 bytes. Using one system read command we can read N characters into a buffer. If fewer than N characters remain in the input file, then a special character, represented by eof, marks the end of the source file. Two pointers to the input are maintained:

- Pointer lexeme beginning marks the beginning of the current lexeme, whose extent we are attempting to determine.
- Pointer forward scans ahead until a pattern match is found. Once the next lexeme is determined, forward is set to the character at its right end.
- The string of characters between the two pointers is the current lexeme.
- After the lexeme is recorded as an attribute value of a token returned to the parser, lexeme beginning is set to the character immediately after the lexeme just found. Advancing forward pointer: Advancing forward pointer requires that we first test whether we have reached the end of one of the buffers, and if so, we must reload the other buffer from the input, and move forward to the beginning of the newly loaded buffer. If the end of second buffer is reached, we must again reload the first buffer with input and the pointer wraps to the beginning of the buffer. Code to advance forward pointer: if forward at end of first half then begin reload second half; forward := forward + 1 end else if forward at end of second half then begin reload second half; move forward to beginning of first half end else forward := forward + 1; Sentinels For each character read, we make two tests: one for the end of the buffer, and one to determine what character is read. We can combine the buffer-end test with the test for the current character if we extend each buffer to hold a sentinel character at the end. The sentinel is a special character that cannot be part of the source program, and a natural choice is the character eof. The sentinel arrangement is as shown below:

```
:::E
::=::M::eof C::*:2::eof:::eof
```

Figure 1.8: Sentinels at end of each buffer half lexeme beginning forward Note that eof retains its use as a marker for the end of the entire input. Any eof that appears other than at the end of a buffer means that the input is at an end. Code to advance forward pointer: forward := forward + 1; if forward ↑ = eof then begin if forward at end of first half then begin reload second half; forward := forward +1 end else if forward at end of second half then begin reload first half; move forward to beginning of first half end else /\* eof within a buffer signifying end of input \*/ terminate lexical analysis end

### 5.3.Specification & Recognition Of Token

Let us understand how the language theory undertakes the following terms:

  - Alphabets Any finite set of symbols {0,1} is a set of binary alphabets, {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F} is a set of Hexadecimal alphabets, {a-z, A-Z} is a set of English language alphabets.
  - Strings Any finite sequence of alphabets is called a string. Length of the string is the total number of occurrence of alphabets, e.g., the length of the string Compiler is 8.
  - Special Symbols A typical high-level language contains the following symbols:-

Arithmetic Symbols Addition(+), Subtraction(-), Modulo(%), Multiplication(\*), Division(/) Punctuation Comma(,), Semicolon(;), Dot(.), Arrow(->) Assignment = Special Assignment +=, /=, \*=, -= Comparison ==, !=, <=, >, >= Preprocessor # Location Specifies & Logical &, &&, |, ||, ! Shift Operator >>, >>>, <<< Table 1.1: Lex Symbol Description Language A language is considered as a finite set of strings over some finite set of alphabets. Computer languages are considered as finite sets, and mathematically set operations can be performed on them. Finite languages can be described by means of regular expressions. Longest Match Rule When the lexical analyzer read the source-code, it scans the code letter by letter; and when it encounters a whitespace, operator symbol, or special symbols, it decides that a word is completed. For example: int intValue; While scanning both lexemes till 'int', the lexical analyzer cannot determine whether it is a keyword int or the initials of identifier int value. The Longest Match Rule states that the lexeme scanned should be determined based on the longest match among all the tokens available. The lexical analyzer also follows rule priority where a reserved word, e.g., a keyword, of a language is given priority over user input. That is, if the lexical analyzer finds a lexeme that matches with any existing reserved word, it should generate an error. Regular Expressions in Compiler design The lexical analyzer needs to scan and identify only a finite set of valid string/token/lexeme that belong to the language in hand. It

searches for the pattern defined by the language rules. Regular expressions have the capability to express finite languages by defining a pattern for finite strings of symbols. The grammar defined by regular expressions is known as regular grammar. The language defined by regular grammar is known as regular language. Regular expression is an important notation for specifying patterns. Each pattern matches a set of strings, so regular expressions serve as names for a set of strings. Programming language tokens can be described by regular languages. The specification of regular expressions is an example of a recursive definition. Regular languages are easy to understand and have efficient implementation. There are a number of algebraic laws that are obeyed by regular expressions, which can be used to manipulate regular expressions into equivalent forms. Operations The various operations on languages are:
 

- Union of two languages L and M is written as  $L \cup M = \{s \mid s \text{ is in } L \text{ or } s \text{ is in } M\}$
- Concatenation of two languages L and M is written as  $LM = \{st \mid s \text{ is in } L \text{ and } t \text{ is in } M\}$
- The Kleene Closure of a language L is written as  $L^* = \text{Zero or more occurrence of language } L$ .

 Notations If r and s are regular expressions denoting the languages  $L(r)$  and  $L(s)$ , then
 

- Union :  $(r) \mid (s)$  is a regular expression denoting  $L(r) \cup L(s)$
- Concatenation :  $(r)(s)$  is a regular expression denoting  $L(r)L(s)$
- Kleene closure :  $(r)^*$  is a regular expression denoting  $(L(r))^*$
- $(r)$  is a regular expression denoting  $L(r)$

 Precedence and Associativity • \*, concatenation (.), and | (pipe sign) are left associative
 

- \* has the highest precedence
- Concatenation (.) has the second highest precedence.
- | (pipe sign) has the lowest precedence of all.

 Representing valid tokens of a language in regular expression If x is a regular expression, then:
 

- $x^*$  means zero or more occurrence of x. i.e., it can generate { e, x, xx, xxx, xxxx, ... }
- $x^+$  means one or more occurrence of x. i.e., it can generate { x, xx, xxx, xxxx ... } or  $x.x^*$
- $x?$  means at most one occurrence of x i.e., it can generate either {x} or {e}.

 [a-z] is all lower-case alphabets of English language. [A-Z] is all upper-case alphabets of English language. [0-9] is all natural digits used in mathematics. Representing occurrence of symbols using regular expressions Letter = [a – z] or [A – Z] Digit = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 or [0-9] Sign = [+ | -] Representing language tokens using regular expressions Decimal = (sign)? (digit)+ Identifier = (letter)(letter | digit)\* The only problem left with the lexical analyzer is how to verify the validity of a regular expression used in specifying the patterns of keywords of a language. A well-accepted solution is to use finite automata for verification. Compiler-construction tools Originally, compilers were written “from scratch”, but now the situation is quite different. A number of tools are available to ease the burden. We will study tools that generate scanners and parsers. This will involve us in some theory, regular expressions for scanners and various grammars for parsers. These techniques are fairly successful. One drawback can be that they do not execute as fast as “hand-crafted” scanners and parsers. We will also see tools for syntax-directed translation and automatic code generation. The automation in these cases is not as complete. Finally, there is the large area of optimization. This is not automated; however, a basic component of optimization is “data-flow analysis” (how values are transmitted between parts of a program) and there are tools to help with this task.

**5.4.LEX** Lex is a tool in lexical analysis phase to recognize tokens using regular expression. Lex tool itself is a lex compiler. Figure 1.9 LEX Diagram
 

- lex.l is an input file written in a language which describes the generation of lexical analyzer. The lex compiler transforms lex.l to a C program known as lex.yy.c.
- lex.yy.c is compiled by the C compiler to a file called a.out.
- The output of C compiler is the working lexical analyzer which takes stream of input characters and produces a stream of tokens.
- yyval is a global variable which is shared by lexical analyzer and parser to return the name and an attribute value of token.
- The attribute value can be numeric code, pointer to symbol table or nothing.
- Another tool for lexical analyzer generation is Flex.

 Structure of Lex Programs Lex program will be in following form declarations %% translation rules %% auxiliary functions Declarations This section includes declaration of variables, constants and regular definitions. Translation rules It contains regular expressions and code segments. Form : Pattern {Action} Pattern is a regular expression or regular definition. Action refers to segments of code. Auxiliary functions This section holds additional functions which are used in actions. These functions are compiled separately and loaded with lexical analyzer. Lexical analyzer produced by Lex starts its process by reading one character at a time until a valid match for a pattern is found. Once a match is found, the associated action takes place to produce token. The token is then given to parser for further processing. Conflict Resolution in Lex Conflict arises when several prefixes of input matches one or more patterns. This can be resolved by the following:
 

- Always prefer a longer prefix than a shorter prefix.
- If two or more patterns are matched for the longest prefix, then the first pattern listed in lex program is preferred.

 Lookahead Operator • Lookahead operator is the additional operator that is read by lex in order to distinguish additional pattern for a token.
 

- Lexical analyzer is used to read one character ahead of valid lexeme and then retracts to produce token.
- At times, it is needed to have certain characters at the end of input to match with a pattern. In such cases, slash

(/) is used to indicate end of part of pattern that matches the lexeme. (eg.) In some languages keywords are not reserved. So the statements IF (I, J) = 5 and IF(condition) THEN results in conflict whether to produce IF as an array name or a keyword. To resolve this the lex rule for keyword IF can be written as, IF/\(.\*)\{letter\} Design of Lexical Analyzer • Lexical analyzer can either be generated by NFA or by DFA. • DFA is preferable in the implementation of lex. Structure of Generated Analyzer Architecture of lexical analyzer generated by lex is given in Figure: Lexical analyzer program includes:

- Program to simulate automata.
- Components created from lex program by lex itself which are listed as follows:
  - o A transition table for automaton.
  - o Functions that are passed directly through lex to the output.
  - o Actions from input program (fragments of code) which are invoked by automaton simulator when needed.

Unit-II Syntax Analysis & Syntax Directed Translation Syntax Analysis: CFGs, Top Down Parsing, Brute Force Approach, Recursive Descent Parsing, Transformation on the Grammars, Predictive Parsing, Bottom up Parsing, Operator Precedence Parsing, LR parsers (SLR,LALR, LR),Parser Generation. Syntax directed Definitions: Construction of Syntax tree, Bottom up Evaluation of S-attributed Definition, Lattribute Definition, Top Down Translation, Bottom Up Evaluation of Inherited Attributes, Recursive Evaluation, Analysis of Syntax Directed Definition. 1 Introduction of Syntax Analysis The second stage of translation is called Syntax analysis or parsing. In this phase expressions, statements, declarations etc... are identified by using the results of lexical analysis. Syntax analysis is aided by using techniques based on formal grammar of the programming language. Where lexical analysis splits the input into tokens, the purpose of syntax analysis (also known as parsing) is to recombine these tokens. Not back into a list of characters, but into something that reflects the structure of the text. This "something" is typically a data structure called the syntax tree of the text. As the name indicates, this is a tree structure. The leaves of this tree are the tokens found by the lexical analysis, and if the leaves are read from left to right, the sequence is the same as in the input text. Hence, what is important in the syntax tree is how these leaves are combined to form the structure of the tree and how the interior nodes of the tree are labeled. In addition to finding the structure of the input text, the syntax analysis must also reject invalid texts by reporting syntax errors. As syntax analysis is less local in nature than lexical analysis, more advanced methods are required. We, however, use the same basic strategy: A notation suitable for human understanding is transformed into a machine-like low-level notation suitable for efficient execution. This process is called parser generation. A syntax analyzer or parser takes the input from a lexical analyzer in the form of token streams. The parser analyzes the source code (token stream) against the production rules to detect any errors in the code. The output of this phase is a parse tree. This way, the parser accomplishes two tasks, i.e., parsing the code, looking for errors and generating a parse tree as the output of the phase.

### 1.1 Context Free Grammar Definition – A context-free grammar (CFG) consisting of a finite set of grammar rules is a quadruple $(N, T, P, S)$ , where $N$ is a set of non-terminal symbols. $T$ is a set of terminals where $N \cap T = \emptyset$ . $P$ is a set of rules, $P: N \rightarrow (N \cup T)^*$ , i.e., the left-hand side of the production rule $P$ does not have any right context or left context. $S$ is the start symbol.

Example The grammar  $\{A\}, \{a, b, c\}, P, A$ ,  $P : A \rightarrow aA, A \rightarrow abc$ . The grammar  $\{S\}, \{a, b\}, \{a, b\}, P, S$ ,  $P: S \rightarrow aSa, S \rightarrow bSb, S \rightarrow \epsilon$

### Generation of Derivation Tree

A derivation tree or parse tree is an ordered rooted tree that graphically represents the semantic information a string derived from a context-free grammar. Representation Technique Root vertex – Must be labeled by the start symbol. Vertex – Labeled by a non-terminal symbol. Leaves – Labeled by a terminal symbol or  $\epsilon$ . If  $S \rightarrow x_1x_2 \dots x_n$  is a production rule in a CFG, then the parse tree / derivation tree will be as follows – Figure 2.1: Derivation tree There are two different approaches to draw a derivation tree – Top-down Approach – Starts with the starting symbol  $S$  Goes down to tree leaves using productions Bottom-up Approach – Starts from tree leaves Proceeds upward to the root which is the starting symbol  $S$

### Leftmost and Rightmost Derivation of a String

**Leftmost derivation** – A leftmost derivation is obtained by applying production to the leftmost variable in each step.

**Rightmost derivation** – A rightmost derivation is obtained by applying production to the rightmost variable in each step.

Example Let any set of production rules in a CFG be  $X \rightarrow X+X \mid X*X \mid X \mid a$  over an alphabet  $\{a\}$ . The leftmost derivation for the string "a+a\*a" may be –  $X \rightarrow X+X \rightarrow a+X \rightarrow a + X*X \rightarrow a+a*X \rightarrow a+a*a$

The stepwise derivation of the above string is shown as below – The rightmost derivation for the above string "a+a\*a" may be –  $X \rightarrow X*X \rightarrow X*a \rightarrow X+X*a \rightarrow a+a*a$

The stepwise derivation of the above string is shown as below – 2. Parsing Syntax analyzers follow production rules defined by means of context-free grammar. The way the production rules are implemented (derivation) divides parsing into two types: top-down parsing and bottom-up parsing.

### 2.1 Top-Down Parsing

S X1 X2 Xn A program that performs syntax analysis is called a parser. A syntax analyzer takes tokens as input and output error message if the program syntax is wrong. The parser uses symbol-look-ahead and an approach called topdown parsing without backtracking. Top-down parsers check to see if a string can be generated by a grammar by

creating a parse tree starting from the initial symbol and working down. Bottom-up parsers, however, check to see a string can be generated from a grammar by creating a parse tree from the leaves, and working up. Early parser generators such as YACC creates bottom-up parsers whereas many of Java parser generators such as JavaCC create top-down parsers. Figure 2.2:Top down parsing 2.2 Brute-Force Approach A top-down parse moves from the goal symbol to a string of terminal symbols. In the terminology of trees, this is moving from the root of the tree to a set of the leaves in the syntax tree for a program. In using full backup we are willing to attempt to create a syntax tree by following branches until the correct set of terminals is reached. In the worst possible case, that of trying to parse a string which is not in the language, all possible combinations are attempted before the failure to parse is recognized. Top-down parsing with full backup is a "brute-force" method of parsing. In general terms, this method operates as follows: 1. Given a particular non-terminal that is to be expanded, the first production for this non-terminal is applied. 2. Then, within this newly expanded string, the next (leftmost) non-terminal is selected for expansion and its first production is applied. 3. This process (step 2) of applying productions is repeated for all subsequent non-terminals that are selected until such time as the process cannot or should not be continued. This termination (if it ever occurs) may be due to two causes. First, no more non-terminals may be present, in which case the string has been successfully parsed. Second, it may result from an incorrect expansion which would be indicated by the production of a substring of terminals which does not match the appropriate segment of the source string. In the case of such an incorrect expansion, the process is "backed up" by undoing the most recently applied production. Instead of using the particular expansion that caused the error, the next production of this non-terminal is used as the next expansion, and then the process of production application continues as before. If, on the other hand, no further productions are available to replace the production that caused the error, this error-causing expansion is replaced by the non-terminal itself, and the process is backed up again to undo the next most recently applied production. This backing up continues either until we are able to resume normal application of productions to selected non-terminals or until we have backed up to the goal symbol and there are no further productions to be tried. In the latter case, the given string must be unparseable because it is not part of the language determined by this particular grammar. As an example of this brute-force parsing technique, let us consider the simple grammar Recursive Descent Top Down Parsing Non tracking parser Back tracking S->aAd/aB A->b/c B->ccd/ddc Where S is the goal or start symbol.

Figure 6-1 illustrates the working of this brute-force parsing technique by showing the sequence of syntax trees generated during the parse of the string 'accd'. 2.3 Recursive Descent Parsing Typically, top-down parsers are implemented as a set of recursive functions that descent through a parse tree for a string. This approach is known as recursive descent parsing, also known as LL(k) parsing where the first L stands for left-to-right, the second L stands for leftmost-derivation, and k indicates k-symbol look ahead. Therefore, a parser using the single symbol look-ahead method and top-down parsing without backtracking is called LL(1) parser. In the following sections, we will also use an extended BNF notation in which some regulation expression operators are to be incorporated. A syntax expression defines sentences of the form, or. A syntax of the form defines sentences that consist of a sentence of the form followed by a sentence of the form followed by a sentence of the form. A syntax of the form defines zero or one occurrence of the form. A syntax of the form defines zero or more occurrences of the form. A usual implementation of an LL(1) parser is.

- initialize its data structures,
- get the lookahead token by calling scanner routines, and
- call the routine that implements the start symbol.

Here is an example.

```
proc syntax Analysis() begin
    initialize(); // initialize global data and structures
    nextToken(); // get the lookahead token
    program(); // parser routine that implements the start symbol
end;
```

Example for backtracking: Consider the grammar  $G : S \rightarrow cAd \quad A \rightarrow ab \mid a$  And the input string  $w=cad$ . The parse tree can be constructed using the following top-down approach:

- Step1: Initially create a tree with single node labeled S. An input pointer points to 'c', the first symbol of w.
- Expand the tree with the production of S.
- Step2: The leftmost leaf 'c' matches the first symbol of w, so advance the input pointer to the second symbol of w 'a' and consider the next leaf 'A'.
- Expand A using the first alternative.
- Step3: The second symbol 'a' of w also matches with second leaf of tree. So advance the input pointer to third symbol of w 'd'.
- But the third leaf of tree is b which does not match with the input symbol d. Hence discard the chosen production and reset the pointer to second position. This is called backtracking.
- Step4: Now try the second alternative for A. Now we can halt and announce the successful completion of parsing.

Example for recursive decent parsing: A left-recursive grammar can cause a recursive-descent parser to go into an infinite loop. Hence, elimination of left-recursion must be done before parsing.

2.4 Predictive Parsing Predictive parsing is a special case of recursive descent parsing where no backtracking is required. The key problem of predictive parsing is to determine the production to be applied for a nonterminal in case of

alternatives. Non-recursive predictive parser INPUT OUTPUT Stack Figure 2.3: Predictive parsing The table-driven predictive parser has an input buffer, stack, a parsing table and an outputstream. Input buffer: It consists of strings to be parsed, followed by \$ to indicate the end of the input string. Stack: It contains a sequence of grammar symbols preceded by \$ to indicate the bottom of the stack. Initially, the stack contains the start symbol on top of \$. Parsing table: It is a two-dimensional array M [A, a], where 'A' is a non-terminal and 'a' is a terminal. Predictive parsing program. The parser is controlled by a program that considers X, the symbol on top of stack, and a, thecurrent input symbol. These two symbols determine the parser action. There are three possibilities:
 

- If  $X = a = \$$ , the parser halts and announces successful completion of parsing.
- If  $X = a \neq \$$ , the parser pops X off the stack and advances the input pointer to the next input symbol.
- If X is a non-terminal, the program consults entry  $M[X, a]$  of the parsing table M. This entry will either be an X-production of the grammar or an error entry.

 • If  $M[X, a] = \{X \rightarrow UVW\}$ ,the parser replaces X on top of the stack by UVW
 • If  $M[X, a] = \text{error}$ , the parser calls an error recovery routine. Predictive parsing table construction: The construction of a predictive parser is aided by two functions associated with a grammar
  $G : a + b \$ X Y Z \$$ 
**Predictive Parsing Program**  
**Predictive Table M**

1. FIRST Rules for first ( ): If X is terminal, then  $\text{FIRST}(X) = \{X\}$ . 2. If  $X \rightarrow \epsilon$  is a production, then add  $\epsilon$  to  $\text{FIRST}(X)$ .
3. If X is non-terminal and  $X \rightarrow a\alpha$  is a production then add a to  $\text{FIRST}(X)$ .
4. If X is non-terminal and  $X \rightarrow Y_1 Y_2 \dots Y_k$  is a production, then place a in  $\text{FIRST}(X)$  if forsome i, a is in  $\text{FIRST}(Y_i)$ , and  $\epsilon$  is in all of  $\text{FIRST}(Y_1), \dots, \text{FIRST}(Y_{i-1})$ ; that is,  $Y_1, \dots, Y_{i-1} \Rightarrow \epsilon$ . If  $\epsilon$  is in  $\text{FIRST}(Y_j)$  for all  $j=1,2,\dots,k$ , then add  $\epsilon$  to  $\text{FIRST}(X)$ .

**Rules for follow ( ):**

1. If S is a start symbol, then  $\text{FOLLOW}(S)$  contains \$.
2. If there is a production  $A \rightarrow \alpha B \beta$ , then everything in  $\text{FIRST}(\beta)$  except  $\epsilon$  is placed infollow (B).
3. If there is a production  $A \rightarrow \alpha B$ , or a production  $A \rightarrow \alpha B \beta$  where  $\text{FIRST}(\beta)$  contains  $\epsilon$ , then everything in  $\text{FOLLOW}(A)$  is in  $\text{FOLLOW}(B)$ .

**Example:** Consider the following grammar:  $E \rightarrow E+T \mid TT \rightarrow T^*F \mid F F \rightarrow (E) \mid \text{id}$  After eliminating left-recursion the grammar is  $E \rightarrow TE' \mid E' \rightarrow +TE' \mid \epsilon T \rightarrow FT' \mid T' \rightarrow *FT' \mid \epsilon F \rightarrow (E) \mid \text{id}$ 
**First ( ):**  $\text{FIRST}(E) = \{ , \text{id}\}$   
 $\text{FIRST}(E') = \{ +, \epsilon \}$   
 $\text{FIRST}(T) = \{ , \text{id} \}$   
 $\text{FIRST}(T') = \{ *, \epsilon \}$   
 $\text{FIRST}(F) = \{ , \text{id} \}$   
**Follow ( ):**  $\text{FOLLOW}(E) = \{ \$, \}$   
 $\text{FOLLOW}(E') = \{ +, \$, \}$   
 $\text{FOLLOW}(T) = \{ +, \$, \}$   
 $\text{FOLLOW}(T') = \{ +, \$, \}$   
 $\text{FOLLOW}(F) = \{ +, *, \$, \}$   
**Non terminal Id**  
 $+ * ( ) \$$   
 $E \rightarrow TE' \mid E' \rightarrow TE' \mid E' \rightarrow +TE' \mid E' \rightarrow \epsilon E' \rightarrow \epsilon T T \rightarrow FT' \mid T' \rightarrow \epsilon T \rightarrow \epsilon T \rightarrow \epsilon F F \rightarrow id$   
**F**  
 $F \rightarrow (E)$ 
**Table 2.1: Predictive parsing table**

- Bottom-Up Parsing Constructing a parse tree for an input string beginning at the leaves and going towards theroot is called bottom-up parsing. A general type of bottom-up parser is a shift-reduce parser.
- 3.1 Shift-Reduce Parsing Shift-reduce parsing uses two unique steps for bottom-up parsing. These steps are known as shift-step and reduce-step.

- Shift step: The shift step refers to the advancement of the input pointer to the next input symbol, which is called the shifted symbol. This symbol is pushed onto the stack. The shifted symbol is treated as a single node of the parse tree.
- Reduce step: When the parser finds a complete grammar rule (RHS) and replaces it to (LHS), it is known as reduce-step. This occurs when the top of the stack contains a handle. To reduce, a POP function is performed on the stack which pops off the handle and replaces it with LHS non-terminal symbol.

**Example:** Consider the grammar:  $S \rightarrow aABe \mid A \rightarrow Abc \mid b B \rightarrow d$  The sentence to be recognized is abbcde
 **REDUCTION (LEFT MOST) RIGHTMOST DERIVATION**
 $abbcde \rightarrow aABe \rightarrow abbcde$ 
**Table 2.2: Shift Reduce Parser**

- 3.2 Operator Precedence Parsing Bottom-up parsers for a large class of context-free grammars can be easily developed using operator grammars. Operator grammars have the property that no production right side is empty or has two adjacent non-terminals. This property enables the implementation of efficient operator-precedence parsers. These parser rely on the following three precedence relations:

Relation Meaning

- a b a takes precedence over b
- These operator precedence relations allow to delimit the handles in the right sentential forms: marks the right end.

Example: The input string:  $id1 + id2 * id3$  After inserting precedence relations becomes  $\$ + * \$$

Having precedence relations allows to identify handles as follows: Scan the string from left until seeing  $\rightarrow$  Scan backwards the string from right to left until seeing forms the handle  $Id + * \$$

 $Id . > . > . > + * . > . > \$$ 
**Table 2.3: Operator precedence parsing**

- 3.3 LR Parsing Introduction The "L" is for left-to-right scanning of the input and the "R" is for constructing a rightmost derivation in reverse.
- Why LR Parsing: LR parsers can be constructed to recognize virtually all programming-language constructs for which contextfree grammars can be written. The LR parsing method is the most general non-backtracking shift-reduce parsing method known, yet it can be implemented as efficiently as other shift-reduce methods. The class of grammars that can be parsed using LR methods is a proper subset of the class of grammars that can be parsed with predictive parsers. An LR parser can detect a syntactic error as soon as it is possible to do so on a left-to-right scan of the input.
- Figure 2.4: LR Parser The disadvantage is that it takes too much work to construct an LR parser by hand for a typical programminglanguage grammar. But there are lots of LR parser generators available to make this task easy. There are three widely used algorithms available for constructing an LR parser: SLR (1) –

Simple LR Parser: • Works on smallest class of grammar • Few number of states, hence very small table • Simple and fast construction LR (1) – LR Parser: • Works on complete set of LR(1) Grammar • Generates large table and large number of states • Slow construction LALR (1) – Look-Ahead LR Parser: • Works on intermediate size of grammar • Number of states are same as in SLR(1) Input Action goto Parser Stack Output 3.4 SLR (1) – Simple LR Parser: Shift-reduce parsing attempts to construct a parse tree for an input string beginning at the leaves and working up towards the root. In other words, it is a process of “reducing” (opposite of deriving a symbol using a production rule) a string w to the start symbol of a grammar. At every (reduction) step, a particular substring matching the RHS of a production rule is replaced by the symbol on the LHS of the production. A general form of shift-reduce parsing is LR (scanning from Left to right and using Right-most derivation in reverse) parsing, which is used in a number of automatic parser generators like Yacc, Bison, etc. A convenient way to implement a shift-reduce parser is to use a stack to hold grammar symbols and an input buffer to hold the string w to be parsed. The symbol \$ is used to mark the bottom of the stack and also the right-end of the input.

Notation ally, the top of the stack is identified through a separator symbol |, and the input string to be parsed appears on the right side of |. The stack content appears on the left of |. For example, an intermediate stage of parsing can be shown as follows: \$id1 | + id2 \* id3\$ .... (1) Here “\$id1” is in the stack, while the input yet to be seen is “+ id2 \* id3\$\*. In shift-reduce parser, there are two fundamental operations: shift and reduce. Shift operation: The next input symbol is shifted onto the top of the stack. After shifting + into the stack, the above state captured in (1) would change into: \$id1 + | id2 \* id3\$ Reduce operation: Replaces a set of grammar symbols on the top of the stack with the LHS of a production rule. After reducing id1 using  $E \rightarrow id$ , the state (1) would change into: \$E | + id2 \* id3\$ In every example, we introduce a new start symbol ( $S'$ ), and define a new production from this new start symbol to the original start symbol of the grammar. Consider the following grammar (putting an explicit end-marker \$ at the end of the first production): (1)  $S' \rightarrow S\$$  (2)  $S \rightarrow Sa$  (3)  $S \rightarrow b$

For this example, the NFA for the stack can be shown as follows: Figure 2.5: Shift Operation After doing  $\epsilon$ -closure, the resulting DFA is as follows:  $S' \rightarrow .S$   $S \rightarrow .Sa$   $S \rightarrow .S$   $S \rightarrow .b$   $S \rightarrow .b$ .  $S \rightarrow .S.a$   $S \rightarrow .Sa$   $S \rightarrow .S.b$   $S \rightarrow .S\$$   $S \rightarrow .Sa$   $S \rightarrow .b$   $S \rightarrow .S.a$   $S \rightarrow .S\$$   $S \rightarrow .Sa$   $S \rightarrow .b$ . a Figure 2.6: Reduce Operation The states of DFA are also called “Canonical Collection of Items”. Using the above notation, the ACTION-GOTO table can be shown as follows: State A B \$ s 1 S3 2 S4, r1 R1 R1 3 R3 R3 R3 4 R2 R2

R2 Table 2.4: Simple LR Parser 3.5. Canonical LR Parsing • Carry extra information in the state so that wrong reductions by  $A \alpha$  will be ruled out. • Redefine LR items to include a terminal symbol as a second component (look ahead symbol). • The general form of the item becomes  $[A \alpha . T, a]$  which is called LR(1) item. • Item  $[A \alpha . , a]$  calls for reduction only if next input is a. The set of symbols Canonical LR parsers solve this problem by storing extra information in the state itself. The problem we have with SLR parsers is because it does reduction even for those symbols of follow (A) for which it is invalid. So LR items are redefined to store 1 terminal (look ahead symbol) along with state and thus, the items now are LR(1) items. An LR(1) item has the form :  $[A \alpha . T, a]$  and reduction is done using this rule only if input is 'a'. Clearly the symbols a's form a subset of follow (A). Closure (I) repeat for each item  $[A \alpha . B T, a]$  in I for each production  $B \gamma$  in  $G'$  and for each terminal b in First( Ta) add item  $[B . \gamma , b]$  to I until no more additions to I To find closure for Canonical LR parsers: Repeat for each item  $[A \alpha . B T, a]$  in I for each production  $B \gamma$  in  $G'$  and for each terminal b in First( Ta) add item  $[B . \gamma , b]$  to I until no more items can be added to I Example Consider the following grammar  $S' S S CC C cC | d$  Compute closure (I) where  $I = \{[S' . S, \$] S' . S, \$ S . CC, \$ C . CC, c C . CC, d C . d, c C . d, d\}$  For the given grammar:  $S' S S CC C cC | d$  I : closure( $[S' . S, \$]$ )  $S' . S$  \$ as first( e \$) =  $\{\$$   $\}$   $S . CC$  \$ as first(C\$) = first(C) = {c, d}  $C . CC$  c as first(Cc) = first(C) = {c, d}  $C . CC$  d as first(Cd) = first(C) = {c, d}  $C . CC$  as first( e c) = {c}  $C . d$  d as first( e d) = {d} Table 2.5: Canonical LR Parsing Example Construct sets of LR(1) items for the grammar on previous slide I0 :  $S' . S, \$ S . CC, \$ C . CC, c / d C . d, c / d$  I1 : goto(I0, S)  $S' . S, \$$  I2 : goto(I0, C)  $S . CC, \$ C . CC, \$ C . d$ , \$ I3 : goto(I0, C)  $C . CC, c / d C . CC, c / d C . d, c / d$  I4 : goto(I0, d)  $C . d, c / d$  I5 : goto(I2, C)  $S . CC, \$$  I6 : goto(I2, C)  $C . CC, \$ C . d, \$$  I7 : goto(I2, d)  $C . d, \$$  I8 : goto(I3, C)  $C . CC, c / d$  I9 : goto(I6, C)  $C . CC, \$$  To construct sets of LR (1) items for the grammar given in previous slide we will begin by computing closure of  $\{[S' . S, \$]\}$ . To compute closure we use the function given previously. In this case  $\alpha = \epsilon$ ,  $B = S$ ,  $\beta = \epsilon$  and  $a = \$$ . So add item  $[S . CC, \$]$ . Now first(C\$) contains c and d so we add following items We have  $A = S$ ,  $\alpha = \epsilon$ ,  $B = C$ ,  $\beta = C$  and  $a = \$$  Now first(C\$) = first(C) contains c and d So we add the items  $[C . CC, c]$ ,  $[C . CC, d]$ ,  $[C . dC, c]$ ,  $[C . dC, d]$ . Similarly we use this function and construct all sets of LR (1) items.

Construction of Canonical LR parse table Construct  $C = \{I0, \dots, I_n\}$  the sets of LR(1) items. If  $[A \alpha . T, b]$  is in  $I_i$  and goto (Ii, a)=Ij then action[i,a]=shift j If  $[A \alpha . , a]$  is in  $I_i$  then action[i,a] reduce A  $\alpha$  If  $[S' . S, \$]$  is in  $I_i$  then action [i,\$] = accept If goto (Ii, A) = Ij then goto[i,A] = j for all non We are representing shift j as sj and reduction by rule number j as rj. Note that entries corresponding to [state, terminal] are related to action table and [state, non-terminal] related to goto table. We

have  $[1, \$]$  as accept because  $[S' S., \$] \in I_1$ . Parse table state  $i + * ( ) \$ E T F 0 s5 s4 1 2 3 1 s6 acc 2 r2 s7 r2 r2 3 r4 r4 r4 4 s5 s4 8 2 3 5 r6 r6 r6 6 s5 s4 9 3 7 s5 s4 10 8 s6 s11 9 r1 s7 r1 r1 10 r3 r3 r3 11 r5 r5 r5$  Table 2.6: Parser Table  
 We are representing shift  $j$  as  $s_j$  and reduction by rule number  $j$  as  $r_j$ . Note that entries corresponding to  $[state, terminal]$  are related to action table and  $[state, non-terminal]$  related to goto table. We have  $[1, \$]$  as accept because  $[S' S., \$] \in I_1$   
 LALR Parse table Look Ahead LR parsers Consider a pair of similar looking states (same kernel and different lookahead) in the set of LR(1) items  $I_4 : C d., c/d I_7 : C d., \$$  Replace  $I_4$  and  $I_7$  by a new state  $I_{47}$  consisting of  $(C d., c/d/\$)$  Similarly  $I_3 & I_6$  and  $I_8 & I_9$  form pairs Merge LR(1) items having the same core We will combine  $I_i$  and  $I_j$  to construct new  $I_{ij}$  if  $I_i$  and  $I_j$  have the same core and the difference is only in look ahead symbols. After merging the sets of LR(1) items for previous example will be as follows:  $I_0 : S' S \$ S .CC \$ C .cC c/d C .d c/d I_1 : goto(I_0 , S) S' S. \$ I_2 : goto(I_0 , C) S C.C \$ C .cC \$ C .d \$ I_36 : goto(I_2 , c) C C.C c/d/\$ C .cC c/d/\$ C .d c/d/\$ I_4 : goto(I_0 , d) C d. c/d I_5 : goto(I_2 , C) S CC. \$ I_7 : goto(I_2 , d) C d. \$ I_89 : goto(I_36 , C) C cC. c/d/\$$  Construct LALR parse table Construct  $C = \{I_0, \dots, I_n\}$  set of LR(1) items For each core present in LR(1) items find all sets having the same core and replace these sets by their union Let  $C' = \{J_0, \dots, J_m\}$  be the resulting set of items Construct action table as was done earlier Let  $J = I_1 U I_2 \dots U I_k$  since  $I_1, I_2 \dots, I_k$  have same core,  $goto(J, X)$  will have the same core Let  $K = goto(I_1, X) U goto(I_2, X) \dots goto(I_k, X)$  the  $goto(J, X) = K$  The construction rules for LALR parse table are similar to construction of LR(1) parse table. LALR parse table state  $i + * ( ) \$ E T F 0 s5 s4 1 2 3 1 s6 acc 2 r2 s7 r2 r2 3,6 r4 r4 r4 r4 9 3 4,7 s5,7 s4,7 8 2 3,10 5,8 s5 r6 r6 s4,8 r6 r6 9,10 r1 s7 r1 r1 10 r3 r3 r3 11 r5 r5 r5$  Table 2.7: LALR parse table. The construction rules for LALR parse table are similar to construction of LR(1) parse table. Notes on LALR parse table Modified parser behaves as original except that it will reduce  $C d$  on inputs like  $ccd$ . The error will eventually be caught before any more symbols are shifted. In general core is a set of LR(0) items and LR(1) grammar may produce more than one set of items with the same core. 4 Parser Generators Some common parser generators YACC: Yet Another Compiler Compiler Bison: GNU Software ANTLR: AN other Tool for Language Recognition Yacc/Bison source program specification (accept LALR grammars) declaration %% translation rules %% supporting C routines Yacc and Lex schema YACC Manual Figure 2.7: YACC 5. Syntax Directed Definition Specifies the values of attributes by associating semantic rules with the grammar productions. It is a context free grammar with attributes and rules together which are associated with grammar symbols and productions respectively. The process of syntax directed translation is two-fold:  

- Construction of syntax tree and
- Computing values of attributes at each node by visiting the nodes of syntax tree. Semantic actions Semantic actions are fragments of code which are embedded within production bodies by syntax directed translation. They are usually enclosed within curly braces  $\{\}$ . It can occur anywhere in a production but usually at the end of production.  $E \rightarrow E_1 + T \{ \text{print } '+' \}$  yacc source  $y.tab.c$  a.out "lex.yy.c" lex.yy.c lex source lex cc Types of translation

- L-attributed translation It performs translation during parsing itself. No need of explicit tree construction. L represents 'left to right'.
- S-attributed translation It is performed in connection with bottom up parsing. 'S' represents synthesized. Types of attributes
- Inherited attributes It is defined by the semantic rule associated with the production at the parent of node. Attributes values are confined to the parent of node, its siblings and by itself. The non-terminal concerned must be in the body of the production.
- Synthesized attributes It is defined by the semantic rule associated with the production at the node. Attributes values are confined to the children of node and by itself. The non-terminal concerned must be in the head of production. Terminals have synthesized attributes which are the lexical values (denoted by lex val) generated by the lexical analyzer. Syntax directed definition of simple desk calculator Production Semantic rules  $L \rightarrow E$   $L.val = E.val$   $E \rightarrow E_1 + T$   $E.val = E_1.val + T.val$   $E \rightarrow T$   $E.val = T.val$   $T \rightarrow T_1 * F$   $T.val = T_1.val * F.val$   $T \rightarrow F$   $T.val = F.val$   $F \rightarrow (E)$   $F.val = E.val$   $F \rightarrow \text{digit}$   $F.val = \text{digit.lexval}$  Table 2.8: Syntax directed Syntax-directed definition-inherited attributes Production Semantic Rules  $D \rightarrow TL$   $L.inh = T.type$   $T \rightarrow \text{int}$   $T.type = \text{integer}$   $T \rightarrow \text{float}$   $T.type = \text{float}$   $L \rightarrow L_1$ ,  $\text{id}$   $L_1.inh = L.inh$  addType  $(\text{id.entry}, L.inh)$   $L \rightarrow \text{id}$  addType  $(\text{id.entry}, L.inh)$  Table 2.9: Syntax directed
  - Symbol  $T$  is associated with a synthesized attribute type.
  - Symbol  $L$  is associated with an inherited attribute  $inh$ , Types of Syntax Directed Definitions 5.1 S-Attributed Definitions Syntax directed definition that involves only synthesized attributes is called S-attributed. Attribute values for the non-terminal at the head is computed from the attribute values of the symbols at the body of the production. The attributes of a S-attributed SDD can be evaluated in bottom up order of nodes of the parse tree. i.e., by performing post order traversal of the parse tree and evaluating the attributes at a node when the traversal leaves that node for the last time. Production Semantic rules  $L \rightarrow E$   $L.val = E.val$   $E \rightarrow E_1 + T$   $E.val = E_1.val + T.val$   $E \rightarrow T$   $E.val = T.val$   $T \rightarrow T_1 * F$   $T.val = T_1.val * F.val$   $T \rightarrow F$   $T.val = F.val$   $F \rightarrow (E)$   $F.val = E.val$   $F \rightarrow \text{digit}$   $F.val = \text{digit.lexval}$  Table 2.10: S-attributed Definitions L-Attributed Definitions The syntax directed definition in which the edges of dependency graph for the attributes in

production body, can go from left to right and not from right to left is called L-attributed definitions. Attributes of L-attributed definitions may either be synthesized or inherited. If the attributes are inherited, it must be computed from:

- Inherited attribute associated with the production head.
- Either by inherited or synthesized attribute associated with the production located to the left of the attribute which is being computed.
- Either by inherited or synthesized attribute associated with the attribute under consideration in such a way that no cycles can be formed by it in dependency graph.

Production Semantic Rules  $T \rightarrow FT' T'.inh = F.val$   $T \rightarrow *FT1' T'1.inh = T'.inh \times F.val$

Table 2.11: L-attributed Definitions In production 1, the inherited attribute  $T'$  is computed from the value of  $F$  which is to its left. In production 2, the inherited attributed  $T1'$  is computed from  $T'.inh$  associated with its head and the value of  $F$  which appears to its left in the production. i.e., for computing inherited attribute it must either use from the above or from the left information of SDD

Construction of Syntax Trees SDDs are useful for construction of syntax trees. A syntax tree is a condensed form of parse tree. Figure 2.8 : Syntax Trees

- Syntax trees are useful for representing programming language constructs like expressions and statements.
- They help compiler design by decoupling parsing from translation.
- Each node of a syntax tree represents a construct; the children of the node represent the meaningful components of the construct.
- e.g. a syntax-tree node representing an expression  $E1 + E2$  has label  $+$  and two children representing the sub expressions  $E1$  and  $E2$
- Each node is implemented by objects with suitable number of fields; each object will have an  $op$  field that is the label of the node with additional fields as follows: If the node is a leaf, an additional field holds the lexical value for the leaf. This is created by function  $Leaf(op, val)$  If the node is an interior node, there are as many fields as the node has children in the syntax tree. This is created by function  $Node(op, c1, c2, \dots, ck)$ . Example: The S-attributed definition in figure below constructs syntax trees for a simple expression grammar involving only the binary operators  $+$  and  $-$ . As usual, these operators are at the same precedence level and are jointly left associative. All non-terminals have one synthesized attribute node, which represents a node of the syntax tree.

S.no Production Semantic Rules

- 1  $E \rightarrow E1+E$   $E.node = \text{new Node}('+', E1.node, T.node)$
- 2  $E \rightarrow E1-T$   $E.node = \text{new Node}('-', E1.node, T.node)$
- 3  $E \rightarrow T$   $E.node = T.node$
- 4  $E \rightarrow (E)$   $E.node = T.node$
- 5  $T \rightarrow id$   $T.node = \text{new leaf}(id, id.entry)$
- 6  $T \rightarrow num$   $T.node = \text{new Leaf}(num, num.val)$

Syntax tree for a-4+c using the above SDD is shown below.

Figure 2.9:L-Attribute 5.2 Bottom-Up Evaluation Of SDT Given an SDT, we can evaluate attributes even during bottom-up parsing. To carry out the semantic actions, parser stack is extended with semantic stack. The set of actions performed on semantic stack are mirror reflections of parser stack. Maintaining semantic stack is very easy. During shift action, the parser pushes grammar symbols on the parser stack, whereas attributes are pushed on to semantic stack. During reduce action, parser reduces handle, whereas in semantic stack, attributes are evaluated by the corresponding semantic action and are replaced by the result. For example, consider the SDT  $A \rightarrow X Y Z \{A \cdot a := f(X \cdot x, Y \cdot y, Z \cdot z)\}$  Strictly speaking, attributes are evaluated as follows  $A \rightarrow X Y Z \{val[ntop] := f(val[top - 2], val[top - 1], val[top])\}$

Evaluation of Synthesized Attributes

- Whenever a token is shifted onto the stack, then it is shifted along with its attribute value placed in  $val[top]$ .
- Just before a reduction takes place the semantic rules are executed.
- If there is a synthesized attribute with the left-hand side non-terminal, then carrying out semantic rules will place the value of the synthesized attribute in  $val[ntop]$ . Let us understand this with an example:  $E \rightarrow E1 "+" T \{val[ntop] := val[top-2] + val[top]\}$   $E \rightarrow T \{val[top] := val[top]\}$   $T \rightarrow T1 "*" F \{val[ntop] := val[top-2] * val[top]\}$   $T \rightarrow F \{val[top] := val[top]\}$   $F \rightarrow "(" E ")" \{val[ntop] := val[top-1]\}$   $F \rightarrow num \{val[top] := num.lvalue\}$

Table 2.12: Figure shows the result of shift action. Now after performing reduce action by  $E \rightarrow E * T$  resulting stack is as shown in figure. Along with bottom-up parsing, this is how attributes can be evaluated using shift action/reduce action.

### 5.3 L-Attributed Definition

It allows both types, that is, synthesized as well as inherited. But if at all an inherited attribute is present, there is a restriction. The restriction is that each inherited attribute is restricted to inherit either from parent or from left sibling only. For example, consider the rule  $A \rightarrow XYZPQ$  assume that there is an inherited attribute, "i" is present with each non-terminal. Then,  $Z \bullet i = f(A \bullet i | X \bullet i | Y \bullet i)$  but  $Z \bullet i = f(P \bullet i | Q \bullet i)$  is wrong as they are right siblings. Semantic actions can be placed anywhere on the right hand side. Attributes are generally evaluated by traversing the parse tree depth first and left to right. It is possible to rewrite any L-attributes to S-attributed definition. L-attributed definition for converting infix to post fix form.

$$E \rightarrow TE'' E'' \rightarrow +T \#1 E'' \mid \epsilon T \rightarrow FT'' T'' \rightarrow * F \#2 T'' \mid \epsilon F \rightarrow id \#3$$

Where #1 corresponds to printing "+" operator, #2 corresponds to printing "\*", and # 3 corresponds to printing  $id.val$ . Look at the above SDT; there are no attributes, it is L-attributed definition as the semantic actions are in between grammar symbols. This is a simple example of L-attributed definition. Let us analyze this L-attributed definition and understand how to evaluate attributes with depth first left to right traversal. Take the parse tree for the input string "a + b\*c" and perform Depth first left to right traversal, i.e. at each node traverse the left sub tree depth wise completely

then right sub tree completely. Follow the traversal in. During the traversal whenever any dummy non-terminal is seen, carry out the translation. Converting L-Attributed to S-Attributed Definition Now that we understand that S-attributed is simple compared to L-attributed definition, let us see how to convert an L-attributed to an equivalent S-attributed. Consider an L-attributed with semantic actions in between the grammar symbols. Suppose we have an Lattributed as follows:  $S \rightarrow A \{ \} B$  How to convert it to an equivalent S-attributed definition? It is very simple!! Replace actions by non-terminal as follows:  $S \rightarrow A M B M \rightarrow \epsilon \{ \}$  Convert the following L-attributed definition to equivalent S-attributed definition.  $E \rightarrow TE'' E'' \rightarrow +T \#1 E'' | \epsilon T \rightarrow FT'' T'' \rightarrow *F \#2 T'' | \epsilon F \rightarrow id \#3$  Table 2.13: Solution Solution: Replace dummy non-terminals that is, actions by non-terminals.  $E \rightarrow TE'' E'' \rightarrow +T A E'' | \epsilon A \rightarrow \{ print(“+”); \} T \rightarrow FT'' T'' \rightarrow *F B T'' | \epsilon B \rightarrow \{ print(“*”); \} F \rightarrow id \{ print(“id”); \}$  Table 2.14: Solution 6. YACC YACC—Yet Another Compiler Compiler—is a tool for construction of automatic LALR parser generator. Using Yacc Yacc specifications are prepared in a file with extension “.y” For example, “test.y.” Then run this file with the Yacc command as “\$yacc test.y.” This translates yacc specifications into C-specifications under the default file mane “y.tab.c,” where all the translations are under a function name called yyparse(); Now compile “y.tab.c” with C-compiler and test the program. The steps to be performed are given below: Commands to execute \$yacc test.y This gives an output “y.tab.c,” which is a parser in c under a function name yyparse(). With –v option (\$yacc -v test.y), produces file y.output, which gives complete information about the LALR parser like DFA states, conflicts, number of terminals used, etc. \$cc y.tab.c \$./a.out Preparing the Yacc specification file Every yacc specification file consists of three sections: the declarations, grammar rules, and supporting subroutines. The sections are separated by double percent “%” marks. declarations % % Translation rules % % supporting subroutines The declaration section is optional. In case if there are no supporting subroutines, then the second % can also be skipped; thus, the smallest legal Yacc specification is % % Translation rules Declarations section Declaration part contains two types of declarations—Yacc declarations or C-declarations. To distinguish between the two, C-declarations are enclosed within %{} and %}. Here we can have C-declarations like global variable declarations (int x=l;), header files (#include....), and macro definitions(#define...). This may be used for defining subroutines in the last section or action part in grammar rules. Yacc declarations are nothing but tokens or terminals. We can define tokens by %token in the declaration part. For example, “num” is a terminal in grammar, then we define % token num in the declaration part. In grammar rules, symbols within single quotes are also taken as terminals. We can define the precedence and associativity of the tokens in the declarations section. This is done using %left, %right, followed by a list of tokens. The tokens defined on the same line will have the same precedence and associativity; the lines are listed in the order of increasing precedence. Thus, %left ‘+’ ‘-’ %left ‘\*’ ‘/’ are used to define the associativity and precedence of the four basic arithmetic operators ‘+’, ‘-’, ‘/’, ‘\*’. Operators ‘\*’ and ‘/’ have higher precedence than ‘+’ and both are left associative. The keyword %left is used to define left associativity and %right is used to define right associativity. Translation rules section This section is the heart of the yacc specification file. Here we write grammar. With each grammar rule, the user may associate actions to be performed each time the rule is recognized in the input process. These actions may return values, and may obtain the values returned by previous actions. Moreover, the lexical analyzer can return values when a token is matched. An action is defined with a set of C statements. Action can do input and output, call subprograms, and alter external vectors and variables. The action normally sets the pseudo variable “\$\$” to some value to return a value. For example, an action that does nothing but return the value 1 is { \$\$ = 1;} To obtain the values returned by previous actions, we use the pseudo-variables \$1, \$2, . . . , which refer to the values returned by the grammar symbol on the right side of a rule, reading from left to right. 7. Analysis Syntax Directed Definition Are a generalizations of context-free grammars in which: 1. Grammar symbols have an associated set of Attributes; 2. Productions are associated with Semantic Rules for computing the values of attributes. • Such formalism generates Annotated Parse-Trees where each node of the tree is a record with a field for each attribute (e.g., X.a indicates the attribute a of the grammar symbol X). • The value of an attribute of a grammar symbol at a given parse-tree node is defined by a semantic rule associated with the production used at that node. • We distinguish between two kinds of attributes: 1. Synthesized Attributes: They are computed from the values of the attributes of the children nodes. 2. Inherited Attributes: They are computed from the values of the attributes of both the siblings and the parent nodes. Form of Syntax Directed Definitions Each production,  $A ! \alpha$ , is associated with a set of semantic rules:  $b := f(c_1, c_2, \dots, c_k)$ , where  $f$  is a function and either.  $b$  is a synthesized attribute of  $A$ , and  $c_1, c_2, \dots, c_k$  are attributes of the grammar symbols of the production, or  $b$  is an inherited attribute of a grammar symbol in  $\alpha$ , and  $c_1, c_2, \dots, c_k$  are attributes of grammar symbols in  $\alpha$  or attributes of  $A$ .

**Checking & Run Time Environment** Type checking: Type system, Specification of Simple Type Checker, Equivalence of Expression, Types, Type Conversion, Overloading of Functions and Operations, Polymorphic Functions. Run Time Environment: Storage Organization, Storage Allocation Strategies, Parameter Passing, Dynamic Storage Allocation , Symbol Table.

1. Type Checking & Run Time Environment Type Checking Parsing cannot detect some errors. Some errors are captured during compile time called static checking (e.g., type compatibility). Languages like C, C++, C#, Java, and Haskell uses static checking. Static checking is even called early binding. During static checking programming errors are caught early. This causes program execution to be efficient. Static checking not only increases the efficiency and reliability of the compiled program, but also makes execution faster. Type checking is not only limited to compile time, it is even performed at execution time. This is done with the help of information gathered by a compiler; the information is gathered during compilation of the source program. Errors that are captured during run time are called dynamic checks (e.g., array bounds check or null pointers dereference check). Languages like Perl, python, and Lisp use dynamic checking. Dynamic checking is also called late binding. Dynamic checking allows some constructs that are rejected during static checking. A sound type system eliminates run-time type checking for type errors. A programming language is stronglytyped, if every program its compiler accepts will execute without type errors. In practice, some of the type checking operations is done at run-time (so, most of the programming languages are not strongly typed). For example, int x[100]; ... x[i] → most of the compilers cannot guarantee that i will be between 0 and 99 A semantic analyzer mainly performs static checking. Static checks can be any one of the following type of checks:

- Uniqueness checks: This ensures uniqueness of variables/objects in situations where it is required. For example, in most of the languages no identifier can be used for two different definitions in the same scope.
- Flow of control checks: Statements that cause flow of control to leave a construct should have a place to transfer flow of control. If this place is missing, it is confusion. For example, in C language, “break” causes flow of control to exit from the innermost loop. If it is used without a loop, it confuses where to leave the flow of control.
- Type checks: A compiler should report an error if an operator is applied to incompatible operands. For example, for binary addition, operands are array and a function is incompatible. In a function, the number of arguments should match with the number of formals and the corresponding types.

Name-related checks: Sometimes, the same name must appear two or more times. For example, in ADA, a loop or a block may have a name that appears at the beginning and end of the construct. The compiler must check whether the same name is used at both places. What does semantic analysis do? It performs checks of many kinds which may include

- All identifiers are declared before being used.
- Type compatibility.
- Inheritance relationships.
- Classes defined only once.
- Methods in a class defined only once.
- Reserved words are not misused.

In this chapter we focus on type checking. The above examples indicate that most of the other static checks are routine and can be implemented using the techniques of SDT discussed in the previous chapter. Some of them can be combined with other activities. For example, for uniqueness check, while entering the identifier into the symbol table, we can ensure that it is entered only once. Now let us see how to design a type checker. A type checker verifies that the type of a construct matches with that expected by its context. For example, in C language, the type checker must verify that the operator “%” should have two integer operands dereferencing is applied through a pointer, indexing is done only on an array, a user-defined function is applied with correct number and type of arguments. The goal of a type checker is to ensure that operations are applied to the correct type of operands. Type information collected by a type checker is used later by code generator.

### 1.1 Type Systems

Consider the assembly language program fragment. Add R1, R2, and R3. What are the types of operands R1, R2, R3? Based on the possible type of operands and its values, operations are legal. It doesn’t make sense to add a character and a function pointer in C language. It does make sense to add two float or int values. Irrespective of the type, the assembly language implementation remains the same for add. A language’s type system specifies which operations are valid for which types. A type system is a collection of rules for assigning types to the various parts of a program. A type checker implements a type system. Types are represented by type expressions. Type system has a set of rules defined that take care of extracting the data types of each variables and check for the compatibility during the operation.

### 1.2 Type Expressions

The type expressions are used to represent the type of a programming language construct. Type expression can be a basic type or formed by recursively applying an operator called a type constructor to other type expressions. The basic types and constructors depend on the source language to be verified. Let us define type expression as follows:

- A basic type is a type expression
- Boolean, char, integer, real, void, type\_error
- A type constructor applied to type expressions is a type expression
- Array: array(l, T)

Array ( $I, T$ ) is a type expression denoting the type of an array with elements of type  $T$  and index set  $I$ , where  $T$  is a type expression. Index set  $I$  often represents a range of integers. For example, the Pascal declaration var C: array[1..20] of integer; associates the type expression array(1..20, integer) with C. • Product:  $T_1 \times T_2$  • If  $T_1$  and  $T_2$  are two type expressions, then their Cartesian product  $T_1 \times T_2$  is a type expression. We assume that  $\times$  associates to the left. • Record: record( $(N_1 \times T_1) \times (N_2 \times T_2)$ ) A record differs from a product. The fields of a record have names. The record type constructor will be applied to a tuple formed from field types and field names. For example, the Pascal program fragment type node = record address: integer ; data : array [1..15] of char end; var node table : array [1..10] of node ; declares the type name “node” representing the type expression record((address×integer) × (data × array(1..15,char))) and the variable “node\_table” to be an array of records of this type. Pointer: pointer ( $T$ ) Pointer( $T$ ) is a type expression denoting the type “pointer to an object of type  $T$  where  $T$  is a type expression. For example, in Pascal, the declaration var ptr: \*row declares variable “ptr” to have type pointer(row). Function:  $D \rightarrow R$  Mathematically, a function is a mapping from elements of one set called domain to another set called range. We may treat functions in programming languages as mapping a domain type “Dom” to a range type “Rg.”. The type of such a function will be denoted by the type expression  $\text{Dom} \rightarrow \text{Rg}$ . For example, the built-in function mod, i.e. modulus of Pascal has type expression  $\text{int} \times \text{int} \rightarrow \text{int}$ . As another example, the Pascal declaration function fun(a, b: char) \* integer; says that the domain type of function “fun” is denoted by “char × char” and range type by “pointer(integer).” The type expression of function “fun” is thus denoted as follows:  $\text{char} \times \text{char} \rightarrow \text{pointer}(\text{integer})$  However, there are some languages like Lisp that allow functions to return objects of arbitrary types. For example, we can define a function “g” of type  $(\text{integer} \rightarrow \text{integer}) \rightarrow (\text{integer} \rightarrow \text{integer})$ . That is, function “g” takes as input a function that maps an integer to an integer and “g” produces another function of the same type as output.

### 1.3 Design Of Simple Type Checker

Different type systems are designed for different languages. The type checking can be done in two ways. The checking done at compile time is called static checking and the checking done at run time is called dynamic checking. A system is said to be a Sound System if it completely eliminates the dynamic check. In such systems, if the type checker assigns any type other than type error for some fragment of code, then there is no need to check for errors when it is run. Practically this is not always true; for example, if an array  $X$  is declared to hold 100 elements. Usually the index would be from 0 to 99 or from 1 to 100 depending on the language support. And there is a statement in the program referred to as  $X[i]$ ; during compilation this would not guarantee error free at runtime as there is possibility that if the value of  $i$  is 120 at run time then there will be an error. Therefore, it is essential that there is a need even for the dynamic check to be done. Let us consider a simple language that has declaration statements followed by statements, where these statements are simple arithmetic statements, conditional statements, iterative statements, and functional statements. The program block of code can be generated by defining the rules as follows:

**Type Declarations**
 $P \rightarrow D ;$ 
 $E \rightarrow D ;$ 
 $D \rightarrow id :$ 
 $T \{ \text{add\_type}(id.\text{entry}, T.\text{type}) \}$ 
 $T \rightarrow \text{char} \{ T.\text{type} := \text{char} \}$ 
 $T \rightarrow \text{integer} \{ T.\text{type} := \text{int} \}$ 
 $\dots$ 
 $T \rightarrow \text{array} ["\text{num}"] \{ T.\text{type} := \text{array}(\text{num.value}, T1.\text{type}) \}$

**Table 3.1: Type Declarations**  
 These rules are defined to write the declaration statements followed by expression statements. The semantic rule  $\{ \text{add\_type}(id.\text{entry}, T.\text{type}) \}$  indicates to associate type in  $T$  with the identifier and add this type info into the symbol table during parsing. A semantic rule of the form  $\{ T.\text{type} := \text{int} \}$  associates the type of  $T$  to integer. So the above SDT collects type information and stores in symbol table.

### 1.4 TYPE CHECKING OF EXPRESSIONS

Let us see how to type check expressions. The expressions like  $3 \bmod 5$ ,  $A[10]$ ,  $*p$  can be generated by the following rules. The semantic rules are defined as follows to extract the type information and to check for compatibility.

$E \rightarrow \text{literal} \{ E.\text{type} := \text{char} \}$ 
 $E \rightarrow \text{num} \{ E.\text{type} := \text{int} \}$ 
 $E \rightarrow id \{ E.\text{type} := \text{lookup}(id.\text{entry}) \}$ 
 $E \rightarrow E1 \bmod E2 \{ E.\text{type} := \text{if } E1.\text{type} = \text{int} \text{ and } E2.\text{type} = \text{int} \text{ then int else type\_error} \}$ 
 $E \rightarrow E1 "[" E2 "] \{ E.\text{type} := \text{if } E1.\text{type} = \text{array}(s, t) \text{ and } E2.\text{type} = \text{int} \text{ then t else type\_error} \}$ 
 $E \rightarrow "*" E1 \{ E.\text{type} := \text{if } E1.\text{type} = \text{pointer}(t) \text{ then t else type\_error} \}$

**Table 3.2: Type Checking Of Expressions**  
 When we write a statement as  $i \bmod 10$ , then while parsing the element  $i$ , it uses the rule as  $E \rightarrow id$  and performs the action of getting the data type for the id from the symbol table using the lookup method. When it parses the lexeme 10, it uses the rule  $E \rightarrow \text{num}$  and assigns the type as int. While parsing the complete statement  $i \bmod 10$ , it uses the rule  $E \rightarrow E1 \bmod E2$ , which checks the data types in both  $E1$  and  $E2$  and if they are the same it returns int otherwise type\_error.

### 1.5 TYPE CHECKING OF STATEMENTS

The statements are simple of the form “ $a = b + c$ ” or “ $a = b$ .” It can be a combination of statements followed by another statement or a conditional statement or iterative. To generate either a simple or a complex group of statements, the rules can be framed as follows: To validate the statement a special data type void is defined, which is assigned to a statement only when it is valid at expression level, otherwise type\_error is assigned to indicate that it is

invalid. If there is an error at expression level, then it is propagated to the statement, from the statement it is propagated to a set of statements and then to the entire block of program.  $P \rightarrow D ; S$   $S \rightarrow id := E$  { $S.type :=$  if  $lookup(id.entry) = E.type$ }  $S \rightarrow S_1 ; S_2$  then void else type\_error} { $S.type :=$  if  $S_1.type = void$  and  $S_2.type = void$  then void else type\_error}  $S \rightarrow$  if  $E$  then  $S_1$  { $S.type :=$  if  $E.type = boolean$  then  $S_1.type$  else type\_error}  $S \rightarrow$  while  $E$  do  $S_1$  { $S.type :=$  if  $E.type = boolean$  then  $S_1.type$  else type\_error} Table 3.3: Type Checking Of Statements 1.6 Type Conversion In an expression, if there are two operands of different type, then it may be required to convert one type to another in order to perform the operation. For example, the expression “ $a + b$ ,” if  $a$  is of integer and  $b$  is real, then to perform the addition  $a$  may be converted to real. The type checker can be designed to do this conversion. The conversion done automatically by the compiler is implicit conversion and this process is known as coercion. If the compiler insists the programmer to specify this conversion, then it is said to be explicit. For instance, all conversions in Ada are said to be explicit. The semantic rules for type conversion are listed below.  $E \rightarrow num$  { $E.type := int$ }  $E \rightarrow num.num$  { $E.type := real$ }  $E \rightarrow id$  { $E.type := lookup(id.entry)$ }  $E \rightarrow E_1 op E_2$  { $E.type :=$  if  $E_1.type = int$  and  $E_2.type = int$  then int else if  $E_1.type = int$  and  $E_2.type = real$  then real else if  $E_1.type = real$  and  $E_2.type = int$  then real else if  $E_1.type = real$  and  $E_2.type = real$  then real else type\_error} Table 3.4: Type Conversion 2. Overloading Of Functions And Operators An operator is overloaded if the same operator performs different operations. For example, in arithmetic expression  $a + b$ , the addition operator “ $+$ ” is overloaded because it performs different operations, when  $a$  and  $b$  are of different types like integer, real, complex, and so on. Another example of operator overloading is overloaded parenthesis in ada, that is, the expression  $A(i)$  has different meanings. It can be the  $i$ th element of an array, or a call to function  $A$  with argument  $i$ , and so on. Operator overloading is resolved when the unique definition for an overloaded operator is determined. The process of resolving overloading is called operator identification because it specifies what operation an operator performs. The overloading of arithmetic operators can be easily resolved by processing only the operands of an operator. Like operator overloading, the function can also be overloaded. In function overloading, the functions have the same name but different numbers and arguments of different types. In Ada, the operator “ $*$ ” has the standard meaning that it takes a pair of integers and returns an integer. The function of “ $*$ ” can be overloaded by adding the following declarations: Function “ $*$ ”( $a,b: integer$ ) return integer. Function “ $*$ ”( $a,b: complex$ ) return integer. Function “ $*$ ”( $a,b: complex$ ) return complex. By addition of the above declarations, now the operator “ $*$ ” can take the following possible types: • It takes a pair of integers and returns an integer • It takes a pair of integers and returns a complex number • It takes a pair of complex numbers and returns a complex number Function overloading can be resolved by the type checker based on the number and types of arguments. The type checking rule for function by assuming that each expression has a unique type is given as  $E \rightarrow E_1(E_2) \{ E.type := t \}$   $E_2.type := t \rightarrow u$  then  $E.type := u$  else  $E.type := type\_error$  }  $E' \rightarrow E$  { $E'.type := E.type$ }  $E \rightarrow id$  { $E.type := lookup(id.entry)$ }  $E \rightarrow E_1(E_2) \{ E.type := \{ u \mid \text{there exists an } s \text{ in } E_2.type \text{ such that } s \rightarrow u \text{ is in } E_1.type \} \}$  Table 3.5: Overloading Of Functions and Operators 3. Polymorphic Functions A piece of code is said to be polymorphic if the statements in the body can be executed with different types. A function that takes the arguments of different types and executes the same code is a polymorphic function. The type checker designed for a language like Ada that supports polymorphic functions, the type expressions are extended to include the expressions that vary with type variables. The same operation performed on different types is called overloading and are often found in object-oriented programming. For example, let us consider the function that takes two arguments and returns the result.  $\text{int add(int, int)}$   $\text{int add(real, real)}$   $\text{real add(real, real)}$  The type expression for the function add is given as  $\text{int} \times \text{int} \rightarrow \text{int}$   $\text{real} \times \text{real} \rightarrow \text{real}$  Write type expression for an array of pointer to real, where the array index ranges from 1 to 100. Solution: The type expression is  $\text{array}[1..100, \text{pointer}(\text{real})]$  Write a type expression for a two-dimensional array of integers (that is, an array of arrays) whose rows are indexed from 0 to 9 and whose columns are indexed from -10 to 10. Solution: Type expression is  $\text{array}[0..9, \text{array}[-10..10, \text{integer}]]$  Write a type expression for a function whose domains are functions from integers to pointers to integers and whose ranges are records consisting of an integer and a character. Solution: Type expression is  $\text{Domain type expression is integer} \rightarrow \text{pointer(integer)}$  Let range has two fields  $a$  and  $b$  of type integer and character respectively. Range type expression is  $\text{record}((a \times \text{integer})(b \times \text{character}))$  The final type expression is  $(\text{integer} \rightarrow \text{pointer(integer)}) \rightarrow \text{record}((a \times \text{integer})(b \times \text{character}))$  Consider the following program in C and the write the type expression for abc. 

```
typedef struct { int a,b; } NODE; NODE abc[100];
```

 Solution: The type expression for NODE is  $\text{record}((a \times \text{integer}) \times (b \times \text{integer}))$  abc is an array of NODE; hence, its type expression is  $\text{array}[0..99, \text{record}((a \times \text{integer}) \times (b \times \text{integer}))]$  Consider the following declarations. type cell=record info: integer; next: pointer(cell) Type link = ↑ cell; Var next = link; last = link; p = ↑ cell; q,r = ↑ cell; Table 3.6: Solution Among the following,

which expressions are structurally equivalent? Which are name equivalent? Justify your answer.

- link
- Pointer(cell)
- Pointer(link)
- Pointer(record ((info × integer) × (next × pointer (cell))))

Solution: Let A = link B = pointer (cell) C = pointer (link) D = Pointer (record ((info × integer) × (next × pointer (cell))))

Table 3.7: Solution To get structural equivalence we need to substitute each type name by its type expression. We know that, link is a type name. If we substitute pointer (cell) for each appearance of link we get, A = pointer (cell) B = pointer (cell) C = pointer (pointer (cell)) D = Pointer (record ((info × integer) × (next × pointer (cell))))

We know that, cell is also type name given by type cell=record info: integer; next: pointer(cell)

Substituting type expression for cell in A and B, we get A = pointer (record ((info × integer) × (next × pointer (cell)))) B = pointer (record ((info × integer) × (next × pointer (cell)))) C = pointer( pointer(cell)) D = Pointer (record ((info × integer) × (next × pointer (cell))))

We have not substituted for the type expression of cell in "C" as it is anyway different from A, B, and D. That is, even if we substitute in C, the type expression will not be the same for A, B, C, and D. We can say that A, B, and D are structurally equivalent. For name equivalence, we will not do any substitutions. Rather we look at type expressions directly. If they are the same then we say they are name equivalent. None of A, B, C, D are name equivalent.

#### 4 Run Time Environment

- 4.1 Storage Allocation Information
  - Information about storage locations is kept in the symbol table
  - If target is assembly code then assembler can take care of storage for various names
  - Compiler needs to generate data definitions to be appended to assembly code
  - If target is machine code then compiler does the allocation
  - For names whose storage is allocated at runtime no storage allocation is done
- Information about the storage locations that will be bound to names at run time is kept in the symbol table. If the target is assembly code, the assembler can take care of storage for various names. All the compiler has to do is to scan the symbol table, after generating assembly code, and generate assembly language data definitions to be appended to the assembly language program for each name. If machine code is to be generated by the compiler, then the position of each data object relative to a fixed origin must be ascertained. The compiler has to do the allocation in this case. In the case of names whose storage is allocated on a stack or heap, the compiler does not allocate storage at all, it plans out the activation record for each procedure.

#### 4.2 Storage Organization:

code Static data stack ... heap

Figure 3.1: Storage stack

This kind of organization of run-time storage is used for languages such as FORTRAN, Pascal and C. The size of the generated target code, as well as that of some of the data objects, is known at compile time. Thus, these can be stored in statically determined areas in the memory. Pascal and C use the stack for procedure activations. Whenever a procedure is called, execution of activation gets interrupted, and information about the machine state (like register values) is stored on the stack. When the called procedure returns, the interrupted activation can be restarted after restoring the saved machine state. The heap may be used to store dynamically allocated data objects, and also other stuff such as activation information (in the case of languages where an activation tree cannot be used to represent lifetimes). Both the stack and the heap change in size during program execution, so they cannot be allocated a fixed amount of space. Generally they start from opposite ends of the memory and can grow as required, towards each other, until the space available has filled up.

Activation Record:

- temporaries: used in expression evaluation
- local data: field for local data
- saved machine status: holds info about machine status before procedure call
- access link : to access non local data
- control link : points to activation record of caller
- actual parameters: field to hold actual parameters
- returned value : field for holding value to be returned

Figure 3.2: Storage Organization

The activation record is used to store the information required by a single procedure call. Not all the fields shown in the figure may be needed for all languages. The record structure can be modified as per the language/compiler requirements. For Pascal and C, the activation record is generally stored on the run-time stack during the period when the procedure is executing. Of the fields shown in the figure, access link and control link are optional (e.g. FORTRAN doesn't need access links). Also, actual parameters and return values are often stored in registers instead of the activation record, for greater efficiency. The activation record for a procedure call is generated by the compiler. Generally, all field sizes can be determined at compile time. However, this is not possible in the case of a procedure which has a local array whose size depends on a parameter. The strategies used for storage allocation in such cases will be discussed in the coming slides.

#### 4.3 Storage Allocation Strategies

These represent the different storage-allocation strategies used in the distinct parts of the run-time memory organization (as shown in slide 8). We will now look at the possibility of using these strategies to allocate memory for activation records. Different languages use different strategies for this purpose. For example, old FORTRAN used static allocation, Algol type languages use stack allocation, and LISP type languages use heap allocation.

Static allocation: Names are bound to storage as the program is compiled. No runtime support is required. Bindings do not change at run time. On every invocation of procedure names are bound to the same storage. Values of

local names are retained across activations of a procedure. These are the fundamental characteristics of static allocation. Since name binding occurs during compilation, there is no need for a run-time support package. The retention of local name values across procedure activations means that when control returns to a procedure, the values of the locals are the same as they were when control last left. For example, suppose we had the following code, written in a language using static allocation: function F( ) { int a; print(a); a = 10; } After calling F( ) once, if it was called a second time, the value of a would initially be 10, and this is what would get printed. Type of a name determines the amount of storage to be set aside. Address of a storage consists of an offset from the end of an activation record. Compiler decides location of each activation. All the addresses can be filled at compile time. Constraints - Size of all data objects must be known at compile time - Recursive procedures are not allowed - Data structures cannot be created dynamically. The type of a name determines its storage requirement, as outlined in slide 11. The address for this storage is an offset from the procedure's activation record, and the compiler positions the records relative to the target code and to one another (on some computers, it may be possible to leave this relative position unspecified, and let the link editor link the activation records to the executable code). After this position has been decided, the addresses of the activation records, and hence of the storage for each name in the records, are fixed. Thus, at compile time, the addresses at which the target code can find the data it operates upon can be filled in. The addresses at which information is to be saved when a procedure call takes place are also known at compile time. Static allocation does have some limitations:

- Size of data objects, as well as any constraints on their positions in memory, must be available at compile time.
- No recursion, because all activations of a given procedure use the same bindings for local names.
- No dynamic data structures, since no mechanism is provided for run time storage allocation.

Stack Allocation

The activation records that are pushed onto and popped from the run time stack as the control flows through the given activation tree. First the procedure is activated. Procedure read array's activation is pushed onto the stack, when the control reaches the first line in the procedure sort. After the control returns from the activation of the read array, its activation is popped. In the activation of sort, the control then reaches a call of qsort with actuals 1 and 9 and an activation of qsort is pushed onto the top of the stack. In the last stage the activations for partition (1,3) and qsort (1,0) have begun and ended during the life time of qsort (1,3), so their activation records have come and gone from the stack, leaving the activation record for qsort (1,3) on top.

Calling Sequence:

A call sequence allocates an activation record and enters information into its field A return sequence restores the state of the machine so that calling procedure can continue execution

Figure 3.3: Calling Sequence in Stack

A call sequence allocates an activation record and enters information into its fields. A return sequence restores the state of the machine so that the calling sequence can continue execution. Calling sequence and activation records differ, even for the same language. The code in the calling sequence is often divided between the calling procedure and the procedure it calls. There is no exact division of runtime tasks between the caller and the callee. As shown in the figure, the register stack top points to the end of the machine status field in the activation record. This position is known to the caller, so it can be made responsible for setting up stack top before control flows to the called procedure. The code for the callee can access its temporaries and the local data using offsets from stack top.

Call Sequence:

Caller evaluates the actual parameters Caller stores return address and other values (control link) into callee's activation record Callee saves register values and other status information Callee initializes its local data and begins execution The fields whose sizes are fixed early are placed in the middle. The decision of whether or not to use the control and access links is part of the design of the compiler, so these fields can be fixed at compiler construction time. If exactly the same amount of machine-status information is saved for each activation, then the same code can do the saving and restoring for all activations. The size of temporaries may not be known to the front end. Temporaries needed by the procedure may be reduced by careful code generation or optimization. This field is shown after that for the local data. The caller usually evaluates the parameters and communicates them to the activation record of the callee. In the runtime stack, the activation record of the caller is just below that for the callee. The fields for parameters and a potential return value are placed next to the activation record of the caller. The caller can then access these fields using offsets from the end of its own activation record. In particular, there is no reason for the caller to know about the local data or temporaries of the callee.

Return Sequence:

Callee places a return value next to activation record of caller Restores registers using information in status field Branch to return address Caller copies return value into its own activation record As described earlier, in the runtime stack, the activation record of the caller is just below that for the callee. The fields for parameters and a potential return value are placed next to the activation record of the caller. The caller can then access these fields using offsets from the end of its own activation record. The caller copies the return

value into its own activation record. In particular, there is no reason for the caller to know about the local data or temporaries of the callee. The given calling sequence allows the number of arguments of the called procedure to depend on the call. At compile time, the target code of the caller knows the number of arguments it is supplying to the callee. The caller knows the size of the parameter field. The target code of the called must be prepared to handle other calls as well, so it waits until it is called, then examines the parameter field. Information describing the parameters must be placed next to the status field so the callee can find it.

**Long Length Data** The procedure P has three local arrays. The storage for these arrays is not part of the activation record for P; only a pointer to the beginning of each array appears in the activation record. The relative addresses of these pointers are known at the compile time, so the target code can access array elements through the pointers. Also shown is the procedure Q called by P. The activation record for Q begins after the arrays of P. Access to data on the stack is through two pointers, top and stack top. The first of these marks the actual top of the stack; it points to the position at which the next activation record begins. The second is used to find the local data. For consistency with the organization of the figure in slide 16, suppose the stack top points to the end of the machine status field. In this figure the stack top points to the end of this field in the activation record for Q. Within the field is a control link to the previous value of stack top when control was in calling activation of P. The code that repositions top and stack top can be generated at compile time, using the sizes of the fields in the activation record. When q returns, the new value of top is stack top minus the length of the machine status and the parameter fields in Q's activation record. This length is known at the compile time, at least to the caller. After adjusting top, the new value of stack top can be copied from the control link of Q.

**Dangling references:** Referring to locations which have been deallocated

```
main() {int *p; p = dangle(); /* dangling reference */ } int *dangle(); { int i=23; return &i; }
```

The problem of dangling references arises, whenever storage is de-allocated. A dangling reference occurs when there is a reference to storage that has been deallocated. It is a logical error to use dangling references, since the value of de-allocated storage is undefined according to the semantics of most languages. Since that storage may later be allocated to another datum, mysterious bugs can appear in the programs with dangling references.

**Heap Allocation:** Stack allocation cannot be used if: The values of the local variables must be retained when an activation ends. A called activation outlives the caller. In such a case de-allocation of activation record cannot occur in last-in first-out fashion. Heap allocation gives out pieces of contiguous storage for activation records. There are two aspects of dynamic allocation :- Runtime allocation and de-allocation of data structures. Languages like Algol have dynamic data structures and it reserves some part of memory for it. If a procedure wants to put a value that is to be used after its activation is over then we cannot use stack for that purpose. That is language like Pascal allows data to be allocated under program control. Also in certain language a called activation may outlive the caller procedure. In such a case last-in-first-out queue will not work and we will require a data structure like heap to store the activation. The last case is not true for those languages whose activation trees correctly depict the flow of control between procedures. Pieces may be de-allocated in any order. Over time the heap will consist of alternate areas that are free and in use. Heap manager is supposed to make use of the free space. For efficiency reasons it may be helpful to handle small activations as a special case. For each size of interest keep a linked list of free blocks of that size. Initializing data-structures may require allocating memory but where to allocate this memory. After doing type inference we have to do storage allocation. It will allocate some chunk of bytes. But in language like lisp it will try to give continuous chunk. The allocation in continuous bytes may lead to problem of fragmentation i.e. you may develop hole in process of allocation and de-allocation. Thus storage allocation of heap may lead us with many holes and fragmented memory which will make it hard to allocate continuous chunk of memory to requesting program. So we have heap managers which manage the free space and allocation and de-allocation of memory. It would be efficient to handle small activations and activations of predictable size as a special case as described in the next slide. The various allocation and de-allocation techniques used will be discussed later.

- Fill a request of size s with block of size s' where s' is the smallest size greater than or equal to s •
- For large blocks of storage use heap manager • For large amount of storage computation may take some time to use up memory so that time taken by the manager may be negligible compared to the computation time. As mentioned earlier, for efficiency reasons we can handle small activations and activations of predictable size as a special case as follows: For each size of interest, keep a linked list of free blocks of that size. If possible, fill a request for size s with a block of size s', where s' is the smallest size greater than or equal to s. When the block is eventually de-allocated, it is returned to the linked list it came from. For large blocks of storage use the heap manager. Heap manager will dynamically allocate memory. This will come with a runtime overhead. As heap manager will have to take care of defragmentation and garbage

collection. But since heap manager saves space otherwise we will have to fix size of activation at compile time, runtime overhead is the price worth it. Access to non-local names : Scope rules determine the treatment of non-local names A common rule is lexical scoping or static scoping (most languages use lexical scoping) The scope rules of a language decide how to reference the non-local variables. There are two methods that are commonly used: 1. Static or Lexical scoping: It determines the declaration that applies to a name by examining the program text alone. E.g., Pascal, C and ADA. 2. Dynamic Scoping: It determines the declaration applicable to a name at run time, by considering the current activations. E.g., Lisp 5. Dynamic Storage Allocation: Generally languages like Lisp and ML which do not allow for explicit de-allocation of memory do garbage collection. A reference to a pointer that is no longer valid is called a 'dangling reference'. For example, consider this C code: int main (void) { int\* a=fun(); } int\* fun() { int a=3; int\* b=&a; return b; } Here, the pointer returned by fun() no longer points to a valid address in memory as the activation of fun() has ended. This kind of situation is called a 'dangling reference'. In case of explicit allocation it is more likely to happen as the user can de-allocate any part of memory, even something that has to a pointer pointing to a valid piece of memory. Explicit Allocation of Fixed Sized Blocks Link the blocks in a list Allocation and de-allocation can be done with very little overhead The simplest form of dynamic allocation involves blocks of a fixed size. By linking the blocks in a list, as shown in the figure, allocation and de-allocation can be done quickly with little or no storage overhead. Explicit Allocation of Fixed Sized Blocks Blocks are drawn from contiguous area of storage An area of each block is used as pointer to the next block A pointer available points to the first block Allocation means removing a block from the available list De-allocation means putting the block in the available list Compiler routines need not know the type of objects to be held in the blocks Each block is treated as a variant record Suppose that blocks are to be drawn from a contiguous area of storage. Initialization of the area is done by using a portion of each block for a link to the next block. A pointer available points to the first block. Generally a list of free nodes and a list of allocated nodes is maintained, and whenever a new block has to be allocated, the block at the head of the free list is taken off and allocated (added to the list of allocated nodes). When a node has to be de-allocated, it is removed from the list of allocated nodes by changing the pointer to it in the list to point to the block previously pointed to by it, and then the removed block is added to the head of the list of free blocks. The compiler routines that manage blocks do not need to know the type of object that will be held in the block by the user program. These blocks can contain any type of data (i.e., they are used as generic memory locations by the compiler). We can treat each block as a variant record, with the compiler routines viewing the block as consisting of some other type. Thus, there is no space overhead because the user program can use the entire block for its own purposes. When the block is returned, then the compiler routines use some of the space from the block itself to link it into the list of available blocks, as shown in the figure in the last slide. Explicit Allocation of Variable Size Blocks Storage can become fragmented Situation may arise If program allocates five blocks then de-allocates second and fourth block Fragmentation is of no consequence if blocks are of fixed size Blocks cannot be allocated even if space is available In explicit allocation of fixed size blocks, internal fragmentation can occur, that is, the heap may consist of alternate blocks that are free and in use, as shown in the figure. The situation shown can occur if a program allocates five blocks and then de-allocates the second and the fourth, for example. Fragmentation is of no consequence if blocks are of fixed size, but if they are of variable size, a situation like this is a problem, because we could not allocate a block larger than any one of the free blocks, even though the space is available in principle. So, if variable-sized blocks are allocated, then internal fragmentation can be avoided, as we only allocate as much space as we need in a block. But this creates the problem of external fragmentation, where enough space is available in total for our requirements, but not enough space is available in continuous memory locations, as needed for a block of allocated memory. For example, consider another case where we need to allocate 400 bytes of data for the next request, and the available continuous regions of memory that we have are of sizes 300, 200 and 100 bytes. So we have a total of 600 bytes, which is more than what we need. But still we are unable to allocate the memory as we do not have enough contiguous storage. The amount of external fragmentation while allocating variable-sized blocks can become very high on using certain strategies for memory allocation. So we try to use certain strategies for memory allocation, so that we can minimize memory wastage due to external fragmentation. These strategies are discussed in the next few slides. 6 Symbol Table Compiler uses symbol table to keep track of scope and binding information about names Symbol table is changed every time a name is encountered in the source; changes to table occur • if a new name is discovered • if new information about an existing name is discovered Symbol table must have mechanism to: • add new entries • find existing information efficiently Two common mechanisms: • linear lists, simple to implement, poor performance • hash tables, greater

programming/space overhead, good performance Compiler should be able to grow symbol table dynamically if size is fixed, it must be large enough for the largest program A compiler uses a symbol table to keep track of scope and binding information about names. It is filled after the AST is made by walking through the tree, discovering and assimilating information about the names. There should be two basic operations - to insert a new name or information into the symbol table as and when discovered and to efficiently lookup a name in the symbol table to retrieve its information.

**Variable Information(type) Space (byte)**

A	Integer	2	B	float	4	C	Float	8	D	Character	1	...	.....
---	---------	---	---	-------	---	---	-------	---	---	-----------	---	-----	-------

Table 3.8: Symbol Table Two common data structures used for the symbol table are -

- Linear lists:- simple to implement, poor performance.
- Hash tables:- greater programming/space overhead, good performance.

Figure 3.4: Symbol table Ideally a compiler should be able to grow the symbol table dynamically, i.e., insert new entries or information as and when needed. But if the size of the table is fixed in advance then ( an array implementation for example), then the size must be big enough in advance to accommodate the largest possible program.

- each entry for a declaration of a name
- format need not be uniform because information depends upon the usage of the name
- each entry is a record consisting of consecutive words
- to keep records uniform some entries may be outside the symbol table
- information is entered into symbol table at various times
- keywords are entered initially
- identifier lexemes are entered by lexical analyzer
- symbol table entry may be set up when role of name becomes clear
- attribute values are filled in as information is available For each declaration of a name, there is an entry in the symbol table. Different entries need to store different information because of the different contexts in which a name can occur. An entry corresponding to a particular name can be inserted into the symbol table at different stages depending on when the role of the name becomes clear. The various attributes that an entry in the symbol table can have are lexeme, type of name, size of storage and in case of functions - the parameter list etc. a name may denote several objects in the same block int x; struct x {float y, z; }
- lexical analyzer returns the name itself and not pointer to symbol table entry
- record in the symbol table is created when role of the name becomes clear
- in this case two symbol table entries will be created
- attributes of a name are entered in response to declarations
- labels are often identified by colon
- syntax of procedure/function specifies that certain identifiers are formals
- characters in a name
- there is a distinction between token id, lexeme and attributes of the names
- it is difficult to work with lexemes
- if there is modest upper bound on length then lexemes can be stored in symbol table
- if limit is large store lexemes separately

There might be multiple entries in the symbol table for the same name, all of them having different roles. It is quite intuitive that the symbol table entries have to be made only when the role of a particular name becomes clear. The lexical analyzer therefore just returns the name and not the symbol table entry as it cannot determine the context of that name. Attributes corresponding to the symbol table are entered for a name in response to the corresponding declaration. There has to be an upper limit for the length of the lexemes for them to be stored in the symbol table.

#### ..... UNIT IV

Code Generation Intermediate Code Generation: Declarations, Assignment statements, Boolean Expressions, Case Statements, Backpatching, Procedure Calls, Code Generation: Issues in the Design of Code Generator, Basic Block and Flow Graphs, Register Allocation and Assignment, DAG Representation of Basic Blocks, Peephole Optimization, Generating Code from DAG.

1. Intermediate Code Generation The intermediate code is useful representation when compilers are designed as two pass system, i.e. as front end and back end. The source program is made source language independent by representing it in intermediate form, so that the back end is filtered from source language dependence. The intermediate code can be generated by modifying the syntax-directed translation rules to represent the program in intermediate form. This phase of intermediate code generation comes after semantic analysis and before code optimization.

Benefits of intermediate code

- Intermediate code makes target code generation easier
- It helps in retargeting, that is, creating more and more compilers for the same source language but for different machines.
- As intermediate code is machine independent, it helps in machine-independent code optimization.

Figure 4.1:Intermediate code generation phase A compiler front end is organized as in figure above, where parsing, static checking, and intermediate-code generation are done sequentially; sometimes they can be combined and folded into parsing. All schemes can be implemented by creating a syntax tree and then walking the tree.

1.1 Intermediate code can be represented in the following four ways.

- Syntax trees
- Directed acyclic graph(DAG)
- Postfix notation
- Three address code

1.1.1 Syntax Trees A syntax tree is a graphical representation of the source program. Here the node represents an operator and children of the node represent operands. It is a hierarchical structure that can be constructed by syntax rules. The target code can be generated by traversing the tree in post order form. For instance, consider an assignment statement  $a = b^* - (c - d) + b^* - (c - d)$  when

represented using the syntax tree. The tree for the statement  $a = b^* - (c - d) + b^* - (c - d)$  is constructed by creating the nodes in the following order.   
 $p1 = \text{mkleaf(id, c)}$   $p2 = \text{mkleaf(id, d)}$   $p3 = \text{mknnode}(' - ', p1, p2)$   $p4 = \text{mknnode}('U', p3, \text{NULL})$   $p5 = \text{mkleaf(id, b)}$    
**Production Semantic Rule**  $S \rightarrow id := E$   $S.\text{nptr} := \text{mknnode}(\text{'assign'}, \text{mkleaf(id, id.place)}, E.\text{nptr})$   $E \rightarrow E1 + E2$   $E.\text{nptr} := \text{mknnode}(' + ', E1.\text{nptr}, E2.\text{nptr})$   $E \rightarrow E1 * E2$   $E.\text{nptr} := \text{mknnode}(' * ', E1.\text{nptr}, E2.\text{nptr})$   $E \rightarrow - E1$   $E.\text{nptr} := \text{mknnode}('uminus', E1.\text{nptr})$   $E \rightarrow (E1)$   $E.\text{nptr} := E1.\text{nptr}$   $E \rightarrow id$   $E.\text{nptr} := \text{mkleaf(id, id.place)}$

**Table 4.1: Production in Syntax tree**

**1.1.2 Directed Acyclic Graph (DAG)** The tree that shows the same information with identified common sub-expression is called Directed Acyclic Graph (DAG). On examining the above example, it is observed that there are some nodes that are unnecessarily created. To avoid extra nodes these functions can be modified to check the existence of similar Front End Intermediate code generator Machine Code Generator node before creating it. If a node exists then the pointer to it is returned instead of creating a new node. This creates a DAG, which reduces the space and time requirement.

**1.1.3 Postfix Notation** Postfix notation is a linear representation of a syntax tree. This can be written by traversing the tree in the post order form. The edges in a syntax tree do not appear explicitly in postfix notation; only the nodes are listed. The order is followed by listing the parent node immediately after listing its left sub tree and its right sub tree. In postfix notation, the operators are placed after the operands.

**1.1.4 Three Address Code** Three address code is a linear representation of a syntax tree or a DAG in which explicit names correspond to the interior nodes of the graph. Three address code is a sequence of statements of the form  $A = B \text{ OP } C$  where A, B and C are the names of variables, constants or the temporary variables generated by the compiler. OP is any arithmetic operation or logical operation applied on the operands B and C. The name reflects that there are at most three variables where two are operands and one is for the result. In three address statement, only one operator is permitted; if the expression is large, then break it into a sequence of sub expressions using the BODMAS rules of arithmetic and store the intermediate results in newly created temporary variables. For example, consider the expression  $a + b * c$ ; this expression is expressed as follows:  $T1 = b * c$   $T2 = a + T1$  Here T1 and T2 are compiler-generated temporary names. This simple representation of a complex expression in three address code makes the task of optimizer and code generator simple. It is also easy to rearrange the sequence for efficient code generation. Three address code for the statement  $a = b^* - (c - d) + b^* - (c - d)$  is as follows:  $T1 = c - d$   $T2 = -T1$   $T3 = b * T2$   $T4 = c - d$   $T5 = -T4$   $T6 = b * T5$   $T7 = T3 + T6$   $a = T7$  The code can also be written for DAG as follows:  $T1 = c - d$   $T2 = -T1$   $T3 = b * T2$   $T4 = T3 + T3$   $a = T4$

**Types of Three Address Statements** For expressing the different programming constructs, the three address statements can be written in different standard formats and these formats are used based on the expression. Some of them are as follows:

- Assignment statements with binary operator. They are of the form  $A := B \text{ op } C$  where op is a binary arithmetic or logical operation.
- Assignment statements with unary operator. They are of the form  $A := op B$  where op is a unary operation like unary plus, unary minus, shift, etc.
- Copy statements. They are of the form  $A := B$  where the value of B is assigned to variable A.
- Unconditional Jumps such as goto L: The label L with three address statement is the next statement number to be executed.
- Conditional Jumps such as if X relop Y goto L. If the condition is satisfied, then this instruction applies a relational operator ( $<=$ ,  $>=$ ,  $=$ ) to X and Y and executes the statement with label L else the statement following if X relop Y goto L is executed.
- Functional calls: The functional calls are written as a sequence of param A, call fun,n, and returnB statements, where A indicates one of the input argument in n arguments to be passed to the function fun that returns B. The return statement is optional.

**2. Declarations** As the sequence of declarations in a procedure or block is examined, we can lay out storage for names local to the procedure. For each local name, we create a symbol-table entry with information like the type and the relative address of the storage for the name. The relative address consists of an offset from the base of the static data area or the field for local data in an activation record.

**Declarations in a Procedure:** The syntax of languages such as C, Pascal and Fortran, allows all the declarations in a single procedure to be processed as a group. In this case, a global variable, say offset, can keep track of the next available relative address. In the translation scheme shown below:

- Non-terminal P generates a sequence of declarations of the form  $id : T$ .
- Before the first declaration is considered, offset is set to 0. As each new name is seen, that name is entered in the symbol table with offset equal to the current value of offset, and offset is incremented by the width of the data object denoted by that name.
- The procedure  $\text{enter( name, type, offset )}$  creates a symbol-table entry for name, gives its type and relative address offset in its data area.
- Attribute type represents a type expression constructed from the basic types integer and real by applying the type constructors pointer and array. If type expressions are represented by graphs, then attribute type might be a pointer to the node representing a type expression.
- The width of an array is obtained by multiplying the width of each element by the number of elements in the array. The width of

each pointer is assumed to be 4. T array [ num ] of T1 T.type = array(num.val, T1 .type) T.width = num.val x T1 .width T T1 T.type = pointer(T1 .type) T.width = 4 This is the continuation of the example in the previous slide Keeping Track of Scope Information: When a nested procedure is seen, processing of declarations in the enclosing procedure is temporarily suspended. This approach will be illustrated by adding semantic rules to the following language: P ->D D ->D ; D | id : T | proc id ; D ; S One possible implementation of a symbol table is a linked list of entries for names. A new symbol table is created when a procedure declaration D proc id D1;S is seen, and entries for the declarations in D1 are created in the new table. The new table points back to the symbol table of the enclosing procedure; the name represented by id itself is local to the enclosing procedure. The only change from the treatment of variable declarations is that the procedure enter is told which symbol table to make an entry in. For example, consider the symbol tables for procedures readarray, exchange, and quicksort pointing back to that for the containing procedure sort, consisting of the entire program. Since partition is declared within quicksort, its table points to that of quicksort. Whenever a procedure declaration Dproc id ; D1 ; S is processed, a new symbol table with a pointer to the symbol table of the enclosing procedure in its header is created and the entries for declarations in D1 are created in the new symbol table. The name represented by id is local to the enclosing procedure and is hence entered into the symbol table of the enclosing procedure. Example Program sort: var a: array[1..n] of integer; X:integer; Procedure readarray; var i: integer ..... Procedure exchange(I,j:integer); ..... Procedure quicksort(m,n : integer); Var k,v:integer; Function partition(x,y:integer):integer; Var l,j:integer; ..... ..... Begin{main} ..... end For the above procedures, entries for x, a and b quicksort are created in the symbol table of sort. A pointer pointing to the symbol table of quicksort is also entered. Similarly, entries for k,v and partition are created in the symbol table of quicksort. The headers of the symbol tables of quicksort and partition have pointers pointing to sort and quicksort respectively This structure follows from the example in the previous slide. Creating symbol table mkttable (previous) create a new symbol table and return a pointer to the new table. The argument previous points to the enclosing procedure enter (table, name, type, offset) creates a new entry addwidth (table, width) records cumulative width of all the entries in a table enterproc (table, name, newtable) creates a new entry for procedure name. newtable points to the symbol table of the new procedure The following operations are designed : • mkttable(previous): creates a new symbol table and returns a pointer to this table. previous is pointer to the symbol table of parent procedure. • enter(table,name,type,offset): creates a new entry for name in the symbol table pointed to by table . • addwidth(table,width): records cumulative width of entries of a table in its header. • enterproc(table,name ,newtable): creates an entry for procedure name in the symbol table pointed to by table . newtable is a pointer to symbol table for name . Creating symbol table. P {t=mkttable(nil); push(t,tblptr); push(0,offset)} D {addwidth(top(tblptr),top(offset)); pop(tblptr); pop(offset)} D D ; D Table 4.2:Symbol table The symbol tables are created using two stacks: tblptr to hold pointers to symbol tables of the enclosing procedures and offset whose top element is the next available relative address for a local of the current procedure. Declarations in nested procedures can be processed by the syntax directed definitions given below. Note that they are basically same as those given above but we have separately dealt with the epsilon productions. 3. Assignment Statements Suppose that the context in which an assignment appears is given by the following grammar. P M D M ε D D ; D | id : T | proc id ; N D ; S N ε Non-terminal P becomes the new start symbol when these productions are added to those in the translation scheme shown below. Translation scheme to produce three-address code for assignments S id := E { p := lookup ( id.name); if p ≠ nil then emit( p ' := ' E.place) else error } E E1 + E2 { E.place := newtemp; emit( E.place ' := ' E1.place ' + ' E2.place ) } E E1 \* E2 { E.place := newtemp; emit( E.place ' := ' E1.place ' \* ' E2.place ) } E - E1 { E.place := newtemp; emit ( E.place ' := ' 'uminus' E1.place ) } E ( E1 ) { E.place := E1.place } E id { p := lookup ( id.name); if p ≠ nil then E.place := p else error } Reusing Temporary Names The temporaries used to hold intermediate values in expression calculations tend to clutter up the symbol table, and space has to be allocated to hold their values. Temporaries can be reused by changing newtemp. The code generated by the rules for E.E1 + E2 has the general form: evaluate E1 into t1 evaluate E2 into t2 t := t1 + t2 The lifetimes of these temporaries are nested like matching pairs of balanced parentheses. Keep a count c , initialized to zero. Whenever a temporary name is used as an operand, decrement c by 1. Whenever a new temporary name is generated, use \$c and increase c by 1. 3.1 Addressing Array Elements Arrays are stored in a block of consecutive locations assume width of each element is w ith element of array A begins in location base + (i - low) x w where base is relative address of A[low]the expression is equivalent to i x w + (base-low x w) i x w + const Elements of an array are stored in a block of consecutive locations. For a single dimensional array, if low is the lower bound of the index and base is the relative address of the storage allocated to the array i.e., the relative address of A[low], then the i th Elements of an array are

stored in a block of consecutive locations For a single dimensional array, if low is the lower bound of the index and base is the relative address of the storage allocated to the array i.e., the relative address of  $A[low]$ , then the  $i$  th elements begins at the location:  $\text{base} + (i - \text{low}) * w$ . This expression can be reorganized as  $i * w + (\text{base} - \text{low}) * w$ . The sub-expression  $\text{base} - \text{low}$  is calculated and stored in the symbol table at compile time when the array declaration is processed, so that the relative address of  $A[i]$  can be obtained by just adding  $i * w$  to it. 3.2 2-dimensional array storage can be either row major or column major in case of 2-D array stored in row major form address of  $A[i_1, i_2]$  can be calculated as  $\text{base} + ((i_1 - \text{low}_1) * n_2 + i_2 - \text{low}_2) * w$  where  $n_2 = \text{high}_2 - \text{low}_2 + 1$  rewriting the expression gives  $((i_1 * n_2) + i_2) * w + (\text{base} - ((\text{low}_1 * n_2) + \text{low}_2) * w)$   $((i_1 * n_2) + i_2) * w + \text{constant}$  this can be generalized for  $A[i_1, i_2, \dots, i_k]$  Similarly, for a row major two dimensional array the address of  $A[i][j]$  can be calculated by the formula :  $\text{base} + ((i * \text{low}_i) * n_2 + j - \text{low}_j) * w$  where  $\text{low}_i$  and  $\text{low}_j$  are lower values of  $i$  and  $j$  and  $n_2$  is number of values  $j$  can take i.e.  $n_2 = \text{high}_2 - \text{low}_2 + 1$ . This can again be written as :  $((i * n_2) + j) * w + (\text{base} - ((\text{low}_i * n_2) + \text{low}_j) * w)$  and the second term can be calculated at compile time. In the same manner, the expression for the location of an element in column major two-dimensional array can be obtained. This addressing can be generalized to multidimensional arrays. 4. Boolean Expressions Boolean expressions have two primary purposes. They are used to compute logical values, but more often they are used as conditional expressions in statements that alter the flow of control, such as if-then-else, or while-do statements. Boolean expressions are composed of the boolean operators ( and, or, and not ) applied to elements that are boolean variables or relational expressions. Relational expressions are of the form  $E_1 \text{ relop } E_2$ , where  $E_1$  and  $E_2$  are arithmetic expressions. Here we consider boolean expressions generated by the following grammar:  $E : E \text{ or } E \mid E \text{ and } E \mid \text{not } E \mid ( E ) \mid \text{id} \text{ relop id} \mid \text{true} \mid \text{false}$ 

**Methods of Translating Boolean Expressions:** There are two principal methods of representing the value of a boolean expression. They are: To encode true and false numerically and to evaluate a boolean expression analogously to an arithmetic expression. Often, 1 is used to denote true and 0 to denote false. To implement boolean expressions by flow of control, that is, representing the value of a boolean expression by a position reached in a program. This method is particularly convenient in implementing the boolean expressions in flow-of-control statements, such as the if-then and while-do statements. 4.1 Numerical representation a or b and not c t1 = not c t2 = b and t1 t3 = a or t2 relational expression a < b is equivalent to if a < b then 1 else 0 1. if a < b goto 4. 2. t = 0 3. goto 5 4. t = 1 Consider the implementation of Boolean expressions using 1 to denote true and 0 to denote false. Expressions are evaluated in a manner similar to arithmetic expressions. For example, the three address code for a or b and not c is: t1 = not c t2 = b and t1 t3 = a or t2 4.2 Syntax directed translation of boolean expressions E E 1 or E2 E.place := newtmp emit(E.place ':=' E 1 .place 'or' E2 .place) E E1 and E2 E.place:= newtmp emit(E.place ':=' E 1 .place 'and' E2 .place) E not E1 E.place := newtmp emit(E.place ':=' 'not' E1 .place) E (E1 ) E.place = E1 .place The above written translation scheme produces three address code for Boolean expressions. It is continued to the next page. 4.3 Syntax directed translation of boolean expressions E id1 relop id2 E.place := newtmp emit(if id1.place relop id2.place goto nextstat+3) emit(E.place = 0) emit(goto nextstat+2) emit(E.place = 1) E true E.place := newtmp emit(E.place = '1') E false E.place := newtmp emit(E.place = '0') In the above scheme, nextstat gives the index of the next three address code in the output sequence and emit increments nextstat after producing each three address statement. Example: Code for a < b or c < d and e < f 100: if a < b goto 103 if e < f goto 111 101: t1 = 0 109: t3 = 0 102: goto 104 110: goto 112 103: t1 = 1 111: t3 = 1 104: 112: if c < d goto 107 t4 = t2 and t3 105: t2 = 0 113: t5 = t1 or t4 106: goto 108 107: t2 = 1 108: Table 4.3:Three address code for above example A relational expression a < b is equivalent to the conditional statement if a < b then 1 else 0 and three address code for this expression is: 100: if a < b goto 103. 101: t = 0 102: goto 104 103: t = 1 104: It is continued from 104 in the same manner as the above written block. 5. Case Statement switch expression begin case value: statement case value: statement ... case value: statement default: statement end evaluate the expression find which value in the list of cases is the same as the value of the expression Default value matches the expression if none of the values explicitly mentioned in the cases matches the expression execute the statement associated with the value found. There is a selector expression, which is to be evaluated, followed by n constant values that the expression can take. This may also include a default value which always matches the expression if no other value does. The intended translation of a switch case code to:
 

- evaluate the expression
- find which value in the list of cases is the same as the value of the expression.
- Default value matches the expression if none of the values explicitly mentioned in the cases matches the expression. execute the statement associated with the value found
- Most machines provide instruction in hardware such that case instruction can be implemented easily. So, case is treated differently and not as a combination of if-then statements. Translation code to

evaluate E into t code to evaluate E into t if  $t \leftrightarrow V_1$  goto L1 goto test code for S1 L1: code for S1 goto next goto next L1 if  $t \leftrightarrow V_2$  goto L2 L2: code for S2 code for S2 goto next goto next .. L2: .. Ln: code for Sn Ln-2 if  $t \leftrightarrow V_{n-1}$  goto Ln-1 goto next code for Sn-1 test: if  $t = V_1$  goto L1 goto next if  $t = V_2$  goto L2 Ln-1: code for Sn .. next: if  $t = V_{n-1}$  goto Ln-1 goto Ln next:  
 Table 4.4: Translation table Efficient for n-way branch There are two ways of implementing switch-case statements, both given above. The above two implementations are equivalent except that in the first case all the jumps are short jumps while in the second case they are long jumps. However, many machines provide the n-way branch which is a hardware instruction. Exploiting this instruction is much easier in the second implementation while it is almost impossible in the first one. So, if hardware has this instruction the second method is much more efficient.

### 6. Backpatching

The easiest way to implement the syntax-directed definitions for boolean expressions is to use two passes. First, construct a syntax tree for the input, and then walk the tree in depth-first order, computing the translations. The main problem with generating code for boolean expressions and flow-of-control statements in a single pass is that during one single pass we may not know the labels that control must go to at the time the jump statements are generated. Hence, series of branching statements with the targets of the jumps left unspecified is generated. Each statement will be put on a list of goto statements whose labels will be filled in when the proper label can be determined. We call this subsequent filling in of labels backpatching.

To manipulate lists of labels, we use three functions:

- makelist(i) creates a new list containing only i, an index into the array of quadruples; makelist returns a pointer to the list it has made.
- merge(p1,p2) concatenates the lists pointed to by p1 and p2, and returns a pointer to the concatenated list.
- backpatch(p,i) inserts i as the target label for each of the statements on the list pointed to by p.

Boolean Expressions E E1 or M E2 | E1 and M E2 | not E1 | (E 1) | id 1 relop id 2 | true | false M ? ε Synthesized attributes truelist and falselist of non-terminal E are used to generate jumping code for boolean expressions. Incomplete jumps with unfilled labels are placed on lists pointed to by E.truelist and E.falselist.

Consider production E E1 and M E2. If E1 is false, then E is also false, so the statements on E1.falselist become part of E.falselist. If E1 is true, then we must next test E2, so the target for the statements E1.truelist must be the beginning of the code generated for E2. This target is obtained using marker non-terminal M. Attribute M.quad records the number of the first statement of E2.code. With the production M ε we associate the semantic action { M.quad := nextquad } The variable nextquad holds the index of the next quadruple to follow. This value will be backpatched onto the E1.truelist when we have seen the remainder of the production E E1 and M E2. The translation scheme is as follows:

- (1) E E1 or M E2 { backpatch ( E1.falselist, M.quad); E.truelist := merge( E1.truelist, E2.truelist); E.falselist := E2.falselist }
- (2) E E1 and M E2 { backpatch ( E1.truelist, M.quad); E.truelist := E2.truelist; E.falselist := merge(E1.falselist, E2.falselist) }
- (3) E not E1 { E.truelist := E1.falselist; E.falselist := E1.truelist; }
- (4) E ( E1 ) { E.truelist := E1.truelist; E.falselist := E1.falselist; }
- (5) E id1 relop id2 { E.truelist := makelist (nextquad); E.falselist := makelist(nextquad + 1); emit('if' id1.place relop.op id2.place 'goto\_') emit('goto\_') }
- (6) E true { E.truelist := makelist(nextquad); emit('goto\_') }
- (7) E false { E.falselist := makelist(nextquad); emit('goto\_') }
- (8) M ε { M.quad := nextquad }

### 7. Procedure Calls

The procedure is such an important and frequently used programming construct that it is imperative for a compiler to generate good code for procedure calls and returns. The run-timeroutines that handle procedure argument passing, calls and returns are part of the run-time support package. Let us consider a grammar for a simple procedure call statement

- (1) S call id ( Elist )
- (2) Elist Elist , E
- (3) Elist E

**Calling Sequences:** The translation for a call includes a calling sequence, a sequence of actions taken on entryto and exit from each procedure. The falling are the actions that take place in a calling sequence: When a procedure call occurs, space must be allocated for the activation record of the called procedure. The arguments of the called procedure must be evaluated and made available to the called procedure in a known place. Environment pointers must be established to enable the called procedure to access data in enclosing blocks. The state of the calling procedure must be saved so it can resume execution after the call. Also saved in a known place is the return address, the location to which the called routine must transfer after it is finished. Finally a jump to the beginning of the code for the called procedure must be generated.

For example, consider the following syntax-directed translation.

- (1) S call id ( Elist ) { for each item p on queue do emit ('param' p ); emit ('call' id.place) }
- (2) Elist Elist , E { append E.place to the end of queue }
- (3) Elist E { initialize queue to contain only E.place }

Here, the code for S is the code for Elist, which evaluates the arguments, followed by a param p statement for each argument, followed by a call statement. Queue is emptied and then gets a single pointer to the symbol table location for the name that denotes the value of E.

### 8. CODE GENERATION

The final phase in compiler model is the code generator. It takes as input an intermediaterepresentation of the source program and produces as output an equivalent target program. The code generation techniques presented below can be used whether or not an optimizing

phase occurs before code generation. Figure 4.2: Code Generation in Compiler 8.1 Issues In The Design Of A Code Generator The following issues arise during the code generation phase: 1. Input to code generator 2. Target program 3. Memory management 4. Instruction selection 5. Register allocation 6. Evaluation order Input to code generator: The input to the code generation consists of the intermediate representation of the source program produced by front end, together with information in the symbol table to determine run-time addresses of the data objects denoted by the names in the intermediate representation. Intermediate representation can be: a. Linear representation such as postfix notation b. Three address representation such as quadruples c. Virtual machine representation such as stack machine code d. Graphical representations such as syntax trees and dags. Prior to code generation, the front end must be scanned, parsed and translated into intermediate representation along with necessary type checking. Therefore, input to code generation is assumed to be error-free. Target program: The output of the code generator is the target program. The output may be: a. Absolute machine language It can be placed in a fixed memory location and can be executed immediately. b. Reloadable machine language It allows subprograms to be compiled separately. c. Assembly language Code generation is made easier. Memory management: • Names in the source program are mapped to addresses of data objects in run-time memory by the front end and code generator. • It makes use of symbol table, that is, a name in a three-address statement refers to a symbol-table entry for the name. • Labels in three-address statements have to be converted to addresses of instructions. Instruction selection: • The instructions of target machine should be complete and uniform. • Instruction speeds and machine idioms are important factors when efficiency of target program is considered. • The quality of the generated code is determined by its speed and size. Register allocation Instructions involving register operands are shorter and faster than those involving operands in memory. The use of registers is subdivided into two sub problems: Register allocation – the set of variables that will reside in registers at a point in the program is selected. Register assignment – the specific register that a variable will reside in is picked. Certain machine requires even-odd register pairs for some operands and results. For example, consider the division instruction of the form: D x, y Where, x – dividend even register in even/odd register pair y – Divisor even register holds the remainder odd register holds the quotient Evaluation order The order in which the computations are performed can affect the efficiency of the target code. Some computation orders require fewer registers to hold intermediate results than others. 9. Basic Blocks And Flow Graphs 9.1 Basic Blocks A basic block is a sequence of consecutive statements in which flow of control enters at the beginning and leaves at the end without any halt or possibility of branching except at the end. The following sequence of three-address statements forms a basic block: t1: = a \* a t2: = a \* b t3: = 2 \* t2 t4: = t1 + t3 t5: = b \* b t6: = t4 + t5 Basic Block Construction: Algorithm: Partition into basic blocks Input: A sequence of three-address statements Output: A list of basic blocks with each three-address statement in exactly one block Method: 1. We first determine the set of leaders, the first statements of basic blocks. The rules we use are of the following: a. The first statement is a leader. b. Any statement that is the target of a conditional or unconditional goto is a leader. c. Any statement that immediately follows a goto or conditional goto statement is a leader. 2. For each leader, its basic block consists of the leader and all statements up to but not including the next leader or the end of the program. Consider the following source code for dot product of two vectors a and b of length 20 begin prod :=0; i:=1; do begin prod :=prod+a[i] \* b[i]; i :=i+1; end while i <= 20 end The three-address code for the above source program is given as: (1) prod := 0 (2) i := 1 (3) t1 := 4\* i (4) t2 := a[t1] /\*compute a[i] \*/ (5) t3 := 4\* i (6) t4 := b[t3] /\*compute b[i] \*/ (7) t5 := t2\*t4 (8) t6 := prod+t5 (9) prod := t6 (10) t7 := i+1 (11) i := t7 (12) if i<=20 goto (3) Transformations on Basic Blocks: A number of transformations can be applied to a basic block without changing the set of expressions computed by the block. Two important classes of transformation are: • Structure-preserving transformations • Algebraic transformations 1. Structure preserving transformations: a) Common sub expression elimination: a := b + c a := b + c b := a - d b := a - d c := b + c c := b + c d := a - d d := b Since the second and fourth expressions compute the same expression, the basic block can be transformed as above. b) Dead-code elimination: Suppose x is dead, that is, never subsequently used, at the point where the statement x := y + z appears in a basic block. Then this statement may be safely removed without changing the value of the basic block. c) Renaming temporary variables: A statement t := b + c ( t is a temporary ) can be changed to u := b + c ( u is a new temporary ) and all uses of this instance of t can be changed to u without changing the value of the basic block. Such a block is called a normal-form block. d) Interchange of statements: Suppose a block has the following two adjacent statements: t1: = b + c t2: = x + y We can interchange the two statements without affecting the value of the block if and only if neither x nor y is t1 and neither b nor c is t2. 2. Algebraic transformations: Algebraic transformations can be used to change the set of expressions

computed by a basic block into an algebraically equivalent set. Examples: i)  $x := x + 0$  or  $x := x * 1$  can be eliminated from a basic block without changing the set of expressions it computes. ii) The exponential statement  $x := y ** 2$  can be replaced by  $x := y * y$ . 9.2 Flow Graphs Flow graph is a directed graph containing the flow-of-control information for the set of basic blocks making up a program. The nodes of the flow graph are basic blocks. It has a distinguished initial node. Loops A loop is a collection of nodes in a flow graph such that 1. All nodes in the collection are strongly connected. 2. The collection of nodes has a unique entry. A loop that contains no other loops is called an inner loop. 9.3 Next-Use Information If the name in a register is no longer needed, then we remove the name from the register and the register can be used to store some other names Input: Basic block B of three-address statements Output: At each statement  $i: x = y \text{ op } z$ , we attach to i the liveness and next-uses of x, y and z. Method: We start at the last statement of B and scan backwards. 1. Attach to statement i the information currently found in the symbol table regarding the next-use and liveness of x, y and z. 2. In the symbol table, set x to "not live" and "no next use". 3. In the symbol table, set y and z to "live", and next-uses of y and z to i. 10. The DAG Representation For Basic Blocks A DAG for a basic block is a directed acyclic graph with the following labels on nodes: 1. Leaves are labeled by unique identifiers, either variable names or constants. 2. Interior nodes are labeled by an operator symbol. 3. Nodes are also optionally given a sequence of identifiers for labels to store the computed values. DAGs are useful data structures for implementing transformations on basic blocks. It gives a picture of how the value computed by a statement is used in subsequent statements. It provides a good way of determining common sub-expressions. Algorithm for construction of DAG Input: A basic block Output: A DAG for the basic block containing the following information: 1. A label for each node. For leaves, the label is an identifier. For interior nodes, an operator symbol. 2. For each node a list of attached identifiers to hold the computed values. Case (i)  $x := y \text{ OP } z$  Case (ii)  $x := \text{OP } y$  Case (iii)  $x := y$  Method: Step 1: If y is undefined then create node(y). If z is undefined, create node(z) for case(i). Step 2: For the case(i), create a node(OP) whose left child is node(y) and right child is node(z). (Checking for common sub expression). Let n be this node. For case(ii), determine whether there is node(OP) with one child node(y). If not create such a node. For case(iii), node n will be node(y). Step 3: Delete x from the list of identifiers for node(x). Append x to the list of attached identifiers for the node n found in step 2 and set node(x) to n. Application of DAGs: 1. We can automatically detect common sub expressions. 2. We can determine which identifiers have their values used in the block. 3. We can determine which statements compute values that could be used outside the block. 11. Peephole Optimization Target code often contains redundant instructions and suboptimal constructs. Examine a short sequence of target instruction (peephole) and replace by a shorter or faster sequence peephole is a small moving window on the target systems. A statement-by-statement code-generation strategy often produces target code that contains redundant instructions and suboptimal constructs. A simple but effective technique for locally improving the target code is peephole optimization, a method for trying to improve the performance of the target program by examining a short sequence of target instructions (called the peephole) and replacing these instructions by a shorter or faster sequence, whenever possible. The peephole is a small, moving window on the target program. The code in the peephole need not be contiguous, although some implementations do require this. Peephole optimization examples. Redundant loads and stores Consider the code sequence Move R0 , a Move a, R0 Instruction 2 can always be removed if it does not have a label. Now, we will give some examples of program transformations that are characteristic of peephole optimization: Redundant loads and stores: If we see the instruction sequence Move R0 , a Move a, R0 We can delete instruction (2) because whenever (2) is executed, (1) will ensure that the value of a is already in register R0. Note that if (2) has a label, we could not be sure that (1) was always executed immediately before (2) and so we could not remove (2). Peephole optimization examples. Unreachable code Consider the following code # define debug 0 If(debug) { Print debugging info } This may be translated as If debug =1 goto L1 Goto L2 L1:print debugging info L2: Eliminate jump over jumps If debug > 1 goto L2 Print debugging information L2: Another opportunity for peephole optimization is the removal of unreachable instructions. 12. Generating Code from DAGs The advantage of generating code for a basic block from its dag representation is that, from a dag we can easily see how to rearrange the order of the final computation sequence than we can starting from a linear sequence of three-address statements or quadruples. Rearranging the order The order in which computations are done can affect the cost of resulting object code. For example, consider the following basic block: t1:= a + b t2:= c + d t3:= e - t2 t4:= t1 - t3 Generated code sequence for basic block: MOV a , R0 ADD b , R0 MOV c , R1 ADD d , R1 MOV R0 , t1 MOV e , R0 SUB R1 , R0 MOV t1 , R1 SUB R0 , R1 MOV R1 , t4 Rearranged basic block: Now t1 occurs immediately before t4. t2:= c + d t3:= e - t2 t1:= a + b t4:= t1 - t3 Revised code sequence: MOV c , R0 ADD d , R0 MOV a

, R0 SUB R0 , R1 MOV a , R0 ADD b , R0 SUB R1 , R0 MOV R0 , t4 In this order, two instructions MOV R0 , t1 and MOV t1 , R1 have been saved. A Heuristic ordering for DAGS The heuristic ordering algorithm attempts to make the evaluation of a node immediately follow the evaluation of its leftmost argument. The algorithm shown below produces the ordering in reverse. Algorithm: 1) while unlisted interior nodes remain do begin 2) select an unlisted node n, all of whose parents have been listed; 3) list n; 4) while the leftmost child m of n has no unlisted parents and is not a leaf do begin 5) list m; 6) n := m end end Code generation phase generates the code based on the numbering assigned to each node T. All the registers available are arranged as a stack to maintain the order of the lower register at the top. This makes an assumption that the required number of registers cannot exceed the number of available registers. In some cases, we may need to spill the intermediate result of a node to memory. This algorithm also does not take advantage of the commutative and associative properties of operators to rearrange the expression tree. It first checks whether the node T is a leaf node; if yes, it generates a load instruction corresponding to it as load top (), T. If the node T is an internal node, then it checks the left l and right r sub tree for the number assigned. There are three possible values, the number on the right is 0 or greater than or less than the number on the left. If it is 0 then call the generate () function with left sub tree l and then generate instruction op top (),r. If the numbering on the left is greater than or equal to right, then call generate () with left sub tree, get new register by popping the top, call generate () with right sub tree, generate new instruction for OP R, top (), and push back the used register on to the stack. UNIT V Code Optimization :Introduction to Code optimization,Sources of Optimization of Basic Blocks, Loops in Flowgraphs, Deadcode Elimination, Loop Optimization, Introduction to Global Data Flow Analysis, Code Improving Transformations, Data Flow Analysis of Structure Flowgraph, Symbolic Debugging of Optimized Code. 1. Code Optimization Code optimization phase is an optional phase in the phases of a compiler, which is either before the code generation phase or after the code generation phase. This chapter focuses on the types of optimizer and the techniques available for optimizing. The code produced by the straight forward compiling algorithms can often be made to run faster or take less space, or both. This improvement is achieved by program transformations that are traditionally called optimizations. Compilers that apply code-improving transformations are called optimizing compilers. Optimizations are classified into two categories. They are Machine independent optimizations: Machine dependent optimizations: 1.1 Machine independent optimizations: Machine independent optimizations are program transformations that improve the target code without taking into consideration any properties of the target machine. 1.1 Machine dependent optimizations: Machine dependent optimizations are based on register allocation and utilization of special machineinstruction sequences. 1.1 The criteria for code improvement transformations: Simply stated, the best program transformations are those that yield the most benefit for the least effort. The transformation must preserve the meaning of programs. That is, the optimization must not change the output produced by a program for a given input, or cause an error such as division by zero, that was not present in the original source program. At all times we take the “safe” approach of missing an opportunity to apply a transformation rather than risk changing what the program does. A transformation must, on the average, speed up programs by a measurable amount. We are also interested in reducing the size of the compiled code although the size of the code has less importance than it once had. Not every transformation succeeds in improving every program, occasionally an “optimization” may slow down a program slightly. The transformation must be worth the effort. It does not make sense for a compiler writer to expend the intellectual effort to implement a code improving transformation and to have the compiler expend the additional time compiling source programs if this effort is not repaid when the target programs are executed. “Peephole” transformations of this kind are simple enough and beneficial enough to be included in any compiler. 2. Principal Sources Of Optimisation A transformation of a program is called local if it can be performed by looking only at the statements in a basic block; otherwise, it is called global. Many transformations can be performed at both the local and global levels. Local transformations are usually performed first. 2.1 Function-Preserving Transformations There are a number of ways in which a compiler can improve a program without changing the function it computes. The transformations • Common sub expression elimination, • Copy propagation, • Dead-code elimination, and • Constant folding Are common examples of such function-preserving transformations? The other transformations come up primarily when global optimizations are performed. Frequently, a program will include several calculations of the same value, such as an offset in an array. Some of the duplicate calculations cannot be avoided by the programmer because they lie below the level of detail accessible within the source language. 2.3 Common Sub expressions elimination An occurrence of an expression E is called a common sub-expression if E was previously computed, and the values of variables in E have not changed since the

previous computation. We can avoid re-computing the expression if we can use the previously computed value. For example  $t1 := 4*i$   $t2 := a$  [t1]  $t3 := 4*j$   $t4 := 4*i$   $t5 := n$   $t6 := b$  [t4] + t5 The above code can be optimized using the common sub-expression elimination as  $t1 := 4*i$   $t2 := a$  [t1]  $t3 := 4*j$   $t5 := n$   $t6 := b$  [t1] + t5 The common sub expression  $t4 := 4*i$  is eliminated as its computation is already in t1. And value of i is not been changed from definition to use.

### 2.4 Copy Propagation:

Assignments of the form  $f := g$  called copy statements, or copies for short. The idea behind the copy propagation transformation is to use g for f, whenever possible after the copy statement  $f := g$ . Copy propagation means use of one variable instead of another. This may not appear to be an improvement, but as we shall see it gives us an opportunity to eliminate x. For example:  $x=Pi; \dots A=x*r^r;$  The optimization using copy propagation can be done as follows:  $A=Pi*r^r;$  Here the variable x is eliminated.

### 2.5 Dead-Code Eliminations:

A variable is live at a point in a program if its value can be used subsequently; otherwise, it is dead at that point. A related idea is dead or useless code, statements that compute values that never get used. While the programmer is unlikely to introduce any dead code intentionally, it may appear as the result of previous transformations. An optimization can be done by eliminating dead code. Example:  $i=0; if(i=1) \{ a=b+5; \}$  Here, 'if' statement is dead code because this condition will never get satisfied.

### 2.6 Constant folding:

We can eliminate both the test and printing from the object code. More generally, deducing at compile time that the value of an expression is a constant and using the constant instead is known as constant folding. One advantage of copy propagation is that it often turns the copy statement into dead code. For example,  $a=3.14157/2$  can be replaced by  $a=1.570$  thereby eliminating a division operation.

### 2.7 Loop Optimizations:

We now give a brief introduction to a very important place for optimizations, namely loops, especially the inner loops where programs tend to spend the bulk of their time. The running time of a program may be improved if we decrease the number of instructions in an inner loop, even if we increase the amount of code outside that loop. Three techniques are important for loop optimization: Code motion, which moves code outside a loop; Induction-variable elimination, which we apply to replace variables from inner loop. Reduction in strength, which replaces an expensive operation by a cheaper one, such as a multiplication by an addition.

### 2.8 Code Motion:

An important modification that decreases the amount of code in a loop is code motion. This transformation takes an expression that yields the same result independent of the number of times a loop is executed ( a loop-invariant computation) and places the expression before the loop. Note that the notion "before the loop" assumes the existence of an entry for the loop. For example, evaluation of limit-2 is a loop-invariant computation in the following while-statement: `while (i <= limit-2) /* statement does not change limit*/` Code motion will result in the equivalent of `t=limit-2; while (i<=t) /* statement does not change limit or t */`

### 2.9 Reduction in Strength:

Reduction in strength replaces expensive operations by equivalent cheaper ones on the target machine. Certain machine instructions are considerably cheaper than others and can often be used as special cases of more expensive operators. For example,  $x^2$  is invariably cheaper to implement as  $x*x$  than as a call to an exponentiation routine. Fixedpoint multiplication or division by a power of two is cheaper to implement as a shift. Floating-point division by a constant can be implemented as multiplication by a constant, which may be cheaper.

### 3. Optimization Of Basic Blocks

There are two types of basic block optimizations. They are:

- Structure-Preserving Transformations**
- Algebraic Transformations**

#### Structure-Preserving Transformations:

The primary Structure-Preserving Transformation on basic blocks are:

- Common sub-expression elimination
- Dead code elimination
- Renaming of temporary variables
- Interchange of two independent adjacent statements.

##### 3.1 Common sub-expression elimination:

Common sub expressions need not be computed over and over again. Instead they can be computed once and kept in store from where it's referenced when encountered again – of course providing the variable values in the expression still remain constant. Example:  $a := b+c$   $b := a-d$   $c := b+c$   $d := a-d$  The 2nd and 4th statements compute the same expression:  $b+c$  and  $a-d$  Basic block can be transformed to  $a := b+c$   $b := a-d$   $c := a$   $d := b$

##### 3.2 Dead code elimination:

It's possible that a large amount of dead (useless) code may exist in the program. This might be especially caused when introducing variables and procedures as part of construction or error-correction of a program – once declared and defined, one forgets to remove them in case they serve no purpose. Eliminating these will definitely optimize the code.

##### 3.3 Renaming of temporary variables:

A statement  $t := b+c$  where t is a temporary name can be changed to  $u := b+c$  where u is another temporary name, and change all uses of t to u. In this we can transform a basic block to its equivalent block called normal-form block.

##### Interchange of two independent adjacent statements:

Two statements  $t1:=b+c$   $t2:=x+y$  can be interchanged or reordered in its computation in the basic block when value of t1 does not affect the value of t2.

##### 3.4 Algebraic Transformations:

Algebraic identities represent another important class of optimizations on basic blocks. This includes simplifying expressions or replacing expensive operation by cheaper ones i.e. Reduction in strength. Another class

of related optimizations is constant folding. Here we evaluate constant expressions at compile time and replace the constant expressions by their values. Thus the expression  $2*3.14$  would be replaced by 6.28. The relational operators  $<=$ ,  $>=$ ,  $,$   $+$  and  $=$  sometimes generate unexpected common sub expressions. Associative laws may also be applied to expose common sub expressions. For example, if the source code has the assignments  $a := b+c$   $e := c+d+b$  the following intermediate code may be generated:  $a := b+c$   $t := c+d$   $e := t+b$  4. Loops In Flow Graph A graph representation of three-address statements, called a flow graph, is useful for understanding codegeneration algorithms, even if the graph is not explicitly constructed by a code-generation algorithm. Nodes in the flow graph represent computations, and the edges represent the flow of control. Dominators: In a flow graph, a node d dominates node n, if every path from initial node of the flow graph to n goes through d. This will be denoted by  $d \text{ dom } n$ . Every initial node dominates all the remaining nodes in the flow graph and the entry of a loop dominates all nodes in the loop. Similarly every node dominates itself. Most programs run as a loop in the system. It becomes necessary to optimize the loops in order to save CPU cycles and memory. Loops can be optimized by the following techniques: Invariant code: A fragment of code that resides in the loop and computes the same value at each iteration is called a loop-invariant code. This code can be moved out of the loop by saving it to be computed only once, rather than with each iteration. Induction analysis: A variable is called an induction variable if its value is altered within the loop by a loopinvariant value. Strength reduction: There are expressions that consume more CPU cycles, time, and memory. These expressions should be replaced with cheaper expressions without compromising the output of expression. For example, multiplication  $(x * 2)$  is expensive in terms of CPU cycles than  $(x << 1)$  and yields the same result. Natural Loop: • One application of dominator information is in determining the loops of a flow graph suitable for improvement. • The properties of loops are A loop must have a single entry point, called the header. This entry pointdominates all nodes in the loop, or it would not be the sole entry to the loop. • There must be at least one way to iterate the loop(i.e.)at least one path back to the header. • One way to find all the loops in a flow graph is to search for edges in the flow graph whose heads dominate their tails. If  $a \rightarrow b$  is an edge, b is the head and a is the tail. These types of edges are called as back edges. 5. Dead-Code Elimination Dead code is one or more than one code statements, which are: • Either never executed or unreachable, • Or if executed, their output is never used. Thus, dead code plays no role in any program operation and therefore it can simply be eliminated. Partially dead code There are some code statements who's computed values are used only under certain circumstances, i.e., sometimes the values are used and sometimes they are not. Such codes are known as partially dead-code. LOOP Figure 5.1: Dead–code elimination The above control flow graph depicts a chunk of program where variable 'a' is used to assign the output of expression ' $x * y$ '. Let us assume that the value assigned to 'a' is never used inside the loop. Immediately after the control leaves the loop, 'a' is assigned the value of variable 'z', which would be used later in the program. We conclude here that the assignment code of 'a' is never used anywhere, therefore it is eligible to be eliminated. Figure 5.2: Dead–code elimination Likewise, the picture above depicts that the conditional statement is always false, implying that the code, written in true case, will never be executed, hence it can be removed. 5.1 Partial Redundancy Redundant expressions are computed more than once in parallel path, without any change in operands. Whereas partial-redundant expressions are computed more than once in a path, without any change in operands. For example, .....  $a=x*y$   $A=z$ ;  $a=1$ ;  $b=10$  ..... . . . . . If  $\text{aid:} = E | S; S | \text{if } E \text{ then } S \text{ else } S | \text{do } S \text{ while } E E->\text{id} + \text{id} | \text{id}$  Expressions in this language are similar to those in the intermediate code, but the flow graphs for statements have restricted forms. We define a portion of a flow graph called a region to be a set of nodes N that includes a header, which dominates all other nodes in the region. All edges between nodes in N are in the region, except for some that enter the header. The portion of flow graph corresponding to a statement S is a region that obeys the further restriction that control can flow to just one outside block when it leaves the region. We say that the beginning points of the dummy blocks at the statement's region are the beginning and end points, respective equations are inductive, or syntax-directed, definition of the sets  $\text{in}[S]$ ,  $\text{out}[S]$ ,  $\text{gen}[S]$ , and  $\text{kill}[S]$  for all statements S.  $\text{gen}[S]$  is the set of definitions "generated" by S while  $\text{kill}[S]$  is the set of definitions that never reach the end of S. 7. CODE IMPROVING TRANSFORMATIONS • Algorithms for performing the code improving transformations rely on data-flow information. Here we consider common sub-expression elimination, copy propagation and transformations for moving loop invariant computations out of loops and for eliminating induction variables. • Global transformations are not substitute for local transformations; both must be performed. 7.1 Elimination of global common sub expressions: The available expressions data-flow problem discussed in the last section allows us to determine if an expression at point p in a flow graph is a common sub-expression. The following

algorithm formalizes the intuitive ideas presented for eliminating common sub expressions. ALGORITHM: Global common sub expression elimination. INPUT: A flow graph with available expression information. OUTPUT: A revised flow graph. METHOD: For every statement  $s$  of the form  $x := y + z$  such that  $y + z$  is available at the beginning of block and neither  $y$  nor  $z$  is defined prior to statement  $s$  in that block, do the following. To discover the evaluations of  $y + z$  that reach  $s$ 's block, we follow flow graph edges, searching backward from  $s$ 's block. However, we do not go through any block that evaluates  $y + z$ . The last evaluation of  $y + z$  in each block encountered is an evaluation of  $y + z$  that reaches  $s$ .

- Create new variable  $u$ .
- Replace each statement  $w := y + z$  found in (1) by  $w := u$ .
- Replace statement  $s$  by  $x := u$ .

• Some remarks about this algorithm are in order.

- The search in step(1) of the algorithm for the evaluations of  $y + z$  that reach statement  $s$  can also be formulated as a data-flow analysis problem. However, it does not make sense to solve it for all expressions  $y + z$  and all statements or blocks because too much irrelevant information is gathered. Not all changes made by algorithm are improvements. We might wish to limit the number of different evaluations reaching  $s$  found in step (1), probably to one. Algorithm will miss the fact that  $a * z$  and  $c * z$  must have the same value in a  $x := y + z$  vs  $b := a * z$   $d := c * z$  Because this simple approach to common sub expressions considers only the literal expressions themselves, rather than the values computed by expressions.

### 7.2 Copy propagation:

- Various algorithms introduce copy statements such as  $x := \text{copies}$  may also be generated directly by the intermediate code generator, although most of these involve temporaries local to one block and can be removed by the dag construction. We may substitute  $y$  for  $x$  in all these places, provided the following conditions are met every such use  $u$  of  $x$ .
- Statement  $s$  must be the only definition of  $x$  reaching  $u$ .
- On every path from  $s$  to  $u$  including paths that go through  $u$  several times, there are no assignments to  $y$ .
- Condition (1) can be checked using ud-changing information. We shall set up a new dataflow analysis problem in which  $\text{in}[B]$  is the set of copies  $s: x := y$  such that every path from initial node to the beginning of  $B$  contains the statement  $s$ , and subsequent to the last occurrence of  $s$ , there are no assignments to  $y$ .

ALGORITHM: Copy propagation. INPUT: a flow graph  $G$ , with ud-chains giving the definitions reaching block  $B$ , and with  $\text{c\_in}[B]$  representing the solution to equations that is the set of copies  $x := y$  that reach block  $B$  along every path, with no assignment to  $x$  or  $y$  following the last occurrence of  $x := y$  on the path. We also need ud-chains giving the uses of each definition. OUTPUT: A revised flow graph. METHOD: For each copy  $s : x := y$  do the following: Determine those uses of  $x$  that are reached by this definition of namely,  $s: x := y$ . Determine whether for every use of  $x$  found in (1),  $s$  is in  $\text{c\_in}[B]$ , where  $B$  is the block of this particular use, and moreover, no definitions of  $x$  or  $y$  occur prior to this use of  $x$  within  $B$ . Recall that if  $s$  is in  $\text{c\_in}[B]$  then  $s$  is the only definition of  $x$  that reaches  $B$ . If  $s$  meets the conditions of (2), then remove  $s$  and replace all uses of  $x$  found in (1) by  $y$ .

### 7.3 Detection of loop-invariant computations:

Ud-chains can be used to detect those computations in a loop that are loop-invariant, that is, whose value does not change as long as control stays within the loop. Loop is a region consisting of set of blocks with a header that dominates all the other blocks, so the only way to enter the loop is through the header. If an assignment  $x := y + z$  is at a position in the loop where all possible definitions of  $y$  and  $z$  are outside the loop, then  $y + z$  is loop-invariant because its value will be the same each time  $x := y + z$  is encountered. Having recognized that value of  $x$  will not change, consider  $v := x + w$ , where  $w$  could only have been defined outside the loop, then  $x + w$  is also loop-invariant.

ALGORITHM: Detection of loop-invariant computations. INPUT: A loop  $L$  consisting of a set of basic blocks, each block containing sequence of three-address statements. We assume ud-chains are available for the individual statements. OUTPUT: the set of three-address statements that compute the same value each time executed, from the time control enters the loop  $L$  until control next leaves  $L$ . METHOD: we shall give a rather informal specification of the algorithm, trusting that the principles will be clear.

- Mark “invariant” those statements whose operands are all either constant or have all their reaching definitions outside  $L$ .
- Repeat step (3) until at some repetition no new statements are marked “invariant”.
- Mark “invariant” all those statements not previously so marked all of whose operands either are constant, have all their reaching definitions outside  $L$ , or have exactly one reaching definition, and that definition is a statement in  $L$  marked invariant.

Performing code motion: Having found the invariant statements within a loop, we can apply to some of them an optimization known as code motion, in which the statements are moved to pre-header of the loop. The following three conditions ensure that code motion does not change what the program computes. Consider  $s: x := y + z$ . The block containing  $s$  dominates all exit nodes of the loop, where an exit of a loop is a node with a successor not in the loop. There is no other statement in the loop that assigns to  $x$ . Again, if  $x$  is a temporary assigned only once, this condition is surely satisfied and need not be changed. No use of  $x$  in the loop is reached by any definition of  $x$  other than  $s$ . This

condition too will be satisfied, normally, if  $x$  is temporary. **ALGORITHM:** Code motion. **INPUT:** A loop  $L$  with ud-chaining information and dominator information. **OUTPUT:** A revised version of the loop with a pre-header and some statements moved to the pre-header. **METHOD:**

- Use loop-invariant computation algorithm to find loop-invariant statements.
- For each statement  $s$  defining  $x$  found in step(1), check: i) That it is in a block that dominates all exits of  $L$ , ii) That  $x$  is not defined elsewhere in  $L$ , and iii) That all uses in  $L$  of  $x$  can only be reached by the definition of  $x$  in statement  $s$ .
- Move, in the order found by loop-invariant algorithm, each statement  $s$  found in (1) and meeting conditions (2i), (2ii), (2iii), to a newly created pre-header, provided any operands of  $s$  that are defined in loop  $L$  have previously had their definition statements moved to the pre-header. To understand why no change to what the program computes can occur, condition (2i) and (2ii) of this algorithm assure that the value of  $x$  computed at  $s$  must be the value of  $x$  after any exit block of  $L$ . When we move  $s$  to a pre-header,  $s$  will still be the definition of  $x$  that reaches the end of any exit block of  $L$ . Condition (2iii) assures that any uses of  $x$  within  $L$  did, and will continue to, use the value of  $x$  computed by  $s$ .

**8. Symbolic Debugging Scheme For Optimized Code:** Symbolic debuggers are system development tools that can accelerate the validation speed of behavioral specifications by allowing a user to interact with an executing code at the source level. In response to a user query, the debugger retrieves the value of a source variable in a manner consistent with respect to the source statement where execution has halted. Symbolic debuggers are system development tools that can accelerate the validation speed of behavioral specifications by allowing a user to interact with an executing code at the source level [Hen82]. Symbolic debugging must ensure that in response to a user inquiry, the debugger is able to retrieve and display the value of a source variable in a manner consistent with respect to a breakpoint in the source code. Code optimization techniques usually makes symbolic debugging harder. While code optimization techniques such as transformations must have the property that the optimized code is functionally equivalent to the unoptimized code, such optimization techniques may produce a different execution sequence from the source statements and alter the intermediate results. Debugging un-optimized rather than optimized code is not acceptable for several reasons, including:

- while an error in the un-optimized code is undetectable, it is detectable in the optimized code,
- optimizations may be necessary to execute a program due to hardware limitations, and
- a symbolic debugger for optimized code is often the only tool for finding errors in an optimization tool.

In a design-for-debugging (DfD) approach that enables retrieval of source values for a globally optimized behavioral specification. The goal of the DfD technique is to modify the original code in a pre-synthesis step such that every variable of the source code is controllable and observable in the optimized program.

Chameli Devi Group of Institutions Department of Artificial Intelligence and Data Science AD 603 (A) Data Mining and Warehousing B. Tech VI Semester Unit -III

..... Syllabus: Unit-III:  
Introduction to Data& Data Mining: Data Types, Quality of data, Data PreProcessing, Similarity measures, Summary statistics, Data distributions, Basic data mining Tasks, Data Mining v/s knowledge discovery in databases. Issues in Data mining, Introduction to Fuzzy sets and fuzzy logic.

..... Introduction to Data & Data Mining: Data mining is the process of extracting useful information from large sets of data. It involves using various techniques from statistics, machine learning, and database systems to identify patterns, relationships, and trends in the data. Data mining tools and technologies are widely used in various industries, including finance, healthcare, retail, and telecommunications. Data Mining is a process used by organizations to extract specific data from huge databases to solve business problems. It primarily turns raw data into useful information. “Mining” is the process of extraction of some valuable material from the earth e.g. coal mining, diamond mining, etc. In the context of computer science, “Data Mining” can be referred to as knowledge mining from data, knowledge extraction, data/pattern analysis, data archaeology, and data dredging. In the case of coal or diamond mining, the result of the extraction process is coal or diamond. But in the case of Data Mining, the result of the extraction process is not data!! Instead, data mining results are the patterns and knowledge that we gain at the end of the extraction process. In that sense, we can think of Data Mining as a step in the process of Knowledge Discovery or Knowledge Extraction. Nowadays, data mining is used in almost all places where a large amount of data is stored and processed. For example, banks typically use ‘data mining’ to find out their prospective

customers who could be interested in credit cards, personal loans, or insurance as well. Since banks have the transaction details and detailed profiles of their customers, they analyze all this data and try to find out patterns that help them predict that certain customers could be interested in personal loans, etc.

**Types of Data:** Data mining can be performed on the following types of data:

- Relational Database:** A relational database is a collection of multiple data sets formally organized by tables, records, and columns from which data can be accessed in various ways without having to recognize the database tables.
- Data warehouses:** A Data Warehouse is the technology that collects the data from various sources within the organization to provide meaningful business insights. The huge amount of data comes from multiple places such as Marketing and Finance. The extracted data is utilized for analytical purposes and helps in decision-making for a business organization.
- Data Repositories:** The Data Repository generally refers to a destination for data storage. However, many IT professionals utilize the term more clearly to refer to a specific kind of setup within an IT structure. For example, a group of databases, where an organization has kept various kinds of information.
- Object-Relational Database:** A combination of an object-oriented database model and relational database model is called an object-relational model. It supports Classes, Objects, Inheritance, etc.

**Transactional Database:** A transactional database refers to a database management system (DBMS) that has the potential to undo a database transaction if it is not performed appropriately.

**Quality of data:** Data quality is the measure of how well suited a data set is to serve its specific purpose. Measures of data quality are based on data quality characteristics such as accuracy, completeness, consistency, validity, uniqueness, and timeliness.

**Data Quality Dimensions**

- Accuracy:** The data should reflect actual, real-world scenarios; the measure of accuracy can be confirmed with a verifiable source.
- Completeness:** Completeness is a measure of the data's ability to effectively deliver all the required values that are available.
- Consistency:** Data consistency refers to the uniformity of data as it moves across networks and applications. The same data values stored in different locations should not conflict with one another.
- Validity:** Data should be collected according to defined business rules and parameters, and should conform to the right format and fall within the right range.
- Uniqueness:** Uniqueness ensures there are no duplications or overlapping of values across all data sets. Data cleansing and reduplication can help remedy a low uniqueness score.

**Timeliness:** Timely data is data that is available when it is required. Data may be updated in real time to ensure that it is readily available and accessible.

**Why is data quality important?**

1. Better business decisions
2. Improved business processes
3. Increased customer satisfaction

**Data preprocessing:** Data preprocessing is an important step in the data mining process. It refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data mining task. Some common steps in data preprocessing include:

- Data Cleaning:** This involves identifying and correcting errors or inconsistencies in the data, such as missing values, outliers, and duplicates. Various techniques can be used for data cleaning, such as imputation, removal, and transformation.
- Data Integration:** This involves combining data from multiple sources to create a unified dataset. Data integration can be challenging as it requires handling data with different formats, structures, and semantics. Techniques such as record linkage and data fusion can be used for data integration.
- Data Transformation:** This involves converting the data into a suitable format for analysis. Common techniques used in data transformation include normalization, standardization, and discretization. Normalization is used to scale the data to a common range, while standardization is used to transform the data to have zero mean and unit variance. Discretization is used to convert continuous data into discrete categories.

**Data Reduction:** This involves reducing the size of the dataset while preserving the important information. Data reduction can be achieved through techniques such as feature selection and feature extraction. Feature selection involves selecting a subset of relevant features from the dataset, while feature extraction involves transforming the data into a lower-dimensional space while preserving the important information.

**Data Discretization:** This involves dividing continuous data into discrete categories or intervals. Discretization is often used in data mining and machine learning algorithms that require categorical data. Discretization can be achieved through techniques such as equal width binning, equal frequency binning, and clustering.

**Data Normalization:** This involves scaling the data to a common range, such as between 0 and 1 or -1 and 1. Normalization is often used to handle data with different units and scales. Common normalization techniques include min-max normalization, z-score normalization, and decimal scaling.

**Data preprocessing** plays a crucial role in ensuring the quality of data and the accuracy of the analysis results. The specific steps involved in data preprocessing may vary depending on the nature of the data and the analysis goals. By performing these steps, the data mining process becomes more efficient and the results become more accurate.

**Similarity measures:** In a Data Mining sense, the similarity measure is a distance with dimensions describing object

features. That means if the distance among two data points is small then there is a high degree of similarity among the objects and vice versa. The similarity is subjective and depends heavily on the context and application. The similarity measures in data mining are generally expressed as a numerical value - it increases when the data samples are more alike. It is often expressed as a number between zero and one by conversion: zero means low similarity (the data objects are dissimilar). One means high similarity (the data objects are very similar). Here are some common similarity measures:

**Euclidean Distance:** This measure calculates the straight-line distance between two points in Euclidean space. It's commonly used in geometric contexts and is defined as the square root of the sum of the squares of the differences between corresponding coordinates.

**Hamming Distance:** This measure is used to compare strings of equal length and calculates the number of positions at which the corresponding symbols are different.

**Summary statistics:** Summary statistics summarize and provide information about the collected data. It characterizes the values in your data set. It can help understand the values better. It may include the total number of values, minimum value, and maximum value, along with the mean value and the standard deviation corresponding to a data collection.

- **Mean:** The average value of the data points. It's calculated by summing all values and dividing by the total number of observations.
- **Median:** The middle value in a sorted list of numbers. It represents the value below which 50% of the data fall.
- **Mode:** The most frequently occurring value in the dataset.
- **Variance:** A measure of the spread of data points around the mean. It's calculated by averaging the squared differences between each data point and the mean.
- **Standard Deviation:** A measure of the amount of variation or dispersion of a set of values. It's the square root of the variance.
- **Range:** The difference between the maximum and minimum values in the dataset.
- **Interquartile Range (IQR):** The range between the first quartile (25th percentile) and the third quartile (75th percentile). It indicates the spread of the middle 50% of the data.

**Percentiles:** Values that divide a dataset into 100 equal parts. Common percentiles include the quartiles (25th, 50th, and 75th percentiles) and the deciles (10th, 20th, ..., 90th percentiles).

**Summary statistics** provide valuable insights into the characteristics of a dataset, helping analysts and data scientists understand its properties and make informed decisions about further analysis or modeling.

**Applications Of Summary Statistics** The applications are far and wide and include an assortment of fields and professions – from academics, finance and investments, or even government organizations. Economic interests may lie in data pertaining to consumer spending, inflation, changes in the GDP, and more. Analysts involved in the Finance domain could be interested in companies and industries, market information with a focus on volumes and prices, consumer sentiment regarding a product or service, and many more variables.

**Data Distribution:** Distribution simply means collection or gathering of data, or scores, on variable. Generally, all these scores are arranged in specific order from smallest to largest. Data distributions are widely used in statistics. Suppose an engineer collects 500 data points on a shop floor. It does not give any value to the management unless they categorize or organize the data in a useful way. Data distribution methods organize the raw data into graphical methods (like histograms, box plots, run charts, etc.) and provide helpful information. Data distributions are a way of describing data sets by plotting individual data points on a graph. In data mining, understanding data distributions is crucial for various tasks such as anomaly detection, clustering, classification, and regression. Data distributions refer to the patterns or shapes that data points form when plotted on a graph. Different types of data distributions exist, each with its own characteristics. Some common data distributions include:

- **Continuous data** Continuous data is a type of information that can range from one extreme to another, usually measured on a scale such as temperature or weight.
- **Discrete data** Discrete data has a limited set of values and ranges, such as countable elements like the student population in a classroom or cars passing through an intersection.

**Discrete Distribution Types**

- **Binomial distribution**
- **Poisson distribution**
- **Hyper geometric distribution**

**Geometric distribution**

**Binomial Distribution:** The binomial distribution describes the probability of a certain number of successes (or failures) in a given number of trials or events. This type of distribution is used when there are only two possible outcomes for each trial, such as success or failure, heads or tails, yes or no etc.,

**Poisson Distribution:** The Poisson distribution describes the probability that an event will occur within a fixed time period when its rate is known but its exact timing cannot be predicted accurately enough to measure it directly. This type of distribution is useful for modelling random occurrences such as customer arrivals at stores, phone calls received by call centers etc.

**Hyper geometric Distribution:** The hyper geometric distribution is used to model scenarios where you draw from a finite population without replacement. Imagine you have an urn containing balls of different colors, and you're interested in understanding the likelihood of drawing a specific number of balls of a certain color in a fixed number of draws

**Geometric Distribution:**

The geometric distribution describes the probability of a success occurring on any given trial in a series of independent

trials when the probability of success for each trial is known. The geometric distribution models the number of trials needed to achieve the first success in a series of independent trials. Example: Flipping a Coin until Getting Heads.

Continuous Distribution Types • Normal Distribution • Lognormal distribution • Exponential distribution • Non-normal distributions

Normal Distribution: This distribution measures data points in a bell-shaped curve, with an equal number of data points to the left and right of the mean value. Normal Distributions can be used to predict future outcomes based on past trends.

Lognormal Distribution: Lognormal distributions measure data points in a curve shaped like a sigmoid function – a curved line beginning at zero and then increasing sharply to a peak and slowly decreasing.

Exponential Distribution: Exponential distributions measure data points with an exponential curve – a curve beginning at zero and gradually increasing in value.

Non-normal Distribution: Non-normal data distributions are often used when data does not fit into the normal data distribution categories, such as highly non-linear or data with outliers.

Basic data mining Tasks:

Basic data mining tasks encompass a range of techniques and methods used to extract useful insights and patterns from large datasets. The data mining tasks can be classified generally into two types based on what a specific task tries to achieve.

Those two categories are descriptive tasks and predictive tasks. The descriptive data mining tasks characterize the general properties of data whereas predictive data mining tasks perform inference on the available data set to predict how a new data set will behave.

Different Data Mining Tasks: There are a number of data mining tasks such as classification, prediction, time-series analysis, association, clustering, summarization etc. All these tasks are either predictive data mining tasks or descriptive data mining tasks. A data mining system can execute one or more of the above specified tasks as part of data mining.

Predictive data mining: This tasks come up with a model from the available data set that is helpful in predicting unknown or future values of another data set of interest. A medical practitioner trying to diagnose a disease based on the medical test results of a patient can be considered as a predictive data mining task.

Classification: Classification derives a model to determine the class of an object based on its attributes.

Classification can be used in direct marketing , that is to reduce marketing costs by targeting a set of customers who are likely to buy a new product. Using the available data, it is possible to know which customers purchased similar products and who did not purchase in the past.

Prediction: Prediction task predicts the possible values of missing or future data. Prediction involves developing a model based on the available data and this model is used in predicting future values of a new data set of interest.

For example, a model can predict the income of an employee based on education, experience and other demographic factors like place of stay, gender etc.

Time - Series Analysis: Time series is a sequence of events where the next event is determined by one or more of the preceding events. Time series reflects the process being measured and there are certain components that affect the behavior of a process.

Time series analysis includes methods to analyze timeseries data in order to extract useful patterns, trends, rules and statistics. Stock market prediction is an important application of time- series analysis.

Descriptive data mining tasks : It usually finds data describing patterns and comes up with new, significant information from the available data set. A retailer trying to identify products that are purchased together can be considered as a descriptive data mining task.

Association: Association discovers the association or connection among a set of items. Association identifies the relationships between objects. Association analysis is used for commodity management, advertising, catalog design, direct marketing etc.

A retailer can identify the products that normally customers purchase together or even find the customers who respond to the promotion of same kind of products.

Clustering: Clustering is used to identify data objects that are similar to one another. The similarity can be decided based on a number of factors like purchase behavior, responsiveness to certain actions, geographical locations and so on.

For example, an insurance company can cluster its customers based on age, residence, income etc.

Summarization: Summarization is the generalization of data. A set of relevant data is summarized which result in a smaller set that gives aggregated information of the data.

For example, the shopping done by a customer can be summarized into total products, total spending, offers used, etc.

Summary: Different data mining tasks are the core of data mining process. Different prediction and classification data mining tasks actually extract the required information from the available data sets.

These tasks provide a foundation for more advanced data mining and machine learning applications and are essential for extracting valuable knowledge from large and complex datasets.

Data mining v/s knowledge discovery in databases: KDD (Knowledge Discovery in Databases) is a field of computer science, which includes the tools and theories to help humans in extracting useful and previously unknown information (i.e., knowledge) from large collections of digitized data.

KDD consists of several steps, and Data Mining is one of them. Data Mining is the application of a specific algorithm to extract patterns from data. Nonetheless, KDD and Data Mining are used interchangeably. What is KDD? KDD

is a computer science field specializing in extracting previously unknown and interesting information from raw data. KDD is the whole process of trying to make sense of data by developing appropriate methods or techniques. This process deals with low-level mapping data into other forms that are more compact, abstract, and useful. This is achieved by creating short reports, modeling the process of generating data, and developing predictive models that can predict future cases. For example, it is currently used for various applications such as social network analysis, fraud detection, science, investment, manufacturing, telecommunications, data cleaning, sports, information retrieval, and marketing. KDD Process Steps: The following steps make up the cyclical process of KDD: → Data cleaning → Data Integration → Data selection → Data transformation → Data mining → Pattern evaluation → Knowledge presentation Data Cleaning: Data cleaning is defined as removal of noisy and irrelevant data from collection. Cleaning in case of Missing values. Data Integration: Data integration is defined as heterogeneous data from multiple sources combined in a common source(Data Warehouse). Data integration using Data Migration tools, Data Synchronization tools and ETL(Extract-Load-Transformation) process. Data Selection: Data selection is defined as the process where data relevant to the analysis is decided and retrieved from the data collection. For this we can use Neural network, Decision Trees, Naive bayes, Clustering, and Regression methods. Data Transformation: Data Transformation is defined as the process of transforming data into appropriate form required by mining procedure. Data Transformation is a two step process: 1. Data Mapping: Assigning elements from source base to destination to capture transformations. 2. Code generation: Creation of the actual transformation program. Data Mining: Data mining is defined as techniques that are applied to extract patterns potentially useful. It transforms task relevant data into patterns, and decides purpose of model using classification or characterization. Pattern Evaluation: Pattern Evaluation is defined as identifying strictly increasing patterns representing knowledge based on given measures. It find interestingness score of each pattern, and uses summarization and Visualization to make data understandable by user. Knowledge Representation: This involves presenting the results in a way that is meaningful and can be used to make decisions. Applications of KDD: • Business and Marketing: User analysis, market prediction, Segmenting clients, and focused marketing are all examples of business and marketing databases. • Manufacturing: Predictive system analysis, process improvement, and quality control. • Finance: Fraud investigation, evaluation of credit risk, and stock market research in the finance sector can be analyzed using the KDD method. • Healthcare: Drug progress, patient monitoring, and disease diagnosis from a large set of patient data. • Scientific research: Identifying patterns in massive scientific databases, such as genetics, astronomy, and climate. Advantages of KDD 1. Improves decision-making: KDD provides valuable insights and knowledge that can help organizations make better decisions. 2. Increased efficiency: KDD automates repetitive and time-consuming tasks and makes the data ready for analysis, which saves time and money. 3. Better customer service: KDD helps organizations gain a better understanding of their customers' needs and preferences, which can help them provide better customer service. 4. Fraud detection: KDD can be used to detect fraudulent activities by identifying patterns and anomalies in the data that may indicate fraud. 5. Predictive modeling: KDD can be used to build predictive models that can forecast future trends and patterns. Disadvantages of KDD 1. Privacy concerns: KDD can raise privacy concerns as it involves collecting and analyzing large amounts of data, which can include sensitive information about individuals. 2. Complexity: KDD can be a complex process that requires specialized skills and knowledge to implement and interpret the results. 3. Unintended consequences: KDD can lead to unintended consequences, such as bias or discrimination, if the data or models are not properly understood or used. 4. Data Quality: KDD process heavily depends on the quality of data, if data is not accurate or consistent, the results can be misleading 5. High cost: KDD can be an expensive process, requiring significant investments in hardware, software, and personnel. 6. Over fitting: KDD process can lead to overfitting, which is a common problem in machine learning where a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new unseen data. What is Data Mining? Data mining, also known as Knowledge Discovery in Databases, refers to the nontrivial extraction of implicit, previously unknown, and potentially useful information from data stored in databases. Data Mining is only a step within the overall KDD process. There are two major Data Mining goals defined by the application's goal: verification of discovery. Verification verifies the user's hypothesis about data, while discovery automatically finds interesting patterns. There are four major data mining tasks: clustering, classification, regression, and association (summarization). Clustering is identifying similar groups from unstructured data. Classification is learning rules that can be applied to new data. Regression is finding functions with minimal error to model data. Association looks for relationships between variables. Then, the specific data mining algorithm needs to be selected. Different algorithms like linear regression, logistic

regression, decision trees, and Naive Bayes can be selected depending on the goal. Then patterns of interest in one or more symbolic forms are searched. Finally, models are evaluated either using predictive accuracy or understandability.

Why do we need Data Mining? The volume of information is increasing every day that we can handle from business transactions, scientific data, sensor data, Pictures, videos, etc. So, we need a system that will be capable of extracting the essence of information available and that can automatically generate reports, views, or summaries of data for better decision-making.

Applications of Data Mining Some of the crucial applications of data mining are as follows:

- Risk evaluation: Data mining can be used to evaluate the probability of an occurrence, such as the defects in a product.
- Recommendation: Data mining can make product suggestions to customers based on their history or browsing patterns.
- Fraud detection: Data mining can spot patterns of shady behaviour, such as fraudulently using a credit card or inflating a policy.
- Medical diagnosis: Data mining can be utilised to diagnose medical disorders by seeing trends in medical records.

Data Mining vs KDD

Key Features

Data Mining KDD

Basic Definition

Data mining is the process of identifying patterns and extracting details about big data sets using intelligent methods. The KDD method is a complex and iterative approach to knowledge extraction from big data.

Goal To extract patterns from datasets. To discover knowledge from datasets. Scope In the KDD method, the fourth phase is called "data mining." KDD is a broad method that includes data mining as one of its steps.

Used Techniques

- Classification
- Clustering
- Decision Trees
- Dimensionality Reduction
- Neural Networks
- Regression
- Data cleaning
- Data Integration
- Data selection
- Data transformation
- Data mining
- Pattern evaluation

Knowledge Presentation Example

Clustering groups of data elements based on how similar they are. Data analysis to find patterns and links.

Issues in Data mining: Data mining is not an easy task, as the algorithms used can get very complex and data is not always available at one place. It needs to be integrated from various heterogeneous data sources. These factors also create some issues. Here in this tutorial, we will discuss the major issues regarding –

- Mining Methodology and User Interaction
- Performance Issues
- Diverse Data Types Issues

The following diagram describes the major issues:

Mining Methodology and User Interaction Issues

It refers to the following kinds of issues –

- Mining different kinds of knowledge in databases – Different users may be interested in different kinds of knowledge. Therefore it is necessary for data mining to cover a broad range of knowledge discovery task.
- Interactive mining of knowledge at multiple levels of abstraction – The data mining process needs to be interactive because it allows users to focus the search for patterns, providing and refining data mining requests based on the returned results.
- Incorporation of background knowledge – To guide discovery process and to express the discovered patterns, the background knowledge can be used. Background knowledge may be used to express the discovered patterns not only in concise terms but at multiple levels of abstraction.

- Data mining query languages and ad hoc data mining – Data Mining Query language that allows the user to describe ad hoc mining tasks, should be integrated with a data warehouse query language and optimized for efficient and flexible data mining.
- Presentation and visualization of data mining results – Once the patterns are discovered it needs to be expressed in high level languages, and visual representations. These representations should be easily understandable.
- Handling noisy or incomplete data – The data cleaning methods are required to handle the noise and incomplete objects while mining the data regularities. If the data cleaning methods are not there then the accuracy of the discovered patterns will be poor.
- Pattern evaluation – The patterns discovered should be interesting because either they represent common knowledge or lack novelty.

Performance Issues: There can be performance-related issues such as follows –

- Efficiency and scalability of data mining algorithms – In order to effectively extract the information from huge amount of data in databases, data mining algorithm must be efficient and scalable.
- Parallel, distributed, and incremental mining algorithms – The factors such as huge size of databases, wide distribution of data, and complexity of data mining methods motivate the development of parallel and distributed data mining algorithms. These algorithms divide the data into partitions which is further processed in a parallel fashion. Then the results from the partitions is merged. The incremental algorithms, update databases without mining the data again from scratch.
- Diverse Data Types Issues
- Handling of relational and complex types of data – The database may contain complex data objects, multimedia data objects, spatial data, temporal data etc. It is not possible for one system to mine all these kind of data.
- Mining information from heterogeneous databases and global information systems – The data is available at different data sources on LAN or WAN. These data source may be structured, semi structured or unstructured. Therefore mining the knowledge from them adds challenges to data mining.

Introduction to Fuzzy sets and fuzzy logic

Fuzzy Logic: The term fuzzy refers to things that are not clear or are vague. In the real world many times we encounter a situation when we can't determine whether the state

is true or false, their fuzzy logic provides very valuable flexibility for reasoning. Fuzzy Logic is a form of many-valued logic in which the truth values of variables may be any real number between 0 and 1, instead of just the traditional values of true or false. Fuzzy Logic is based on the idea that in many cases, the concept of true or false is too restrictive, and that there are many shades of gray in between. It allows for partial truths, where a statement can be partially true or false, rather than fully true or false. Fuzzy Logic is used in a wide range of applications, such as control systems, image processing, natural language processing, medical diagnosis, and artificial intelligence. Fuzzy Logic is implemented using Fuzzy Rules, which are if-then statements that express the relationship between input variables and output variables in a fuzzy way. Fuzzy Logic is implemented using Fuzzy Rules, which are if-then statements that express the relationship between input variables and output variables in a fuzzy way. The output of a Fuzzy Logic system is a fuzzy set, which is a set of membership degrees for each possible output value. Example of Fuzzy Logic as comparing to Boolean Logic Application — It has been used in the automotive system for speed control, traffic control. — It is used for decision-making support systems and personal evaluation in the large company business. — Fuzzy logic is used in Natural language processing and various intensive applications in Artificial Intelligence. — Fuzzy logic is extensively used in modern control systems such as expert systems. — Fuzzy Logic is used with Neural Networks as it mimics how a person would make decisions, only much faster. Fuzzy sets: Fuzzy set is a set having degrees of membership between 1 and 0. Fuzzy sets are represented with tilde character( $\sim$ ). For example, Number of cars following traffic signals at a particular time out of all cars present will have membership value between [0,1]. A set is a term, which is a collection of unordered or ordered elements. Following are the various examples of a set: 1. A set of all-natural numbers 2. A set of students in a class. 3. A set of all cities in a state. 4. A set of upper-case letters of the alphabet. In the fuzzy set, the partial membership also exists. This theory released as an extension of classical set theory. This theory is denoted mathematically as A fuzzy set ( $\tilde{A}$ ) is a pair of U and M, where U is the Universe of discourse and M is the membership function which takes on values in the interval [ 0, 1 ]. The universe of discourse (U) is also denoted by  $\Omega$  or X. Example: In a fuzzy set representing the height of people in a "tall" category, a person who is 6 feet tall might have a membership value of 0.8, indicating a high degree of belongingness to the set "tall." 1. Union Operation: The union operation of a fuzzy set is defined by:  $\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$  Example: Let's suppose A is a set which contains following elements:  $A = \{(X_1, 0.6), (X_2, 0.2), (X_3, 1), (X_4, 0.4)\}$  And, B is a set which contains following elements:  $B = \{(X_1, 0.1), (X_2, 0.8), (X_3, 0), (X_4, 0.9)\}$  then,  $A \cup B = \{(X_1, 0.6), (X_2, 0.8), (X_3, 1), (X_4, 0.9)\}$  2. Intersection Operation: The intersection operation of fuzzy set is defined by:  $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$  Example: Let's suppose A is a set which contains following elements:  $A = \{(X_1, 0.3), (X_2, 0.7), (X_3, 0.5), (X_4, 0.1)\}$  And, B is a set which contains following elements:  $B = \{(X_1, 0.8), (X_2, 0.2), (X_3, 0.4), (X_4, 0.9)\}$  then,  $A \cap B = \{(X_1, 0.3), (X_2, 0.2), (X_3, 0.4), (X_4, 0.1)\}$  3. Complement Operation: The complement operation of fuzzy set is defined by:  $\mu_{\tilde{A}}(x) = 1 - \mu_A(x)$ , Example: Let's suppose A is a set which contains following elements:  $A = \{(X_1, 0.3), (X_2, 0.8), (X_3, 0.5), (X_4, 0.1)\}$  Then,  $\tilde{A} = \{(X_1, 0.7), (X_2, 0.2), (X_3, 0.5), (X_4, 0.9)\}$

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Unit – 4 Syllabus: Supervised Learning: Classification: Statistical-based algorithms, Distance-based algorithms, Decision tree-based algorithms, Neural network-based algorithms, Rule-based algorithms, Probabilistic Classifiers. Supervised Learning: Supervised learning, also known as supervised machine learning, is a subcategory of machine learning and artificial intelligence. It is defined by its use of labeled datasets to train algorithms that to classify data or predict outcomes accurately. Supervised learning is utilized in voice recognition to help virtual assistants and other applications recognize and understand spoken commands. A labeled dataset of spoken words and phrases with corresponding text transcripts is used to train a machine learning algorithm in such scenarios. With the rise of big data, supervised learning has become critical for industries such as finance, healthcare, and e-commerce. To appreciate exactly why it has gained such importance, let's first understand what supervised learning is. In simple terms, supervised learning is a standard machine learning technique that involves training a model with labeled data. This blog will explain the fundamentals of supervised learning, its types, algorithms, and applications. We will also go over the steps involved in implementing supervised learning and some of the challenges that come with it. What is Supervised Learning? Supervised learning is a type of machine learning in which a computer

algorithm learns to make predictions or decisions based on labeled data. Labeled data is made up of previously known input variables (also known as features) and output variables (also known as labels). By analyzing patterns and relationships between input and output variables in labeled data, the algorithm learns to make predictions. Image and speech recognition, recommendation systems, and fraud detection are all examples of how supervised learning is used. The examples below will help explain what supervised learning is. Examples of Supervised Learning Email Filtering Supervised learning is commonly used in email filtering to classify incoming emails as spam or legitimate. A machine learning algorithm is trained using a labeled dataset containing examples of both spam and legitimate emails. The algorithm then extracts relevant information from each email, such as the sender's information, the subject, the message body, and so on. It learns from the labeled dataset to identify patterns and relationships between these features and their corresponding labels (spam or legitimate). Once trained, the algorithm can use the extracted features to predict the label of new, unseen emails. If an email is predicted to be spam, it can be automatically filtered into a spam folder, saving the user's inbox space.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Credit Scoring In credit scoring, supervised learning is used to predict the creditworthiness of loan applicants. A labeled dataset containing examples of past loan applicants and their credit history, income, employment status, and other relevant factors is used to train a machine learning algorithm. The algorithm learns to recognize patterns and relationships between these features and their corresponding labels, such as whether or not the loan was repaid. Once trained, the algorithm can predict loan repayment likelihood for new loan applicants based on their input features. Voice Recognition Supervised learning is utilized in voice recognition to help virtual assistants and other applications recognize and understand spoken commands. A labeled dataset of spoken words and phrases with corresponding text transcripts is used to train a machine learning algorithm in such scenarios. The algorithm learns to recognize relationships between spoken word audio features such as pitch, amplitude, and frequency and their textual representations from the labeled dataset. Following the training phase, the algorithm can begin analyzing new audio inputs and attempting to transcribe them into text form. This allows virtual assistants to understand and respond to spoken commands like managing reminders, playing music, or controlling smart home devices.

Different Types of Supervised Learning Regression Regression is a supervised learning method for determining the relationship between dependent and independent variables. In addition, it employs labeled datasets in an algorithm to forecast continuous output for various data. Here, it is widely used in situations where the output must be a single value, such as weight or height. There are two types of regression:

1. Linear regression: This is used to detect the relationship between two variables and to make future predictions. It is further subdivided according to the number of independent and dependent variables. Simple linear regression, for example, is used when there is only one independent and one dependent variable. Multiple linear regression is used when there are two or more independent and dependent variables.
2. Logistic regression: Logistic regression is used when the dependent variable is categorical or has binary outputs such as 'yes' or 'no'. Since logistic regression is used to solve binary classification problems, it predicts discrete values for variables.

Naive Bayes The Naive Bayes algorithm is well-suited for large datasets because each program in the algorithm operates independently, and the presence of one feature has no effect on the other. Its applications include text classification, and recommendation systems, among others. There are various Naive Bayes models of which the decision tree is commonly used in business. A decision tree, unlike a flowchart, is a supervised learning algorithm composed of control statements containing decisions and their consequences. Iterative Dichotomiser 3 (ID3) and Classification algorithm and Regression Trees (CART) are two popular decision tree algorithms used in a variety of industries.

Classification Classification is a type of supervised learning algorithm which involves the process of accurately assigning data to different categories or classes. In essence, it entails identifying and analyzing specific entities in order to determine the appropriate category or class. K-nearest neighbor, Random forest, Support vector machines, Decision trees, and Linear classifiers are some popular classification algorithms. Neural Networks Neural Networks perform the process of grouping or categorizing raw data. Additionally, this algorithm is also employed in the interpretation of sensory data and the identification of patterns. The algorithm's use, however, is limited due to the need for high computational resources.

Random Forest The random forest algorithm is known as an ensemble method as it combines multiple supervised learning techniques to make a conclusion. Moreover, it uses several decision trees to classify each tree, making it a popular choice in a variety of industries.

Steps Involved in Supervised Learning

The following are some of the common steps involved in supervised learning:

- Gather labeled data
- Divide the data into two sets: Training and Testing
- Select an appropriate

algorithm On the training set, train the algorithm Analyze the algorithm's performance on the testing set If necessary, fine-tune the model to improve performance Make predictions on new, unlabeled data using the trained model Advantages and Disadvantages- When implemented in a professional context, supervised learning can foster a healthy workplace environment that prioritizes ongoing education and supports a culture of continuous growth. Some of its chief advantages include: It gathers previous data, which aids in learning from past mistakes It is a powerful Artificial Intelligence (AI) tool that can handle a wide range of business functions on its own Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science It is a reliable algorithm Some of the drawbacks of supervised learning are: Large data sets tend to be difficult to categorize To operate, a certain level of expertise is required It takes a long time to process To conclude, supervised learning is a well-known machine learning technique used for training models to predict outputs based on input data. With proper model selection and training, supervised learning can be a powerful tool for solving a wide variety of real-world problems. To learn more about these subjects, explore these machine learning and artificial intelligence courses offered by Emeritus in association with the best universities around the world. Supervised and Unsupervised learning Supervised learning: Supervised learning, as the name indicates, has the presence of a supervisor as a teacher. Basically supervised learning is when we teach or train the machine using data that is well-labelled. Which means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples (data) so that the supervised learning algorithm analyses the training data(set of training examples) and produces a correct outcome from labeled data. For instance, suppose you are given a basket filled with different kinds of fruits. Now the first step is to train the machine with all the different fruits one by one like this: If the shape of the object is rounded and has a depression at the top, is red in color, then it will be labeled as –Apple. If the shape of the object is a long curving cylinder having Green-Yellow color, then it will be labeled as –Banana. Now suppose after training the data, you have given a new separate fruit, say Banana from the basket, and asked to identify it. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Since the machine has already learned the things from previous data and this time has to use it wisely. It will first classify the fruit with its shape and color and would confirm the fruit name as BANANA and put it in the Banana category. Thus the machine learns the things from training data(basket containing fruits) and then applies the knowledge to test data(new fruit). Supervised learning is classified into two categories of algorithms: Classification: A classification problem is when the output variable is a category, such as “Red” or “blue”, “disease” or “no disease”. Regression: A regression problem is when the output variable is a real value, such as “dollars” or “weight”. Supervised learning deals with or learns with “labeled” data. This implies that some data is already tagged with the correct answer. Types:- Regression Logistic Regression Classification Naive Bayes Classifiers K-NN (k nearest neighbors) Decision Trees Support Vector Machine Advantages:- Supervised learning allows collecting data and produces data output from previous experiences. Helps to optimize performance criteria with the help of experience. Supervised machine learning helps to solve various types of real-world computation problems. It performs classification and regression tasks. It allows estimating or mapping the result to a new sample. We have complete control over choosing the number of classes we want in the training data. Disadvantages:- Classifying big data can be challenging. Training for supervised learning needs a lot of computation time. So, it requires a lot of time. Supervised learning cannot handle all complex tasks in Machine Learning. Computation time is vast for supervised learning. It requires a labelled data set. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science It requires a training process. Unsupervised learning Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of the machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data. Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore the machine is restricted to find the hidden structure in unlabeled data by itself. For instance, suppose it is given an image having both dogs and cats which it has never seen. Thus the machine has no idea about the features of dogs and cats so we can't categorize it as ‘dogs and cats’. But it can categorize them according to their similarities, patterns, and differences, i.e., we can easily categorize the above picture into two parts. The first may contain all pics having dogs in them and the second part may contain all pics having cats in them. Here you didn't learn anything before, which means no training data or examples. It allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with unlabelled data. Unsupervised learning is classified into two categories of algorithms: Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by

purchasing behavior. Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y. Types of Unsupervised Learning:- Clustering 1. Exclusive (partitioning) 2. Agglomerative 3. Overlapping 4. Probabilistic Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Clustering Types:- 1. Hierarchical clustering 2. K-means clustering 3. Principal Component Analysis 4. Singular Value Decomposition 5. Independent Component Analysis Supervised vs. Unsupervised Machine Learning: Parameters Supervised machine learning Unsupervised machine learning Input Data Algorithms are trained using labeled data. Algorithms are used against data that is not labeled Computational Simpler method Computationally complex Complexity Accuracy Highly accurate Less accurate No. of classes No. of classes is known No. of classes is not known Data Analysis Uses offline analysis Uses real-time analysis of data Algorithms Linear and Logistics regression, Random K-Means clustering, Hierarchical forest, Support Vector Machine, Neural used clustering, Apriori algorithm, etc. Network, etc. Output Desired output is given. Desired output is not given. Training data Use training data to infer model. No training data is used. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science It does not require training data to be labeled. Dimensionality reduction can be easily accomplished using unsupervised learning. Capable of finding previously unknown patterns in data. Flexibility: Unsupervised learning is flexible in that it can be applied to a wide variety of problems, including clustering, anomaly detection, and association rule mining. Exploration: Unsupervised learning allows for the exploration of data and the discovery of novel and potentially useful patterns that may not be apparent from the outset. Low cost: Unsupervised learning is often less expensive than supervised learning because it doesn't require labeled data, which can be time-consuming and costly to obtain. Disadvantages of unsupervised learning : Difficult to measure accuracy or effectiveness due to lack of predefined answers during training. The results often have lesser accuracy. The user needs to spend time interpreting and label the classes which follow that classification. Lack of guidance: Unsupervised learning lacks the guidance and feedback provided by labeled data, which can make it difficult to know whether the discovered patterns are relevant or useful. Sensitivity to data quality: Unsupervised learning can be sensitive to data quality, including missing values, outliers, and noisy data. Scalability: Unsupervised learning can be computationally expensive, particularly for large datasets or complex algorithms, which can limit its scalability. It is not possible to learn larger and more complex models than with supervised learning. It is possible to learn complex models with unsupervised learning. Complex model Model We can test our model. We cannot test our model. Called as Supervised learning is also called Unsupervised learning is also called classification. clustering. Example Example: Optical character recognition. Example: Find a face in an image. Advantages of unsupervised learning: Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Classification: Statistical-based algorithms- A statistical or data mining algorithm is a mathematical expression of certain aspects of the patterns they find in data. Different algorithms provide different perspectives on the complete nature of the pattern. There are a variety of different algorithms that can be used for statistical classification, but some of the most popular include support vector machines, logistic regression, and decision trees. Each algorithm has its own strengths and weaknesses, so it is important to choose the right one for your specific problem. Statistical classification deals with rules of case assignment to categories or classes. The classification, or decision rule, is expressed in terms of a set of random variables — the case features. There are two types of statistical-based algorithms which are as follows – Regression – Regression issues deal with the evaluation of an output value located on input values. When utilized for classification, the input values are values from the database and the output values define the classes. Regression can be used to clarify classification issues, but it is used for different applications including forecasting. The elementary form of regression is simple linear regression that includes only one predictor and a prediction. Regression can be used to implement classification using two various methods which are as follows – o Division – The data are divided into regions located on class. o Prediction – Formulas are created to predict the output class's value. Bayesian Classification – Statistical classifiers are used for the classification. Bayesian classification is based on the Bayes theorem. Bayesian classifiers view high efficiency and speed when used to high databases. Bayes Theorem – Let  $X$  be a data tuple. In the Bayesian method,  $X$  is treated as "evidence." Let  $H$  be some hypothesis, including that the data tuple  $X$  belongs to a particularized class  $C$ . The probability  $P(H|X)$  is decided to define the data. This probability  $P(H|X)$  is the probability that hypothesis  $H$ 's influence has given the "evidence" or noticed data tuple  $X$ .  $P(H|X)$  is the posterior probability of  $H$  conditioned on  $X$ . For instance, consider the nature of data tuples is limited to users defined by the attribute age and income, commonly, and that  $X$  is 30 years old users with Rs. 20,000 income. Assume that  $H$  is the hypothesis that the user will purchase a computer. Thus  $P(H|X)$  reverses the probability

that user X will purchase a computer given that the user's age and income are acknowledged.  $P(H)$  is the prior probability of H. For instance, this is the probability that any given user will purchase a computer, regardless of age, income, or some other data. The posterior probability  $P(H|X)$  is located on more data than the prior probability  $P(H)$ , which is free of X. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Likewise,  $P(X|H)$  is the posterior probability of X conditioned on H. It is the probability that a user X is 30 years old and gains Rs. 20,000.  $P(H)$ ,  $P(X|H)$ , and  $P(X)$  can be measured from the given information. Bayes theorem supports a method of computing the posterior probability  $P(H|X)$ , from  $P(H)$ ,  $P(X|H)$ , and  $P(X)$ . It is given by  $P(H|X)=P(X|H).P(H) / P(X)$  Statistical Methods in Data Mining Data mining refers to extracting or mining knowledge from large amounts of data. In other words, data mining is the science, art, and technology of discovering large and complex bodies of data in order to discover useful patterns. Theoreticians and practitioners are continually seeking improved techniques to make the process more efficient, cost-effective, and accurate. Any situation can be analyzed in two ways in data mining: Statistical Analysis: In statistics, data is collected, analyzed, explored, and presented to identify patterns and trends. Alternatively, it is referred to as quantitative analysis. Non-statistical Analysis: This analysis provides generalized information and includes sound, still images, and moving images. Descriptive Statistics: The purpose of descriptive statistics is to organize data and identify the main characteristics of that data. Graphs or numbers summarize the data. Average, Mode, SD(Standard Deviation), and Correlation are some of the commonly used descriptive statistical methods. Inferential Statistics: The process of drawing conclusions based on probability theory and generalizing the data. By analyzing sample statistics, you can infer parameters about populations and make models of relationships within data. There are various statistical terms that one should be aware of while dealing with statistics. Some of these are: Population Sample Variable Quantitative Variable Qualitative Variable Discrete Variable Continuous Variable Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Now, let's start discussing statistical methods. This is the analysis of raw data using mathematical formulas, models, and techniques. Through the use of statistical methods, information is extracted from research data, and different ways are available to judge the robustness of research outputs. As a matter of fact, today's statistical methods used in the data mining field typically are derived from the vast statistical toolkit developed to answer problems arising in other fields. These techniques are taught in science curriculums. It is necessary to check and test several hypotheses. The hypotheses described above help us assess the validity of our data mining endeavor when attempting to infer any inferences from the data under study. When using more complex and sophisticated statistical estimators and tests, these issues become more pronounced. For extracting knowledge from databases containing different types of observations, a variety of statistical methods are available in Data Mining and some of these are: Logistic regression analysis Correlation analysis Regression analysis Discriminate analysis Linear discriminant analysis (LDA) Classification Clustering Outlier detection Classification and regression trees, Correspondence analysis Nonparametric regression, Statistical pattern recognition, Categorical data analysis, Time-series methods for trends and periodicity Artificial neural networks Now, let's try to understand some of the important statistical methods which are used in data mining: Linear Regression: The linear regression method uses the best linear relationship between the independent and dependent variables to predict the target variable. In order to achieve the best fit, make sure that all the distances between the shape and the actual observations at each point are as small as possible. A good fit can be determined by determining that no other position would produce fewer errors given the shape chosen. Simple linear regression and multiple linear regression are the two major types of linear regression. By fitting a linear relationship to the independent variable, the simple linear regression predicts the dependent variable. Using multiple independent variables, multiple linear Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science regression fits the best linear relationship with the dependent variable. For more details, you can refer linear regression. Classification: This is a method of data mining in which a collection of data is categorized so that a greater degree of accuracy can be predicted and analyzed. An effective way to analyze very large datasets is to classify them. Classification is one of several methods aimed at improving the efficiency of the analysis process. A Logistic Regression and a Discriminant Analysis stand out as two major classification techniques. Logistic Regression: It can also be applied to machine learning applications and predictive analytics. In this approach, the dependent variable is either binary (binary regression) or multinomial (multinomial regression): either one of the two or a set of one, two, three, or four options. With a logistic regression equation, one can estimate probabilities regarding the relationship between the independent variable and the dependent variable. For understanding logistic regression analysis in detail, you can refer to logistic regression. Discriminant Analysis: A

Discriminant Analysis is a statistical method of analyzing data based on the measurements of categories or clusters and categorizing new observations into one or more populations that were identified a priori. The discriminant analysis models each response class independently then uses Bayes's theorem to flip these projections around to estimate the likelihood of each response category given the value of X. These models can be either linear or quadratic. Linear Discriminant Analysis: According to Linear Discriminant Analysis, each observation is assigned a discriminant score to classify it into a response variable class. By combining the independent variables in a linear fashion, these scores can be obtained. Based on this model, observations are drawn from a Gaussian distribution, and the predictor variables are correlated across all k levels of the response variable, Y. and for further details linear discriminant analysis Quadratic Discriminant Analysis: An alternative approach is provided by Quadratic Discriminant Analysis. LDA and QDA both assume Gaussian distributions for the observations of the Y classes. Unlike LDA, QDA considers each class to have its own covariance matrix. As a result, the predictor variables have different variances across the k levels in Y. Correlation Analysis: In statistical terms, correlation analysis captures the relationship between variables in a pair. The value of such variables is usually stored in a column or rows of a database table and represents a property of an object. Regression Analysis: Based on a set of numeric data, regression is a data mining method that predicts a range of numerical values (also known as continuous values). You could, for instance, use regression to predict the cost of goods and services based on other variables. A regression model is used across Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science numerous industries for forecasting financial data, modeling environmental conditions, and analyzing trends. The first step in creating good statistics is having good data that was derived with an aim in mind. There are two main types of data: an input (independent or predictor) variable, which we control or are able to measure, and an output (dependent or response) variable which is observed. A few will be quantitative measurements, but others may be qualitative or categorical variables (called factors). Distance-based algorithmsDistance based algorithms are methods of classification that each item that is mapped to the same class may be thought of as more similar to other items in the class than it is to the items found in the other classes. An object o in a data set S is a distance-based (DB) outlier with parameters p and d, i.e., DB (p, d), if minimum a fraction p of the objects in S lie at a distance higher than d from o. In other words, instead of depending on statistical tests, it can think of distance-based outliers as those objects who do not have enough neighbors. The neighbors are represented based on distance from the given object. In comparison with statistical-based methods, distance-based outlier detection generalizes or merges the ideas behind discordancy testing for standard distributions. Hence, a distance-based outlier is also known as a unified outlier or UO-outlier. Distance-based outlier detection prevents the excessive calculation that can be related to fitting the observed distribution into some standard distribution and in choosing discordancy tests. For some Discordancy tests, it can be displayed that if an object o is an outlier as per the given test, then o is also a DB (p, d) outlier for some properly represented p and d. For instance, if objects that lie 3 or more standard deviations from the mean are treated to be outliers, considering a normal distribution, then this representation can be "unified" by a DB(0.9988, 0.13s)-an outlier. There are several efficient algorithms for mining distance-based outliers that have been created which are as follows – Index-based algorithm – Given a data set, the index-based algorithm facilitates multidimensional indexing structures, including R-trees or k-d trees, to search for neighbors of each object o inside radius d around that object. Let M be the maximum number of objects within the d-neighborhood of an outlier. Hence, once M + 1 neighbors of object o are discovered, it is accessible that o is not an outlier. This algorithm has the lowest case complexity of O (k \* n<sup>2</sup> ), where k is the dimensionality, and n is the number of objects in the data set. Nested-loop algorithm – The nested-loop algorithm has the same evaluation complexity as the index-based algorithm but avoids index structure construction and tries to minimize the number of I/O's. It divides the memory buffer areas into two halves, and the data is set into several logical blocks. Cell-based algorithm – It can avoid O(n<sup>2</sup> ) computational complexity, a cell-based algorithm was developed for memory-resident data sets. Its complexity is O (ek + n), where c is a constant based on the number of cells, and k is the dimensionality. In this method, the data space is partitioned into cells with a side length similar to d/vk. Each cell has two layers surrounding it. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science The first layer is one cell thick, while the second is vk cells thick, rounded up to the closest integer. The algorithm counts outliers on a cell-by-cell instead of an object-by-object basis. For a given cell, it accumulates three counts including the number of objects in the cell, in the cell and the first layer together, and in the cell and both layers together. Decision tree-based algorithmsDecision trees use multiple algorithms to decide to split a node into two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-

nodes. In other words, we can say that the purity of the node increases with respect to the target variable. A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node. The following decision tree is for the concept buy\_computer that indicates whether a customer at a company is likely to buy a computer or not. Each internal node represents a test on an attribute. Each leaf node represents a class. The benefits of having a decision tree are as follows – It does not require any domain knowledge. It is easy to comprehend. The learning and classification steps of a decision tree are simple and fast. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Decision Tree Induction Algorithm A machine researcher named J. Ross Quinlan in 1980 developed a decision tree algorithm known as ID3 (Iterative Dichotomiser). Later, he presented C4.5, which was the successor of ID3. ID3 and C4.5 adopt a greedy approach. In this algorithm, there is no backtracking; the trees are constructed in a top-down recursive divide-and-conquer manner. Generating a decision tree form training tuples of data partition D Algorithm : Generate\_decision\_tree Input: Data partition, D, which is a set of training tuples and their associated class labels. attribute\_list, the set of candidate attributes. Attribute selection method, a procedure to determine the splitting criterion that best partitions the data tuples into individual classes. This criterion includes a splitting\_attribute and either a splitting point or splitting subset. Output: A Decision Tree Method create a node N; if tuples in D are all of the same class, C then return N as leaf node labeled with class C; if attribute\_list is empty then return N as leaf node with labeled with majority class in D; || majority voting apply attribute\_selection\_method(D, attribute\_list) to find the best splitting\_criterion; label node N with splitting\_criterion; if splitting\_attribute is discrete-valued and multiway splits allowed then // no restricted to binary trees attribute\_list = splitting\_attribute; // remove splitting attribute for each outcome j of splitting criterion // partition the tuples and grow subtrees for each partition Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science let  $D_j$  be the set of data tuples in D satisfying outcome  $j$ ; // a partition if  $D_j$  is empty then attach a leaf labeled with the majority class in D to node N; else attach the node returned by Generate decision tree( $D_j$ , attribute list) to node N; end for return N; Tree Pruning Tree pruning is performed in order to remove anomalies in the training data due to noise or outliers. The pruned trees are smaller and less complex. Tree Pruning Approaches There are two approaches to prune a tree – Pre-pruning – The tree is pruned by halting its construction early. Post-pruning - This approach removes a sub-tree from a fully grown tree. Cost Complexity The cost complexity is measured by the following two parameters – Number of leaves in the tree, and Error rate of the tree. Neural network-based algorithms Neural Network is an information processing paradigm that is inspired by the human nervous system. As in the Human Nervous system, we have Biological neurons in the same way in Neural networks we have Artificial Neurons which is a Mathematical Function that originates from biological neurons. The human brain is estimated to have around 10 billion neurons each connected on average to 10,000 other neurons. Each neuron receives signals through synapses that control the effects of the signal on the neuron. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science How Artificial Neural Network Work? Let us Suppose that there are n input like  $X_1, X_2, \dots, X_n$  to a neuron.  $\Rightarrow$  The weight connecting n number of inputs to a neuron are represented by  $[W] = [W_1, W_2, \dots, W_n]$ .  $\Rightarrow$  The Function of summing junction of an artificial neuron is to collect the weighted inputs and sum them up.  $Y_{in} = [X_1 * W_1 + X_2 * W_2 + \dots + X_n * W_n]$   $\Rightarrow$  The output of summing junction may sometimes become equal to zero and to prevent such a situation, a bias of fixed value  $B_0$  is added to it.  $Y_{out} = [X_1 * W_1 + X_2 * W_2 + \dots + X_n * W_n] + B_0$  //  $Y_{in}$  then move toward the Activation Function.  $\Rightarrow$  The output Y of a neuron largely depends on its Activation Function (also known as transfer function).  $\Rightarrow$  There are different types of Activation Function are in use, Such as 1. Identity Function 2. Binary Step Function With Threshold 3. Bipolar Step Function With Threshold 4. Binary Sigmoid Function 5. Bipolar Sigmoid Function Rule-based algorithms Rule-based classifier makes use of a set of IF-THEN rules for classification. We can express a rule in the following from – IF condition THEN conclusion Let us consider a rule R1, R1: IF age = youth AND student = yes THEN buy\_computer = yes Points to remember – The IF part of the rule is called rule antecedent or precondition. The THEN part of the rule is called rule consequent. The antecedent part the condition consist of one or more attribute tests and these tests are logically ANDed. The consequent part consists of class prediction. Note – We can also write rule R1 as follows – R1: (age = youth) ^ (student = yes)) / (buys computer = yes) Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science If the condition holds true for a given tuple, then the antecedent is satisfied. Rule Extraction Here we will learn how to build a rule-based classifier by extracting IF-THEN rules from a decision tree. Points to remember – To extract a rule from a decision tree – One rule is

created for each path from the root to the leaf node. To form a rule antecedent, each splitting criterion is logically ANDed. The leaf node holds the class prediction, forming the rule consequent. Rule Induction Using Sequential Covering Algorithm Sequential Covering Algorithm can be used to extract IF-THEN rules from the training data. We do not require to generate a decision tree first. In this algorithm, each rule for a given class covers many of the tuples of that class. Some of the sequential Covering Algorithms are AQ, CN2, and RIPPER. As per the general strategy the rules are learned one at a time. For each time rules are learned, a tuple covered by the rule is removed and the process continues for the rest of the tuples. This is because the path to each leaf in a decision tree corresponds to a rule. The Following is the sequential learning Algorithm where rules are learned for one class at a time. When learning a rule from a class  $C_i$ , we want the rule to cover all the tuples from class  $C$  only and no tuple from any other class. Algorithm: Sequential Covering Input:  $D$ , a data set class-labeled tuples,  $Att\_vals$ , the set of all attributes and their possible values. Output: A Set of IF-THEN rules.

Method: Rule\_set={ }; // initial set of rules learned is empty for each class  $c$  do repeat Rule = Learn\_One\_Rule( $D$ ,  $Att\_vals$ ,  $c$ ); remove tuples covered by Rule from  $D$ ; until termination condition; Rule\_set=Rule\_set+Rule; // add a new rule to rule-set end for return Rule\_Set; Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Rule Pruning The rule is pruned is due to the following reason – The Assessment of quality is made on the original set of training data. The rule may perform well on training data but less well on subsequent data. That's why the rule pruning is required. The rule is pruned by removing conjunct. The rule  $R$  is pruned, if pruned version of  $R$  has greater quality than what was assessed on an independent set of tuples. FOIL is one of the simple and effective method for rule pruning. For a given rule  $R$ ,  $FOIL\_Prune = pos - neg / pos + neg$  where  $pos$  and  $neg$  is the number of positive tuples covered by  $R$ , respectively. Note – This value will increase with the accuracy of  $R$  on the pruning set. Hence, if the  $FOIL\_Prune$  value is higher for the pruned version of  $R$ , then we prune  $R$ . Probabilistic ClassifiersProbabilistic models are an essential component of machine learning, which aims to learn patterns from data and make predictions on new, unseen data. They are statistical models that capture the inherent uncertainty in data and incorporate it into their predictions. Probabilistic models are used in various applications such as image and speech recognition, natural language processing, and recommendation systems. In recent years, significant progress has been made in developing probabilistic models that can handle large datasets efficiently. Categories Of Probabilistic Models These models can be classified into the following categories:

Generative models Discriminative models Graphical models Generative models: Generative models aim to model the joint distribution of the input and output variables. These models generate new data based on the probability distribution of the original dataset. Generative models are powerful because they can generate new data that resembles the training data. They can be used for tasks such as image and speech synthesis, language translation, and text generation.

Discriminative models The discriminative model aims to model the conditional distribution of the output variable given the input variable. They learn a decision boundary that separates the different classes of the output variable. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Discriminative models are useful when the focus is on making accurate predictions rather than generating new data. They can be used for tasks such as image recognition, speech recognition, and sentiment analysis.

Graphical models These models use graphical representations to show the conditional dependence between variables. They are commonly used for tasks such as image recognition, natural language processing, and causal inference.

Naive Bayes Algorithm in Probabilistic Models The Naive Bayes algorithm is a widely used approach in probabilistic models, demonstrating remarkable efficiency and effectiveness in solving classification problems. By leveraging the power of the Bayes theorem and making simplifying assumptions about feature independence, the algorithm calculates the probability of the target class given the feature set. This method has found diverse applications across various industries, ranging from spam filtering to medical diagnosis.

Despite its simplicity, the Naive Bayes algorithm has proven to be highly robust, providing rapid results in a multitude of real-world problems. Naive Bayes is a probabilistic algorithm that is used for classification problems. It is based on the Bayes theorem of Probability and assumes that the features are conditionally independent of each other given the class. The Naive Bayes Algorithm is used to calculate the probability of a given sample belonging to a particular class. This is done by calculating the posterior probability of each class given the sample and then selecting the class with the highest posterior probability as the predicted class. The algorithm works as follows:

1. Collect a labeled dataset of samples, where each sample has a set of features and a class label.
2. For each feature in the dataset, calculate the conditional probability of the feature given the class.
3. This is done by counting the number of times the feature occurs in samples of the class and dividing by the total number of samples in the class.
4. Calculate the prior probability of each class by counting the

number of samples in each class and dividing by the total number of samples in the dataset. 5. Given a new sample with a set of features, calculate the posterior probability of each class using the Bayes theorem and the conditional probabilities and prior probabilities calculated in steps 2 and 3. Select the class with the highest posterior probability as the predicted class for the new sample

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Unit – 4 Syllabus: Supervised Learning: Classification: Statistical-based algorithms, Distance-based algorithms, Decision tree-based algorithms, Neural network-based algorithms, Rule-based algorithms, Probabilistic Classifiers. Supervised Learning: Supervised learning, also known as supervised machine learning, is a subcategory of machine learning and artificial intelligence. It is defined by its use of labeled datasets to train algorithms that to classify data or predict outcomes accurately. Supervised learning is utilized in voice recognition to help virtual assistants and other applications recognize and understand spoken commands. A labeled dataset of spoken words and phrases with corresponding text transcripts is used to train a machine learning algorithm in such scenarios. With the rise of big data, supervised learning has become critical for industries such as finance, healthcare, and e-commerce. To appreciate exactly why it has gained such importance, let's first understand what supervised learning is. In simple terms, supervised learning is a standard machine learning technique that involves training a model with labeled data. This blog will explain the fundamentals of supervised learning, its types, algorithms, and applications. We will also go over the steps involved in implementing supervised learning and some of the challenges that come with it. What is Supervised Learning? Supervised learning is a type of machine learning in which a computer algorithm learns to make predictions or decisions based on labeled data. Labeled data is made up of previously known input variables (also known as features) and output variables (also known as labels). By analyzing patterns and relationships between input and output variables in labeled data, the algorithm learns to make predictions. Image and speech recognition, recommendation systems, and fraud detection are all examples of how supervised learning is used. The examples below will help explain what supervised learning is. Examples of Supervised Learning Email Filtering  
Supervised learning is commonly used in email filtering to classify incoming emails as spam or legitimate. A machine learning algorithm is trained using a labeled dataset containing examples of both spam and legitimate emails. The algorithm then extracts relevant information from each email, such as the sender's information, the subject, the message body, and so on. It learns from the labeled dataset to identify patterns and relationships between these features and their corresponding labels (spam or legitimate). Once trained, the algorithm can use the extracted features to predict the label of new, unseen emails. If an email is predicted to be spam, it can be automatically filtered into a spam folder, saving the user's inbox space. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Credit Scoring  
In credit scoring, supervised learning is used to predict the creditworthiness of loan applicants. A labeled dataset containing examples of past loan applicants and their credit history, income, employment status, and other relevant factors is used to train a machine learning algorithm. The algorithm learns to recognize patterns and relationships between these features and their corresponding labels, such as whether or not the loan was repaid. Once trained, the algorithm can predict loan repayment likelihood for new loan applicants based on their input features. Voice Recognition  
Supervised learning is utilized in voice recognition to help virtual assistants and other applications recognize and understand spoken commands. A labeled dataset of spoken words and phrases with corresponding text transcripts is used to train a machine learning algorithm in such scenarios. The algorithm learns to recognize relationships between spoken word audio features such as pitch, amplitude, and frequency and their textual representations from the labeled dataset. Following the training phase, the algorithm can begin analyzing new audio inputs and attempting to transcribe them into text form. This allows virtual assistants to understand and respond to spoken commands like managing reminders, playing music, or controlling smart home devices. Different Types of Supervised Learning Regression Regression is a supervised learning method for determining the relationship between dependent and independent variables. In addition, it employs labeled datasets in an algorithm to forecast continuous output for various data. Here, it is widely used in situations where the output must be a single value, such as weight or height. There are two types of regression: 1. Linear regression: This is used to detect the relationship between two variables and to make future predictions. It is further subdivided according to the number of independent and dependent variables. Simple linear regression, for example, is used when there is only one independent and one dependent variable. Multiple linear regression is used when there are two or more

independent and dependent variables. 2. Logistic regression: Logistic regression is used when the dependent variable is categorical or has binary outputs such as 'yes' or 'no'. Since logistic regression is used to solve binary classification problems, it predicts discrete values for variables. Naive Bayes The Naive Bayes algorithm is well-suited for large datasets because each program in the algorithm operates independently, and the presence of one feature has no effect on the other. Its applications include text Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science classification, and recommendation systems, among others. There are various Naive Bayes models of which the decision tree is commonly used in business. A decision tree, unlike a flowchart, is a supervised learning algorithm composed of control statements containing decisions and their consequences. Iterative Dichotomiser 3 (ID3) and Classification algorithm and Regression Trees (CART) are two popular decision tree algorithms used in a variety of industries.

Classification Classification is a type of supervised learning algorithm which involves the process of accurately assigning data to different categories or classes. In essence, it entails identifying and analyzing specific entities in order to determine the appropriate category or class. K-nearest neighbor, Random forest, Support vector machines, Decision trees, and Linear classifiers are some popular classification algorithms. Neural Networks Neutral Networks perform the process of grouping or categorizing raw data. Additionally, this algorithm is also employed in the interpretation of sensory data and the identification of patterns. The algorithm's use, however, is limited due to the need for high computational resources. Random Forest The random forest algorithm is known as an ensemble method as it combines multiple supervised learning techniques to make a conclusion. Moreover, it uses several decision trees to classify each tree, making it a popular choice in a variety of industries. Steps Involved in Supervised Learning The following are some of the common steps involved in supervised learning:

- Gather labeled data
- Divide the data into two sets: Training and Testing
- Select an appropriate algorithm
- On the training set, train the algorithm
- Analyze the algorithm's performance on the testing set
- If necessary, fine-tune the model to improve performance
- Make predictions on new, unlabeled data using the trained model

Advantages and Disadvantages- When implemented in a professional context, supervised learning can foster a healthy workplace environment that prioritizes ongoing education and supports a culture of continuous growth. Some of its chief advantages include: It gathers previous data, which aids in learning from past mistakes It is a powerful Artificial Intelligence (AI) tool that can handle a wide range of business functions on its own Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science It is a reliable algorithm Some of the drawbacks of supervised learning are: Large data sets tend to be difficult to categorize To operate, a certain level of expertise is required It takes a long time to process To conclude, supervised learning is a well-known machine learning technique used for training models to predict outputs based on input data. With proper model selection and training, supervised learning can be a powerful tool for solving a wide variety of real-world problems. To learn more about these subjects, explore these machine learning and artificial intelligence courses offered by Emeritus in association with the best universities around the world. Supervised and Unsupervised learning

Supervised learning: Supervised learning, as the name indicates, has the presence of a supervisor as a teacher. Basically supervised learning is when we teach or train the machine using data that is well-labelled. Which means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples (data) so that the supervised learning algorithm analyses the training data(set of training examples) and produces a correct outcome from labeled data. For instance, suppose you are given a basket filled with different kinds of fruits. Now the first step is to train the machine with all the different fruits one by one like this: If the shape of the object is rounded and has a depression at the top, is red in color, then it will be labeled as –Apple. If the shape of the object is a long curving cylinder having Green-Yellow color, then it will be labeled as –Banana. Now suppose after training the data, you have given a new separate fruit, say Banana from the basket, and asked to identify it. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Since the machine has already learned the things from previous data and this time has to use it wisely. It will first classify the fruit with its shape and color and would confirm the fruit name as BANANA and put it in the Banana category. Thus the machine learns the things from training data(basket containing fruits) and then applies the knowledge to test data(new fruit). Supervised learning is classified into two categories of algorithms:

- Classification: A classification problem is when the output variable is a category, such as "Red" or "blue", "disease" or "no disease".
- Regression: A regression problem is when the output variable is a real value, such as "dollars" or "weight".

Supervised learning deals with or learns with "labeled" data. This implies that some data is already tagged with the correct answer.

Types:-

- Regression
- Logistic Regression
- Classification
- Naive Bayes
- Classifiers
- K-NN (k nearest neighbors)
- Decision Trees
- Support Vector Machine

Advantages:- Supervised learning allows collecting data and

produces data output from previous experiences. Helps to optimize performance criteria with the help of experience. Supervised machine learning helps to solve various types of real-world computation problems. It performs classification and regression tasks. It allows estimating or mapping the result to a new sample. We have complete control over choosing the number of classes we want in the training data. Disadvantages:- Classifying big data can be challenging. Training for supervised learning needs a lot of computation time. So, it requires a lot of time. Supervised learning cannot handle all complex tasks in Machine Learning. Computation time is vast for supervised learning. It requires a labelled data set.

Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science It requires a training process.

Unsupervised learning Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of the machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data. Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore the machine is restricted to find the hidden structure in unlabeled data by itself. For instance, suppose it is given an image having both dogs and cats which it has never seen. Thus the machine has no idea about the features of dogs and cats so we can't categorize it as 'dogs and cats'. But it can categorize them according to their similarities, patterns, and differences, i.e., we can easily categorize the above picture into two parts. The first may contain all pics having dogs in them and the second part may contain all pics having cats in them. Here you didn't learn anything before, which means no training data or examples. It allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with unlabelled data. Unsupervised learning is classified into two categories of algorithms: Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior. Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y. Types of Unsupervised Learning:- Clustering 1. Exclusive (partitioning) 2. Agglomerative 3. Overlapping 4. Probabilistic Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Clustering Types:- 1. Hierarchical clustering 2. K-means clustering 3. Principal Component Analysis 4. Singular Value Decomposition 5. Independent Component Analysis Supervised vs. Unsupervised Machine Learning: Parameters Supervised machine learning Unsupervised machine learning Input Data Algorithms are trained using labeled data. Algorithms are used against data that is not labeled Computational Simpler method Computationally complex Complexity Accuracy Highly accurate Less accurate No. of classes No. of classes is known No. of classes is not known Data Analysis Uses offline analysis Uses real-time analysis of data Algorithms Linear and Logistics regression, Random K-Means clustering, Hierarchical forest, Support Vector Machine, Neural used clustering, Apriori algorithm, etc. Network, etc. Output Desired output is given. Desired output is not given. Training data Use training data to infer model. No training data is used. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science It does not require training data to be labeled. Dimensionality reduction can be easily accomplished using unsupervised learning. Capable of finding previously unknown patterns in data. Flexibility: Unsupervised learning is flexible in that it can be applied to a wide variety of problems, including clustering, anomaly detection, and association rule mining. Exploration: Unsupervised learning allows for the exploration of data and the discovery of novel and potentially useful patterns that may not be apparent from the outset. Low cost: Unsupervised learning is often less expensive than supervised learning because it doesn't require labeled data, which can be time-consuming and costly to obtain. Disadvantages of unsupervised learning : Difficult to measure accuracy or effectiveness due to lack of predefined answers during training. The results often have lesser accuracy. The user needs to spend time interpreting and label the classes which follow that classification. Lack of guidance: Unsupervised learning lacks the guidance and feedback provided by labeled data, which can make it difficult to know whether the discovered patterns are relevant or useful. Sensitivity to data quality: Unsupervised learning can be sensitive to data quality, including missing values, outliers, and noisy data. Scalability: Unsupervised learning can be computationally expensive, particularly for large datasets or complex algorithms, which can limit its scalability. It is not possible to learn larger and more complex models than with supervised learning. It is possible to learn complex models with unsupervised learning. Complex model Model We can test our model. We cannot test our model. Called as Supervised learning is also called Unsupervised learning is also called classification. clustering. Example Example: Optical character recognition. Example: Find a face in an image. Advantages of unsupervised learning: Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Classification: Statistical-based algorithms- A statistical or data mining algorithm is a mathematical expression of certain aspects of the patterns they find

in data. Different algorithms provide different perspectives on the complete nature of the pattern. There are a variety of different algorithms that can be used for statistical classification, but some of the most popular include support vector machines, logistic regression, and decision trees. Each algorithm has its own strengths and weaknesses, so it is important to choose the right one for your specific problem. Statistical classification deals with rules of case assignment to categories or classes. The classification, or decision rule, is expressed in terms of a set of random variables — the case features. There are two types of statistical-based algorithms which are as follows – Regression – Regression issues deal with the evaluation of an output value located on input values. When utilized for classification, the input values are values from the database and the output values define the classes. Regression can be used to clarify classification issues, but it is used for different applications including forecasting. The elementary form of regression is simple linear regression that includes only one predictor and a prediction. Regression can be used to implement classification using two various methods which are as follows – o Division – The data are divided into regions located on class. o Prediction – Formulas are created to predict the output class's value. Bayesian Classification – Statistical classifiers are used for the classification. Bayesian classification is based on the Bayes theorem. Bayesian classifiers view high efficiency and speed when used to high databases. Bayes Theorem – Let X be a data tuple. In the Bayesian method, X is treated as "evidence." Let H be some hypothesis, including that the data tuple X belongs to a particularized class C. The probability  $P(H|X)$  is decided to define the data. This probability  $P(H|X)$  is the probability that hypothesis H's influence has given the "evidence" or noticed data tuple X.  $P(H|X)$  is the posterior probability of H conditioned on X. For instance, consider the nature of data tuples is limited to users defined by the attribute age and income, commonly, and that X is 30 years old users with Rs. 20,000 income. Assume that H is the hypothesis that the user will purchase a computer. Thus  $P(H|X)$  reverses the probability that user X will purchase a computer given that the user's age and income are acknowledged.  $P(H)$  is the prior probability of H. For instance, this is the probability that any given user will purchase a computer, regardless of age, income, or some other data. The posterior probability  $P(H|X)$  is located on more data than the prior probability  $P(H)$ , which is free of X. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Likewise,  $P(X|H)$  is the posterior probability of X conditioned on H. It is the probability that a user X is 30 years old and gains Rs. 20,000.  $P(H)$ ,  $P(X|H)$ , and  $P(X)$  can be measured from the given information. Bayes theorem supports a method of computing the posterior probability  $P(H|X)$ , from  $P(H)$ ,  $P(X|H)$ , and  $P(X)$ . It is given by  $P(H|X)=P(X|H).P(H) / P(X)$  Statistical Methods in Data Mining Data mining refers to extracting or mining knowledge from large amounts of data. In other words, data mining is the science, art, and technology of discovering large and complex bodies of data in order to discover useful patterns. Theoreticians and practitioners are continually seeking improved techniques to make the process more efficient, cost-effective, and accurate. Any situation can be analyzed in two ways in data mining: Statistical Analysis: In statistics, data is collected, analyzed, explored, and presented to identify patterns and trends. Alternatively, it is referred to as quantitative analysis. Non-statistical Analysis: This analysis provides generalized information and includes sound, still images, and moving images. Descriptive Statistics: The purpose of descriptive statistics is to organize data and identify the main characteristics of that data. Graphs or numbers summarize the data. Average, Mode, SD(Standard Deviation), and Correlation are some of the commonly used descriptive statistical methods. Inferential Statistics: The process of drawing conclusions based on probability theory and generalizing the data. By analyzing sample statistics, you can infer parameters about populations and make models of relationships within data. There are various statistical terms that one should be aware of while dealing with statistics. Some of these are: Population Sample Variable Quantitative Variable Qualitative Variable Discrete Variable Continuous Variable Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Now, let's start discussing statistical methods. This is the analysis of raw data using mathematical formulas, models, and techniques. Through the use of statistical methods, information is extracted from research data, and different ways are available to judge the robustness of research outputs. As a matter of fact, today's statistical methods used in the data mining field typically are derived from the vast statistical toolkit developed to answer problems arising in other fields. These techniques are taught in science curriculums. It is necessary to check and test several hypotheses. The hypotheses described above help us assess the validity of our data mining endeavor when attempting to infer any inferences from the data under study. When using more complex and sophisticated statistical estimators and tests, these issues become more pronounced. For extracting knowledge from databases containing different types of observations, a variety of statistical methods are available in Data Mining and some of these are: Logistic regression analysis Correlation analysis Regression analysis Discriminate analysis Linear discriminant analysis (LDA)

Classification Clustering Outlier detection Classification and regression trees, Correspondence analysis Nonparametric regression, Statistical pattern recognition, Categorical data analysis, Time-series methods for trends and periodicity Artificial neural networks Now, let's try to understand some of the important statistical methods which are used in data mining:

**Linear Regression:** The linear regression method uses the best linear relationship between the independent and dependent variables to predict the target variable. In order to achieve the best fit, make sure that all the distances between the shape and the actual observations at each point are as small as possible. A good fit can be determined by determining that no other position would produce fewer errors given the shape chosen. Simple linear regression and multiple linear regression are the two major types of linear regression. By fitting a linear relationship to the independent variable, the simple linear regression predicts the dependent variable. Using multiple independent variables, multiple linear regression fits the best linear relationship with the dependent variable. For more details, you can refer linear regression.

**Classification:** This is a method of data mining in which a collection of data is categorized so that a greater degree of accuracy can be predicted and analyzed. An effective way to analyze very large datasets is to classify them. Classification is one of several methods aimed at improving the efficiency of the analysis process. A Logistic Regression and a Discriminant Analysis stand out as two major classification techniques.

**Logistic Regression:** It can also be applied to machine learning applications and predictive analytics. In this approach, the dependent variable is either binary (binary regression) or multinomial (multinomial regression): either one of the two or a set of one, two, three, or four options. With a logistic regression equation, one can estimate probabilities regarding the relationship between the independent variable and the dependent variable. For understanding logistic regression analysis in detail, you can refer to logistic regression.

**Discriminant Analysis:** A Discriminant Analysis is a statistical method of analyzing data based on the measurements of categories or clusters and categorizing new observations into one or more populations that were identified a priori. The discriminant analysis models each response class independently then uses Bayes's theorem to flip these projections around to estimate the likelihood of each response category given the value of X. These models can be either linear or quadratic.

**Linear Discriminant Analysis:** According to Linear Discriminant Analysis, each observation is assigned a discriminant score to classify it into a response variable class. By combining the independent variables in a linear fashion, these scores can be obtained. Based on this model, observations are drawn from a Gaussian distribution, and the predictor variables are correlated across all k levels of the response variable, Y.

**Quadratic Discriminant Analysis:** An alternative approach is provided by Quadratic Discriminant Analysis. LDA and QDA both assume Gaussian distributions for the observations of the Y classes. Unlike LDA, QDA considers each class to have its own covariance matrix. As a result, the predictor variables have different variances across the k levels in Y.

**Correlation Analysis:** In statistical terms, correlation analysis captures the relationship between variables in a pair. The value of such variables is usually stored in a column or rows of a database table and represents a property of an object.

**Regression Analysis:** Based on a set of numeric data, regression is a data mining method that predicts a range of numerical values (also known as continuous values). You could, for instance, use regression to predict the cost of goods and services based on other variables. A regression model is used across Chamel Devi Group of Institutions Department of Artificial Intelligence & Data Science numerous industries for forecasting financial data, modeling environmental conditions, and analyzing trends.

The first step in creating good statistics is having good data that was derived with an aim in mind. There are two main types of data: an input (independent or predictor) variable, which we control or are able to measure, and an output (dependent or response) variable which is observed. A few will be quantitative measurements, but others may be qualitative or categorical variables (called factors).

**Distance-based algorithms** Distance based algorithms are methods of classification that each item that is mapped to the same class may be thought of as more similar to other items in the class than it is to the items found in the other classes. An object  $o$  in a data set  $S$  is a distance-based (DB) outlier with parameters  $p$  and  $d$ , i.e.,  $DB(p, d)$ , if minimum a fraction  $p$  of the objects in  $S$  lie at a distance higher than  $d$  from  $o$ . In other words, instead of depending on statistical tests, it can think of distance-based outliers as those objects who do not have enough neighbors. The neighbors are represented based on distance from the given object. In comparison with statistical-based methods, distance-based outlier detection generalizes or merges the ideas behind discordancy testing for standard distributions. Hence, a distance-based outlier is also known as a unified outlier or UO-outlier. Distance-based outlier detection prevents the excessive calculation that can be related to fitting the observed distribution into some standard distribution and in choosing discordancy tests. For some Discordancy tests, it can be displayed that if an object  $o$

is an outlier as per the given test, then o is also a DB (p, d) outlier for some properly represented p and d. For instance, if objects that lie 3 or more standard deviations from the mean are treated to be outliers, considering a normal distribution, then this representation can be “unified” by a DB(0.9988, 0.13s)-an outlier. There are several efficient algorithms for mining distance-based outliers that have been created which are as follows – Index-based algorithm – Given a data set, the index-based algorithm facilitates multidimensional indexing structures, including R-trees or k-d trees, to search for neighbors of each object o inside radius d around that object. Let M be the maximum number of objects within the d-neighborhood of an outlier. Hence, once M + 1 neighbors of object o are discovered, it is accessible that o is not an outlier. This algorithm has the lowest case complexity of O (k \* n<sup>2</sup>), where k is the dimensionality, and n is the number of objects in the data set. Nested-loop algorithm – The nested-loop algorithm has the same evaluation complexity as the index-based algorithm but avoids index structure construction and tries to minimize the number of I/O’s. It divides the memory buffer areas into two halves, and the data is set into several logical blocks. Cell-based algorithm – It can avoid O(n<sup>2</sup>) computational complexity, a cell-based algorithm was developed for memory-resident data sets. Its complexity is O (ek + n), where c is a constant based on the number of cells, and k is the dimensionality. In this method, the data space is partitioned into cells with a side length similar to d/vk. Each cell has two layers surrounding it. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science The first layer is one cell thick, while the second is vk cells thick, rounded up to the closest integer. The algorithm counts outliers on a cell-by-cell instead of an object-by-object basis. For a given cell, it accumulates three counts including the number of objects in the cell, in the cell and the first layer together, and in the cell and both layers together. Decision tree-based algorithmsDecision trees use multiple algorithms to decide to split a node into two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that the purity of the node increases with respect to the target variable. A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node. The following decision tree is for the concept buy\_computer that indicates whether a customer at a company is likely to buy a computer or not. Each internal node represents a test on an attribute. Each leaf node represents a class. The benefits of having a decision tree are as follows – It does not require any domain knowledge. It is easy to comprehend. The learning and classification steps of a decision tree are simple and fast. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Decision Tree Induction Algorithm A machine researcher named J. Ross Quinlan in 1980 developed a decision tree algorithm known as ID3 (Iterative Dichotomiser). Later, he presented C4.5, which was the successor of ID3. ID3 and C4.5 adopt a greedy approach. In this algorithm, there is no backtracking; the trees are constructed in a top-down recursive divide-and-conquer manner. Generating a decision tree form training tuples of data partition D Algorithm : Generate\_decision\_tree Input: Data partition, D, which is a set of training tuples and their associated class labels. attribute\_list, the set of candidate attributes. Attribute selection method, a procedure to determine the splitting criterion that best partitions the data tuples into individual classes. This criterion includes a splitting\_attribute and either a splitting point or splitting subset. Output: A Decision Tree Method create a node N; if tuples in D are all of the same class, C then return N as leaf node labeled with class C; if attribute\_list is empty then return N as leaf node with labeled with majority class in D; || majority voting apply attribute\_selection\_method(D, attribute\_list) to find the best splitting\_criterion; label node N with splitting\_criterion; if splitting\_attribute is discrete-valued and multiway splits allowed then // no restricted to binary trees attribute\_list = splitting\_attribute; // remove splitting attribute for each outcome j of splitting criterion // partition the tuples and grow subtrees for each partition Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science let Dj be the set of data tuples in D satisfying outcome j; // a partition if Dj is empty then attach a leaf labeled with the majority class in D to node N; else attach the node returned by Generate decision tree(Dj, attribute list) to node N; end for return N; Tree Pruning Tree pruning is performed in order to remove anomalies in the training data due to noise or outliers. The pruned trees are smaller and less complex. Tree Pruning Approaches There are two approaches to prune a tree – Pre-pruning – The tree is pruned by halting its construction early. Post-pruning - This approach removes a sub-tree from a fully grown tree. Cost Complexity The cost complexity is measured by the following two parameters – Number of leaves in the tree, and Error rate of the tree. Neural network-based algorithmsNeural Network is an information processing paradigm that is inspired by the human nervous system. As in the Human Nervous system, we have Biological neurons in the same way in Neural networks we have Artificial Neurons which is a Mathematical Function that originates from biological

neurons. The human brain is estimated to have around 10 billion neurons each connected on average to 10,000 other neurons. Each neuron receives signals through synapses that control the effects of the signal on the neuron. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science How Artificial Neural Network Work? Let us Suppose that there are n input like  $X_1, X_2, \dots, X_n$  to a neuron.  $\Rightarrow$  The weight connecting n number of inputs to a neuron are represented by  $[W] = [W_1, W_2, \dots, W_n]$ .  $\Rightarrow$  The Function of summing junction of an artificial neuron is to collect the weighted inputs and sum them up.  $Y_{in} = [X_1 * W_1 + X_2 * W_2 + \dots + X_n * W_n]$   $\Rightarrow$  The output of summing junction may sometimes become equal to zero and to prevent such a situation, a bias of fixed value  $B_0$  is added to it.  $Y_{out} = [X_1 * W_1 + X_2 * W_2 + \dots + X_n * W_n] + B_0$  //  $Y_{in}$  then move toward the Activation Function.  $\Rightarrow$  The output Y of a neuron largely depends on its Activation Function (also known as transfer function).  $\Rightarrow$  There are different types of Activation Function are in use, Such as 1. Identity Function 2. Binary Step Function With Threshold 3. Bipolar Step Function With Threshold 4. Binary Sigmoid Function 5. Bipolar Sigmoid Function Rule-based algorithms Rule-based classifier makes use of a set of IF-THEN rules for classification. We can express a rule in the following form – IF condition THEN conclusion Let us consider a rule R1, R1: IF age = youth AND student = yes THEN buy\_computer = yes Points to remember – The IF part of the rule is called rule antecedent or precondition. The THEN part of the rule is called rule consequent. The antecedent part the condition consist of one or more attribute tests and these tests are logically ANDed. The consequent part consists of class prediction. Note – We can also write rule R1 as follows – R1: (age = youth)  $\wedge$  (student = yes)) (buys computer = yes) Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science If the condition holds true for a given tuple, then the antecedent is satisfied. Rule Extraction Here we will learn how to build a rule-based classifier by extracting IF-THEN rules from a decision tree. Points to remember – To extract a rule from a decision tree – One rule is created for each path from the root to the leaf node. To form a rule antecedent, each splitting criterion is logically ANDed. The leaf node holds the class prediction, forming the rule consequent. Rule Induction Using Sequential Covering Algorithm Sequential Covering Algorithm can be used to extract IF-THEN rules form the training data. We do not require to generate a decision tree first. In this algorithm, each rule for a given class covers many of the tuples of that class. Some of the sequential Covering Algorithms are AQ, CN2, and RIPPER. As per the general strategy the rules are learned one at a time. For each time rules are learned, a tuple covered by the rule is removed and the process continues for the rest of the tuples. This is because the path to each leaf in a decision tree corresponds to a rule. The Following is the sequential learning Algorithm where rules are learned for one class at a time. When learning a rule from a class  $C_i$ , we want the rule to cover all the tuples from class  $C$  only and no tuple form any other class. Algorithm: Sequential Covering Input: D, a data set class-labeled tuples, Att\_vals, the set of all attributes and their possible values. Output: A Set of IF-THEN rules. Method: Rule\_set = {} ; // initial set of rules learned is empty for each class c do repeat Rule = Learn\_One\_Rule(D, Att\_vals, c); remove tuples covered by Rule from D; until termination condition; Rule\_set = Rule\_set + Rule; // add a new rule to rule-set end for return Rule\_Set; Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Rule Pruning The rule is pruned is due to the following reason – The Assessment of quality is made on the original set of training data. The rule may perform well on training data but less well on subsequent data. That's why the rule pruning is required. The rule is pruned by removing conjunct. The rule R is pruned, if pruned version of R has greater quality than what was assessed on an independent set of tuples. FOIL is one of the simple and effective method for rule pruning. For a given rule R,  $FOIL\_Prune = pos - neg / pos + neg$  where pos and neg is the number of positive tuples covered by R, respectively. Note – This value will increase with the accuracy of R on the pruning set. Hence, if the  $FOIL\_Prune$  value is higher for the pruned version of R, then we prune R. Probabilistic Classifiers Probabilistic models are an essential component of machine learning, which aims to learn patterns from data and make predictions on new, unseen data. They are statistical models that capture the inherent uncertainty in data and incorporate it into their predictions. Probabilistic models are used in various applications such as image and speech recognition, natural language processing, and recommendation systems. In recent years, significant progress has been made in developing probabilistic models that can handle large datasets efficiently. Categories Of Probabilistic Models These models can be classified into the following categories: Generative models Discriminative models Graphical models Generative models: Generative models aim to model the joint distribution of the input and output variables. These models generate new data based on the probability distribution of the original dataset. Generative models are powerful because they can generate new data that resembles the training data. They can be used for tasks such as image and speech synthesis, language translation, and text generation. Discriminative models The discriminative model aims to model the conditional distribution of the output

variable given the input variable. They learn a decision boundary that separates the different classes of the output variable. Chameli Devi Group of Institutions Department of Artificial Intelligence & Data Science Discriminative models are useful when the focus is on making accurate predictions rather than generating new data. They can be used for tasks such as image recognition, speech recognition, and sentiment analysis. Graphical models These models use graphical representations to show the conditional dependence between variables. They are commonly used for tasks such as image recognition, natural language processing, and causal inference. Naive Bayes Algorithm in Probabilistic Models The Naive Bayes algorithm is a widely used approach in probabilistic models, demonstrating remarkable efficiency and effectiveness in solving classification problems. By leveraging the power of the Bayes theorem and making simplifying assumptions about feature independence, the algorithm calculates the probability of the target class given the feature set. This method has found diverse applications across various industries, ranging from spam filtering to medical diagnosis. Despite its simplicity, the Naive Bayes algorithm has proven to be highly robust, providing rapid results in a multitude of real-world problems. Naive Bayes is a probabilistic algorithm that is used for classification problems. It is based on the Bayes theorem of Probability and assumes that the features are conditionally independent of each other given the class. The Naive Bayes Algorithm is used to calculate the probability of a given sample belonging to a particular class. This is done by calculating the posterior probability of each class given the sample and then selecting the class with the highest posterior probability as the predicted class. The algorithm works as follows: 1. Collect a labeled dataset of samples, where each sample has a set of features and a class label. 2. For each feature in the dataset, calculate the conditional probability of the feature given the class. 3. This is done by counting the number of times the feature occurs in samples of the class and dividing by the total number of samples in the class. 4. Calculate the prior probability of each class by counting the number of samples in each class and dividing by the total number of samples in the dataset. 5. Given a new sample with a set of features, calculate the posterior probability of each class using the Bayes theorem and the conditional probabilities and prior probabilities calculated in steps 2 and 3. Select the class with the highest posterior probability as the predicted class for the new sample

**Chameli Devi Group of Institutions, Indore**

**Department of Information Technology**

**Subject: Computer Networks (IT502)**

---

## Unit V

Application Layer: WWW and HTTP, FTP, SSH, Email (SMTP, MIME, IMAP), DNS, Network Management (SNMP).

Network Security: Introduction to security, Traditional Ciphers, Modern Ciphers, Message Integrity and Authentication.

---

## **UNIT V APPLICATION LAYER**

### **World Wide Web (WWW)**

What is World Wide Web (WWW, W3)?

The World Wide Web -- also known as the web, WWW or W3 -- refers to all the public websites or pages that users can access on their local computers and other devices through the [internet](#). These pages and documents are interconnected by means of hyperlinks that users click on for information. This information can be in different formats, including text, images, audio and video.

The term *World Wide Web* isn't synonymous with the internet. Rather, the World Wide Web is part of the internet.

### **How does the World Wide Web work?**

Paving the way for an internet revolution that has transformed the world in only three decades, the World Wide Web consists of multiple components that enable users to access various resources, documents and web pages on the internet. Thus, the WWW is like a vast electronic book whose pages are stored or hosted on different servers worldwide.

These pages are the primary component or building blocks of the WWW and are linked through hyperlinks, which provide access from one specific spot in a hypertext or hypermedia document to another spot within that document or a different one. Hyperlinks are another defining concept of the WWW and provide its identity as a collection of interconnected documents.

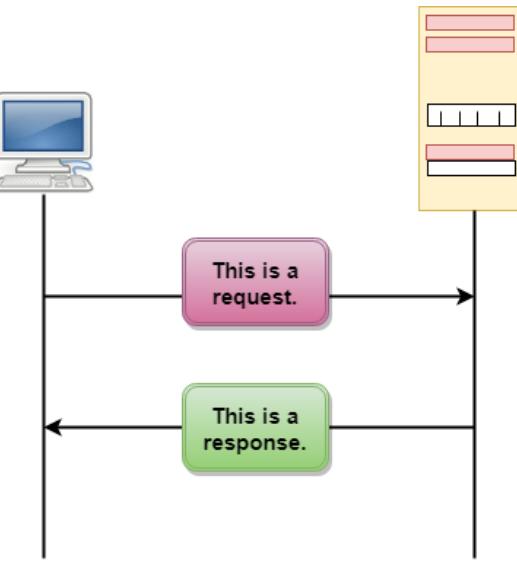
### **HTTP**

- HTTP stands for HyperText Transfer Protocol.
- It is a protocol used to access the data on the World Wide Web (www).
- The HTTP protocol can be used to transfer the data in the form of plain text, hypertext, audio, video, and so on.
- This protocol is known as HyperText Transfer Protocol because of its efficiency that allows us to use in a hypertext environment where there are rapid jumps from one document to another document.
- HTTP is similar to the FTP as it also transfers the files from one host to another host. But, HTTP is simpler than FTP as HTTP uses only one connection, i.e., no control connection to transfer the files.

### **Features of HTTP:**

- Connectionless protocol: HTTP is a connectionless protocol. HTTP client initiates a request and waits for a response from the server. When the server receives the request, the server processes the request and sends back the response to the HTTP client after which the client disconnects the connection. The connection between client and server exist only during the current request and response time only.
- Media independent: HTTP protocol is a media independent as data can be sent as long as both the client and server know how to handle the data content. It is required for both the client and server to specify the content type in MIME-type header.
- Stateless: HTTP is a stateless protocol as both the client and server know each other only during the current request. Due to this nature of the protocol, both the client and server do not retain the information between various requests of the web pages.

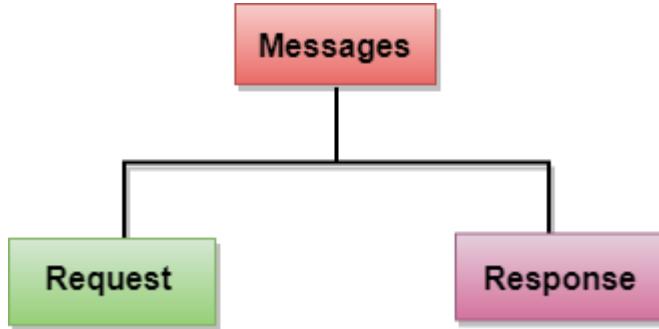
### **HTTP Transactions**



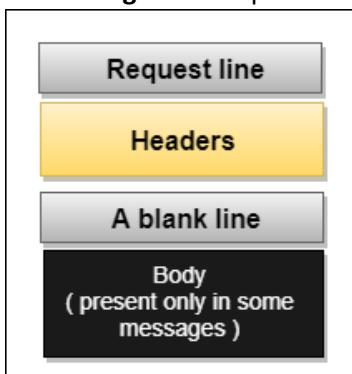
The above figure shows the HTTP transaction between client and server. The client initiates a transaction by sending a request message to the server. The server replies to the request message by sending a response message.

## Messages

HTTP messages are of two types: request and response. Both the message types follow the same message format.

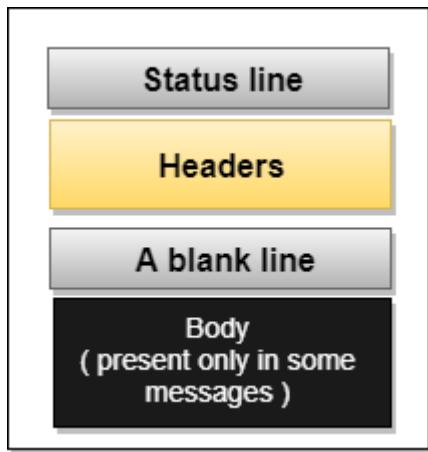


**Request Message:** The request message is sent by the client that consists of a request line, headers, and sometimes a



body.

**Response Message:** The response message is sent by the server to the client that consists of a status line, headers, and sometimes a body.



## FTP

FTP stands for File Transfer Protocol, and it is basically the protocol, or procedure, for transferring files between computers. Files are not actually moved over to the source system, rather, they are copied from one computer to another. This exchange of files happens over Internet channels, formally referred to as a TCP/IP network. TCP/IP (Transmission Control Protocol/Internet Protocol) stands for the standard protocol of communication over the internet. FTP is a standard protocol, and any computer with the ability to use FTP can make use of the protocol.

## CONNECTION

For FTP to work there needs to be a client and a server. The client connects to the service. The client and server communicate back and forth, the server must validate that the client has access to parts of the server it is trying to access and then the client copies the files to the directories.

### Connection Modes: Active vs. Passive.

FTP may work in either active or passive mode, which sets how the data connection is setup. Each type involves the client creating a connection to the FTP server on a certain port (usually port 21). A port is a number assigned to servers in a TCP/IP network. They simply indicate the purpose of the data being transferred (e.g., web page, voice call). Servers will monitor a given port to know when data starts coming in. FTP servers usually watch port 21.

#### Active Mode

In active mode, the client connects, indicating that it will be monitoring port 21 for incoming data from the server. Behind the scenes, the client sends a command PORT to tell the server which port it's on. However, most clients are now behind firewalls, either personal or corporate, and the firewall doesn't let in incoming connections. For that purpose, passive mode will work better.

Active mode requires most of the setup to be completed on the client side. If using a firewall, it must be configured to allow ports for incoming connections.

#### Passive Mode

In passive mode, the client sends a PASV command to the server, the server then sends back an IP (Internet) address and port number. The client can use this connection to the server and data transfer/copying can occur.

For this connection mode, most of the work is on the server side. The server needs to setup the system so that it can accept incoming data, not only from port 21 but from a range of possible ports for incoming connections.

FTP uses TCP services. It needs two TCP connections. One is Control connection and another is Data connection. For control connection, it uses well-known port 21 and for data connection, it uses well-known port 20

### Control Connection

A server site control connection uses a well-known port 21. There are two steps to establish a control connection –

- Server issues a passive open on the well-known port 21 and waits for the client
- After severing issues passive open, the client issue active open using an ephemeral port.

This control connection remains open throughout the process. Since the user and the server uses the interactive connection for communication, their service used by internet protocol minimizes the delay. For communication, user types the command and in return, servers give responses without any delay.

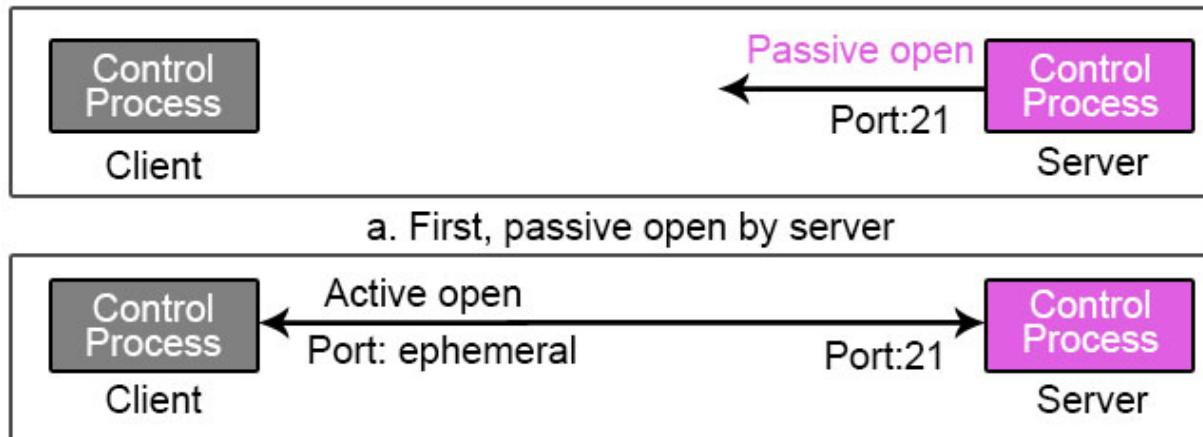
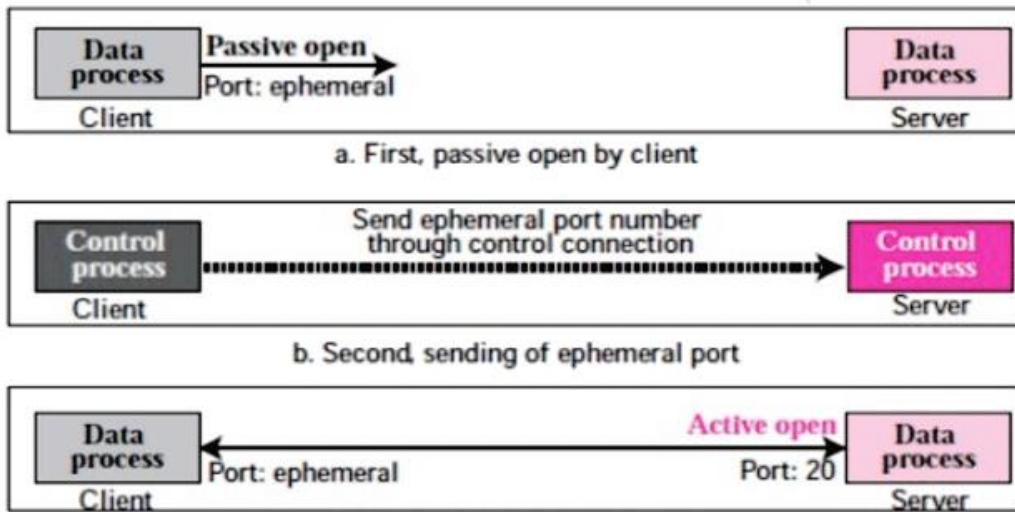


Figure 5.6: FTP Connection

### Control Connection

Data Connections: At the server site, the data connection uses well-known port 20. There are three steps to establish a data connection –

- Using ephemeral port client issues a passive open. This step must be done by the client not the server because the client wants to transform the file.
- Using the PORT command client sends this port number to the server.
- When the server receives this port number from the client, it issues active open using well-known port 20.



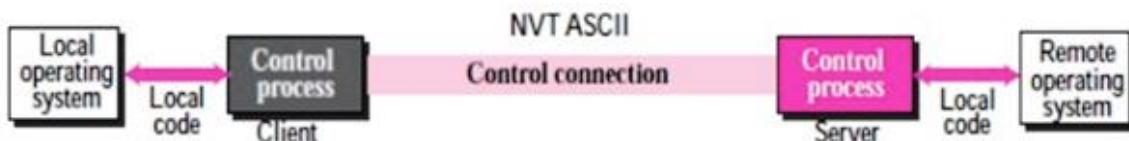
**Figure 5.7: FTP Control Connection**

## COMMUNICATION

Both the client and the server which runs on two different systems must be communicated for transforming data. For communication, it uses two approaches i.e. communication over control connection and communication over a data connection.

### Communication over the control connection

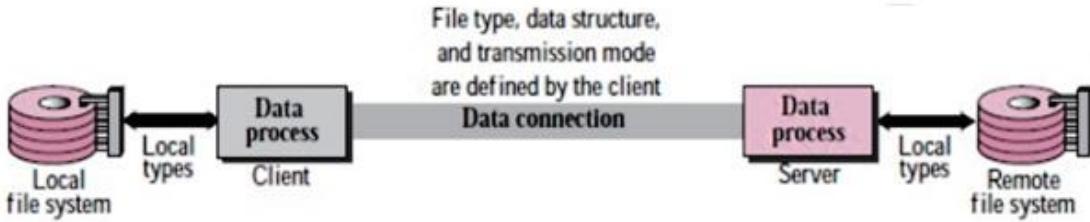
To communicate over control connection FTP uses TELNET or SMTP. Communication over control connection is done by commands and responses. The first command is sent over the connection and in return, a response is sent by another system. We can send a command or response at a time. There is only one-way communication.



**Figure 5.8: Communication over the control connection**

### Communication over the data connection

For transforming file over the data connection, the client must define the type of file which needs to be transformed, transmission mode, and the data structure. It solves the heterogeneity problem by defining these three attributes.



**Figure 5.9: Communication over the data connection**

## COMMAND PROCESSING

To establish communication between the client system and the server system FTP uses a control connection. During this process, the client sends commands to the server and in return, the server sends a response to the client.

### Types of FTP Transfers

It can transfer following file types across the internet connections

1. **ASCII file:** This is the default format for transforming a file from one to another. Each character is encoded by NVT ASCII i.e. Network Virtual Terminal ASCII character set. Both the sender and the receiver transform their file from its own representation into NVT ASCII.
2. **EBCDIC:** If sender or receiver connections use the EBCDIC encoding method, then for transforming file FTP uses EBCDIC encoding.
3. **Image File:** For transforming the binary file, the image file is the default mode. The file is transformed over the internet connections in the form of stream bits without encoding.

## TFTP

Trivial File Transfer Protocol, is a simple high-level protocol for transferring data servers use to boot diskless workstations, X-terminals, and routers by using User Data Protocol (UDP).

Although it may sound similar, TFTP works differently than FTP (File Transfer Protocol) and HTTP (HyperText Transfer Protocol). Although TFTP is also based in FTP technology, TFTP is an entirely different protocol. Among the differences is that TFTP's transport protocol uses UDP which is not secure while FTP uses Transmission Control Protocol (TCP) to secure information.

TFTP was primarily designed to read or write files by using a remote server. However, TFTP is a multi-purpose protocol that can be leveraged for an array of different tasks.

### TFTP Configuration Uses

TFTP configuration used for:

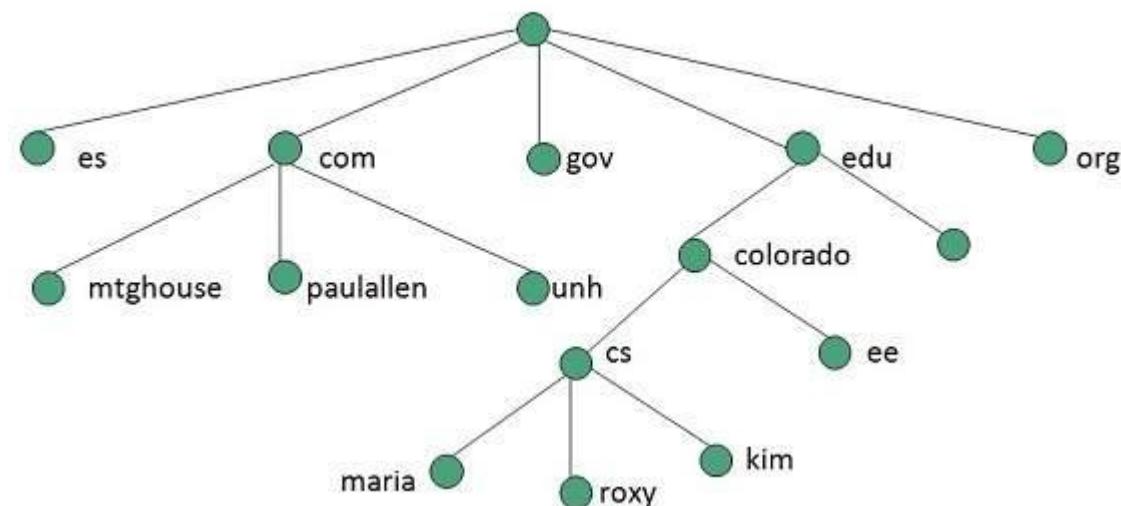
- Transferring files
- Remote-booting without hard drives
- Upgrading codes

- Backing up network configurations
- Backing up router configuration files
- Saving IOS images
- Booting PCs without a disk

After a workstation has been booted from a network card's ROM, the TFTP installation will download a program and then run it from a central server.

## DNS

The domain name space refers a hierarchy in the internet naming structure. This hierarchy has multiple levels (from 0 to 127), with a root at the top. The following diagram shows the domain name space hierarchy:



**Figure 5.5: Domain name space**

DNS is an industry-standard service used to locate computers on an Internet Protocol (IP)-based network. IP networks, such as the Internet and Windows 2000 networks, rely on number-based addresses, such as 207.46.131.137 to ferry information throughout the network. Network users rely on character-based names, such as www.microsoft.com. Therefore, it is necessary to translate character or user-friendly addresses (www.microsoft.com) into the number-based addresses (207.46.131.137) that the network can recognize.

DNS replaced the hosts file's flat name space with a hierarchical name space. With a hierarchical name space, information about host names and IP addresses can be partitioned and distributed, thus, scalability is achieved. For example, in the fictional widgets.products.microsoft.com domain, responsibility for name resolution can be partitioned so that various servers can handle name resolution for different parts of the name space:

- One server can be responsible for resolving the first part (microsoft.com), and can then forward the name-resolution request to the next DNS Server in the hierarchy.
- The next DNS Server can be responsible for resolving the next part of the name space.
- Finally, the request can be forwarded to a third DNS server that is responsible for resolving the last part of the name.

## DISTRIBUTION OF NAME SPACES

A namespace is a context within which the names of all objects must be unambiguously resolvable. For example, the internet is a single DNS name space, within which all network devices with a DNS name can be resolved to a particular address (for example, www.microsoft.com resolves to 207.46.131.13).

## DNS IN THE INTERNET

DNS translates the domain name into IP address automatically. Following are the steps of domain resolution process:

- When user type www.cdgi.com into the browser, it asks the local DNS Server for its IP address.
- When the local DNS does not find the IP address of requested domain name, it forwards the request to the root DNS server and again enquires about IP address of it.
- The root DNS server replies with delegation that user do not know the IP address of www.cdgi.com but know the IP address of DNS Server.
- The local DNS server then asks the com DNS Server the same question.
- The com DNS Server replies the same that it does not know the IP address of www.cdgi.com but knows the address of cdgi.com.
- Then the local DNS asks the cdgi.com DNS server the same question.
- Then cdgi.com DNS server replies with IP address of www.cdgi.com.
- Now, the local DNS sends the IP address of www.cdgi.com to the computer that sends the request.

## SSH Protocol

SSH stands for Secure Shell or Secure Socket Shell. It is a cryptographic network protocol that allows two computers to communicate and share the data over an insecure network such as the internet. It is used to login to a remote server to execute commands and data transfer from one machine to another machine.

A simple example can be understood, such as suppose you want to transfer a package to one of your friends. Without SSH protocol, it can be opened and read by anyone. But if you will send it using SSH protocol, it will be encrypted and secured with the public keys, and only the receiver can open it.

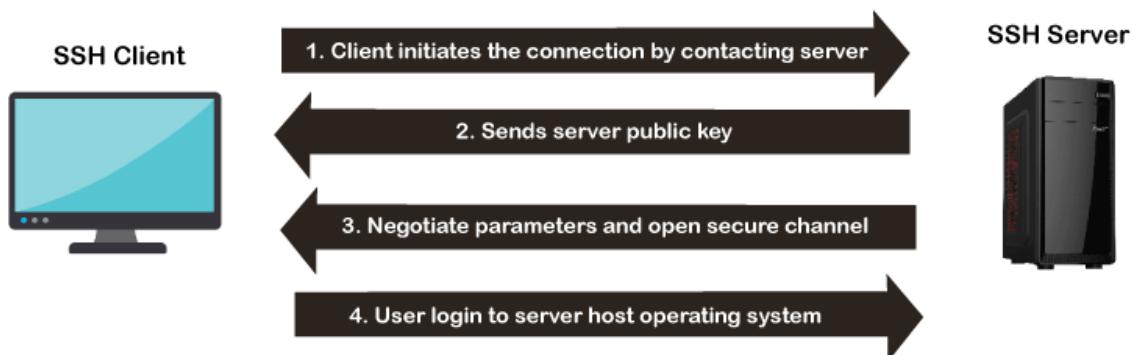
### Usages of SSH protocol

- It provides secure access to users and automated processes.
- It is an easy and secure way to transfer files from one system to another over an insecure network.
- It also issues remote commands to the users.
- It helps the users to manage the network infrastructure and other critical system components

- It is used to log in to shell on a remote system (Host), which replaces **Telnet** and **rlogin** and is used to execute a single command on the host, which replaces **rsh**.
- It combines with **rsync** utility to backup, copy, and mirror files with complete security and efficiency.
- It can be used for forwarding a port.
- By using SSH, we can set up the automatic login to a remote server such as OpenSSH.
- We can securely browse the web through the encrypted proxy connection with the SSH client, supporting the SOCKS protocol.

### How does SSH Works?

The SSH protocol works in a **client-server** model, which means it connects a secure shell client application (End where the session is displayed) with the SSH server (End where session executes).



## **E-MAIL**

### **SMTP**

SMTP stands for Simple Mail Transfer Protocol and it's the industry standard protocol for email sending.

With SMTP users are sending, relaying, or forwarding messages from a mail client (like Microsoft Outlook) to a receiving email server. A sender will use an SMTP server to carry out the process of transmitting an email message.

If user looking to enable email sending within the application, then he must use SMTP over IMAP.

### **MIME(Multipurpose Internet Mail Extension Protocol):**

Multipurpose Internet Mail Extension Protocol is an additional email protocol that allows non-ASCII data to be sent through SMTP. It allows users to send and receive different types of data like audio, images, videos and other application programs on the Internet. It allows to send multiple attachments with single message. It allows to send message of unlimited length.

### **POP**

POP stands for Post Office Protocol. And the number three stands for "version 3," which is the latest version and the most widely used — hence the term "POP3."

POP3 downloads the email from a server to a single computer, and then deletes the email from the server. On the other hand, IMAP stores the message on a server and synchronizes the message across multiple devices.

### **IMAP**

IMAP (Internet Access Message Protocol) is an email protocol that deals with managing and retrieving email messages from the receiving server. Since IMAP deals with message retrieval, user will not be able to use the IMAP protocol to send email. Instead, IMAP will be used for receiving messages.

### **SNMP**

Simple Network Management Protocol (SNMP) is a way for different devices on a network to share information with one another. It allows devices to communicate even if the devices are different hardware and run different software. Without a protocol like SNMP, there would be no way for network management tools to identify devices, monitor network performance, keep track of changes to the network, or determine the status of network devices in real time.

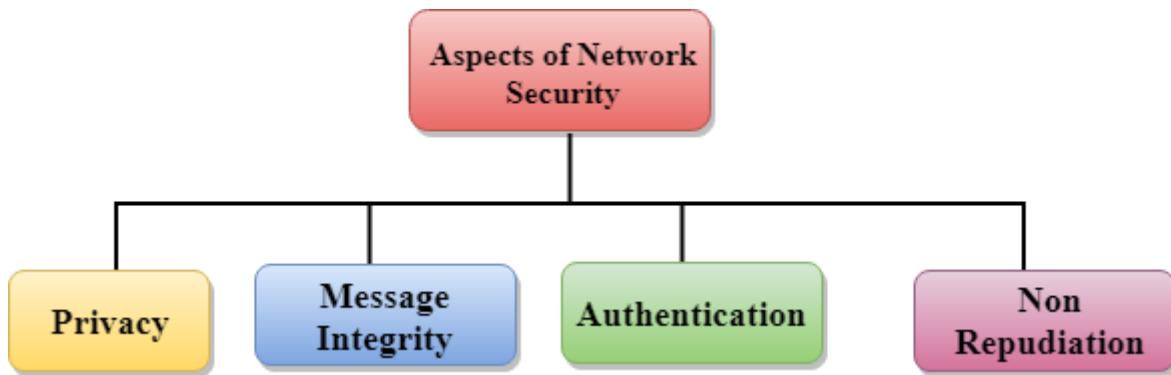
SNMP has a simple architecture based on a client-server model. The servers, called managers, collect and process information about devices on the network.

The clients, called agents, are any type of device or device component connected to the network. They can include not just computers but also network switches, phones, printers. Some devices may have multiple device components. For example, a laptop typically contains a wired as well as a wireless network interface.

All the measures used to safeguard a computer network's integrity and the data on it are collectively referred to as network security. Network security is crucial because it protects sensitive data from online threats and guarantees the network's dependability. Multiple security measures are used in successful network security plans to shield users and organizations from malware and online threats like distributed denial of service.

#### Aspects of Network Security

Following are the desirable properties to achieve secure communication:



#### Cryptography –

**Traditional Ciphers**:- Traditional ciphers encode messages in a way that only those with the secret key can decode it. Throughout history, it has been used for private communication.

These ciphers were used extensively before the advent of modern computer encryption techniques. Although relatively simple, when used properly, it can also be effective and provide interesting insights into the history of encryption.

#### Here are a few common types –

- Caesar Cipher
- Vigenere Cipher
- Simple Substitution Cipher
- Monoalphabetic and Polyalphabetic Cipher
- Transposition Cipher
- Playfair Cipher

#### Caesar Cipher

Caesar cipher is a type of substitution cipher in which every letter in the given plaintext is shifted with a certain number of places down or up the alphabet. The shift number is called the key. It is created by Julius Caesar. It is believed that this cipher is used to communicate secretly with his generals.

For this type of method both the sender and the receiver should agree on a 'secret shift key' for shifting the alphabet of the given plaintext. The number which can be between 0 and 25 is the key of encryption.

For example - If plaintext is shift of 4:

- 'A' will become 'E'
- 'B' will become 'F'
- 'C' will become 'G'
- and so on...

### **Vigenere Cipher**

Vigenere is also a traditional cipher algorithm which uses a text string (Let us say a word) as a key. Then it is used for shifting a number of shifts on the given plaintext.

For example – let us say the key is 'tutor'. In this each alphabet of the key is changed to its respective numeric value: In our case,

$t \rightarrow 20, u \rightarrow 21, t \rightarrow 20, o \rightarrow 15, \text{ and } r \rightarrow 18.$

Thus, the key is: 20 21 20 15 18.

### **Simple Substitution Cipher**

Simple substitution cipher is a form of encryption in which the letters in Plaintext is replaced by another character in cipher text according to a fixed order. In other words it is a method of encoding with any letter and characters are mapped to other characters.

For example, in Simple Substitution Cipher –

- 'A' can be replaced by 'X'
- 'B' can be replaced by 'K'
- 'C' can be replaced by 'Q'
- So on... until each character in the alphabet is replaced by another one.

### **Monoalphabetic and Polyalphabetic Cipher:**

A Monoalphabetic cipher is a type of substitution cipher in which fixed characters are always replaced in the ciphertext which means that every character in the plaintext is always replaced by the same corresponding character in the ciphertext. The replacement remains constant throughout the encryption process.

For example, if 'A' is replaced by 'E' in the encryption process, any occurrence of 'A' in plain text will be replaced by 'E' in ciphertext and if 'B' is replaced by 'F'. replace 'any B ' in plaintext with 'F' in ciphertext.

### Transposition Cipher

This is a new type of cipher in which the sequence of letters is rearranged in plain text to create ciphertext. They are not replaced by actual plaintext.

An example is a simple column substitution cipher in which simple increments are written in a specific alphabet width. The ciphertext is then read directly as indicated.

For example, the plain text is "The tajmahal is in agra" and the randomly chosen hidden key is "Four". We format this text directly in a table with a number of columns including key value. The resulting text is shown below.

T	h	e	t
a	j	m	a
h	a	l	i
s	i	n	a
g	r	a	

So now we can get the ciphertext just reading columns from first to last by vertically downward. And the ciphertext will be 'Tahsghjairemlnataia'.

### Playfair Cipher

The Playfair cipher uses a 5x5 grid of letters unless it is double and usually 'J' for encryption. Messages are encrypted by pairing letters and using rules: if on the same letter, turn right; If it is in one column, turn down; If there are rows and columns, make a triangle and replace them with opposite corners. Decryption follows the opposite rule. Playfair encrypts character pairs, making it more secure than single-character ciphers, but still vulnerable to an attack.

### Modern Ciphers - Base64 Encoding & Decoding

Details of Base64 encoding

Base64 refers to a group of related encoding techniques that encode binary data numerically and translate it into a base-64 representation. The word Base64 comes from a specific MIME-content transfer encoding.

Base64 has a specific set of characters –

- 26 Uppercase letters

- 26 Lowercase letters
- 10 Numbers
- Plus sign (+) and slash (/) for new lines

## Applications

Base64 encoding is often used in a variety of applications. It is widely used to encrypt binary data, especially if it needs to be communicated by email or used in other text fields. It is also used in a variety of web and internet protocols, as well as to encode digital signatures and certificates.

## Limitations

Here are the limitations of Base64 encoding –

- Base64 encoding usually maximizes the size of the data by about 33%. This can impact transmission and storage efficiency, mainly for large datasets.
- It is basically not a form of encryption. It simply converts data into a different format. As a result, sensitive information is not kept confidential or secure.
- Base64 adds padding characters ('=') to the end of the encoded data to ensure it aligns properly. This way it can complicate parsing and processing of the encoded data.
- It uses a limited set of characters (A-Z, a-z, 0-9, '+', '/') for encoding. This can lead to issues when the encoded data needs to be transmitted or processed in systems that have restrictions on certain characters.
- The encoding of Base64 does not compress data. It is only meant for representing binary data in a text-based format, not for reducing file size.

## Message Integrity in Cryptography:-

Message integrity in cryptography is the process of checking the message's authenticity. It ensures the message has not been altered or tampered with. Message integrity means that a message has not been tampered with or manipulated. Integrity is important because it ensures that the message has not been modified or tampered with.

Message integrity is commonly used in computing systems for integrity verification and information authentication. They are regarded cryptographically “weak” since they can be solved in polynomial time but are difficult to interpret.

Message integrity enhances traditional hash algorithms with security characteristics, making it more difficult to discover message content or receiver and sender information.

## Steps to Verify the Integrity of a Message

- **Message Authentication Codes:** Suppose two users, a sender, and a receiver, want to connect via messages. In MAC, or Message Authentication Codes, the transmitter and receiver use the same MAC algorithm or key.

- **Certificates:** A certificate is a digital document that validates a public key. The certificate provides information about the key, the owner's identity, and the organization's digital signature, which has verified the certificate's contents.
- **Nonrepudiation:** Nonrepudiation is the property of agreeing to adhere to an obligation. More specifically, it is the inability to refute responsibility.
- **Message Authentication Codes:** In the case of MAC, there is no public key. There is just one private key, which is known only to the sender and receiver. As a result, there is no interference from external parties. Even if a third-party user had access to the secret key, he could not guarantee that either the sender or the recipient signed the message because both can encrypt or decrypt it.

### Message authentication

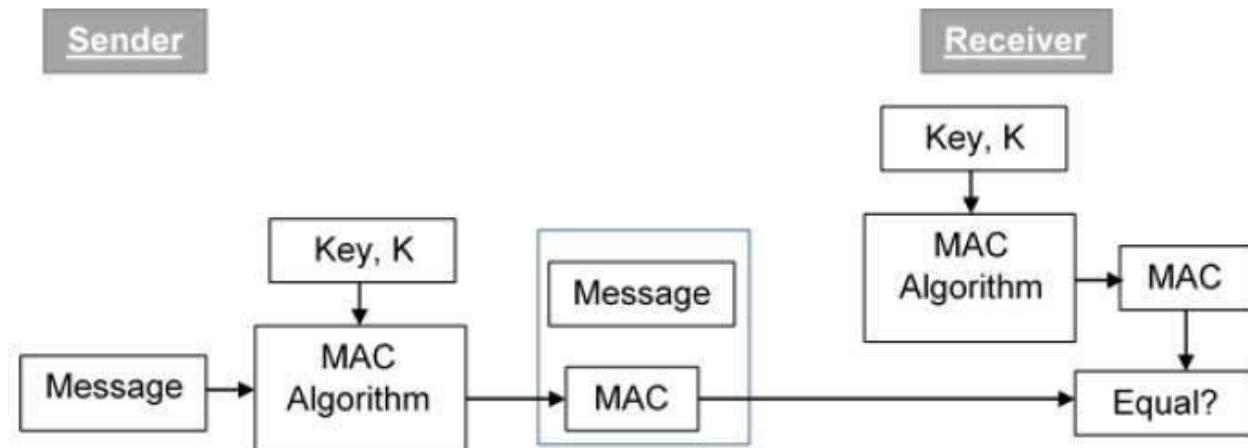
Message authentication can be provided using the cryptographic techniques that use secret keys as done in case of encryption.

Message Authentication Code (MAC):

MAC algorithm is a symmetric key cryptographic technique to provide message authentication. For establishing MAC process, the sender and receiver share a symmetric key K.

Essentially, a MAC is an encrypted checksum generated on the underlying message that is sent along with a message to ensure message authentication.

The process of using MAC for authentication is depicted in the following illustration –



Let us now try to understand the entire process in detail –

- The sender uses some publicly known MAC algorithm, inputs the message and the secret key K and produces a MAC value.
- Similar to hash, MAC function also compresses an arbitrary long input into a fixed length output. The major difference between hash and MAC is that MAC uses secret key during the compression.
- The sender forwards the message along with the MAC. Here, we assume that the message is sent in the clear, as we are concerned of providing message origin authentication, not confidentiality. If confidentiality is required then the message needs encryption.
- On receipt of the message and the MAC, the receiver feeds the received message and the shared secret key K into the MAC algorithm and re-computes the MAC value.
- The receiver now checks equality of freshly computed MAC with the MAC received from the sender. If they match, then the receiver accepts the message and assures himself that the message has been sent by the intended sender.
- If the computed MAC does not match the MAC sent by the sender, the receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified. As a bottom-line, a receiver safely assumes that the message is not the genuine.

**Chameli Devi Group of Institutions, Indore**

**Department of Information Technology**

**Subject: Computer Networks (IT502)**

---

## **Unit V**

Application Layer: WWW and HTTP, FTP, SSH, Email (SMTP, MIME, IMAP), DNS, Network Management (SNMP).

Network Security: Introduction to security, Traditional Ciphers, Modern Ciphers, Message Integrity and Authentication.

---

## **UNIT V APPLICATION LAYER**

### **World Wide Web (WWW)**

What is World Wide Web (WWW, W3)?

The World Wide Web -- also known as the web, WWW or W3 -- refers to all the public websites or pages that users can access on their local computers and other devices through the [internet](#). These pages and documents are interconnected by means of hyperlinks that users click on for information. This information can be in different formats, including text, images, audio and video.

The term *World Wide Web* isn't synonymous with the internet. Rather, the World Wide Web is part of the internet.

### **How does the World Wide Web work?**

Paving the way for an internet revolution that has transformed the world in only three decades, the World Wide Web consists of multiple components that enable users to access various resources, documents and web pages on the internet. Thus, the WWW is like a vast electronic book whose pages are stored or hosted on different servers worldwide.

These pages are the primary component or building blocks of the WWW and are linked through hyperlinks, which provide access from one specific spot in a hypertext or hypermedia document to another spot within that document or a different one. Hyperlinks are another defining concept of the WWW and provide its identity as a collection of interconnected documents.

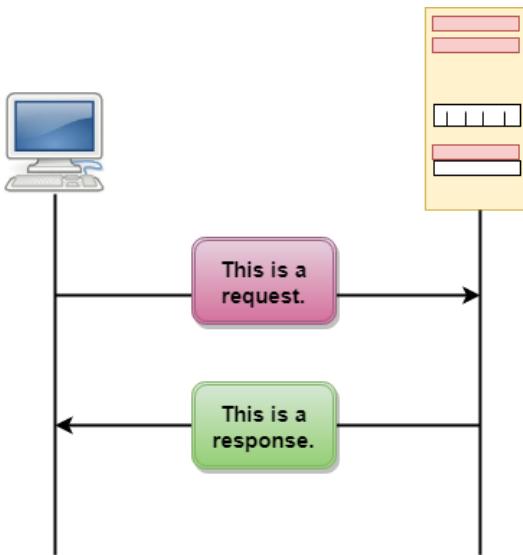
### **HTTP**

- HTTP stands for HyperText Transfer Protocol.
- It is a protocol used to access the data on the World Wide Web (www).
- The HTTP protocol can be used to transfer the data in the form of plain text, hypertext, audio, video, and so on.
- This protocol is known as HyperText Transfer Protocol because of its efficiency that allows us to use in a hypertext environment where there are rapid jumps from one document to another document.
- HTTP is similar to the FTP as it also transfers the files from one host to another host. But, HTTP is simpler than FTP as HTTP uses only one connection, i.e., no control connection to transfer the files.

### **Features of HTTP:**

- Connectionless protocol: HTTP is a connectionless protocol. HTTP client initiates a request and waits for a response from the server. When the server receives the request, the server processes the request and sends back the response to the HTTP client after which the client disconnects the connection. The connection between client and server exist only during the current request and response time only.
- Media independent: HTTP protocol is a media independent as data can be sent as long as both the client and server know how to handle the data content. It is required for both the client and server to specify the content type in MIME-type header.
- Stateless: HTTP is a stateless protocol as both the client and server know each other only during the current request. Due to this nature of the protocol, both the client and server do not retain the information between various requests of the web pages.

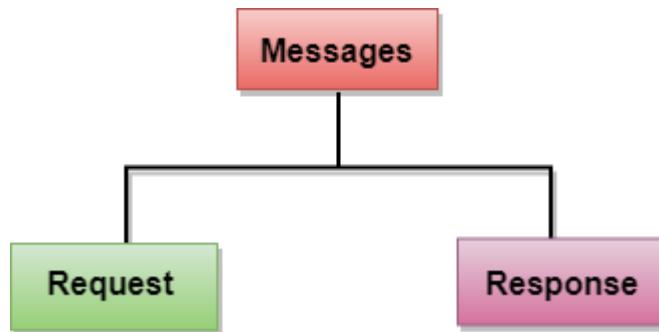
### **HTTP Transactions**



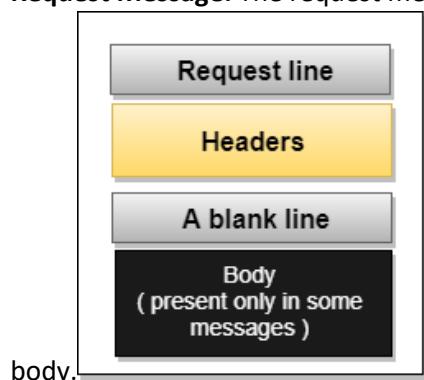
The above figure shows the HTTP transaction between client and server. The client initiates a transaction by sending a request message to the server. The server replies to the request message by sending a response message.

## Messages

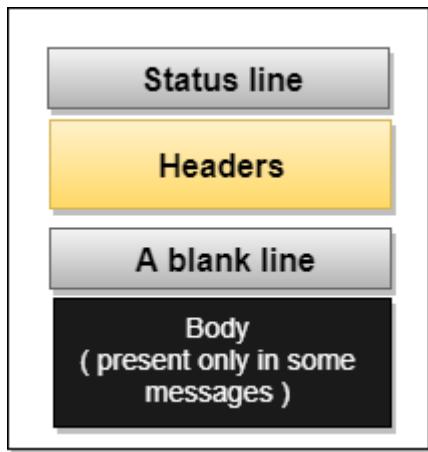
HTTP messages are of two types: request and response. Both the message types follow the same message format.



**Request Message:** The request message is sent by the client that consists of a request line, headers, and sometimes a



**Response Message:** The response message is sent by the server to the client that consists of a status line, headers, and sometimes a body.



## FTP

FTP stands for File Transfer Protocol, and it is basically the protocol, or procedure, for transferring files between computers. Files are not actually moved over to the source system, rather, they are copied from one computer to another. This exchange of files happens over Internet channels, formally referred to as a TCP/IP network. TCP/IP (Transmission Control Protocol/Internet Protocol) stands for the standard protocol of communication over the internet. FTP is a standard protocol, and any computer with the ability to use FTP can make use of the protocol.

## CONNECTION

For FTP to work there needs to be a client and a server. The client connects to the service. The client and server communicate back and forth, the server must validate that the client has access to parts of the server it is trying to access and then the client copies the files to the directories.

### Connection Modes: Active vs. Passive.

FTP may work in either active or passive mode, which sets how the data connection is setup. Each type involves the client creating a connection to the FTP server on a certain port (usually port 21). A port is a number assigned to servers in a TCP/IP network. They simply indicate the purpose of the data being transferred (e.g., web page, voice call). Servers will monitor a given port to know when data starts coming in. FTP servers usually watch port 21.

#### Active Mode

In active mode, the client connects, indicating that it will be monitoring port 21 for incoming data from the server. Behind the scenes, the client sends a command PORT to tell the server which port it's on. However, most clients are now behind firewalls, either personal or corporate, and the firewall doesn't let in incoming connections. For that purpose, passive mode will work better.

Active mode requires most of the setup to be completed on the client side. If using a firewall, it must be configured to allow ports for incoming connections.

#### Passive Mode

In passive mode, the client sends a PASV command to the server, the server then sends back an IP (Internet) address and port number. The client can use this connection to the server and data transfer/copying can occur.

For this connection mode, most of the work is on the server side. The server needs to setup the system so that it can accept incoming data, not only from port 21 but from a range of possible ports for incoming connections.

FTP uses TCP services. It needs two TCP connections. One is Control connection and another is Data connection. For control connection, it uses well-known port 21 and for data connection, it uses well-known port 20

### Control Connection

A server site control connection uses a well-known port 21. There are two steps to establish a control connection –

- Server issues a passive open on the well-known port 21 and waits for the client
- After severing issues passive open, the client issue active open using an ephemeral port.

This control connection remains open throughout the process. Since the user and the server uses the interactive connection for communication, their service used by internet protocol minimizes the delay. For communication, user types the command and in return, servers give responses without any delay.

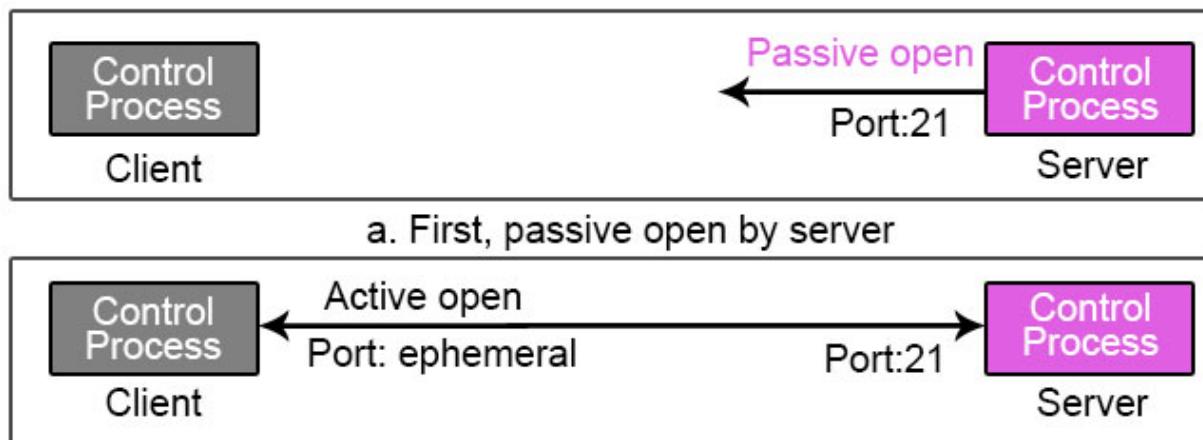
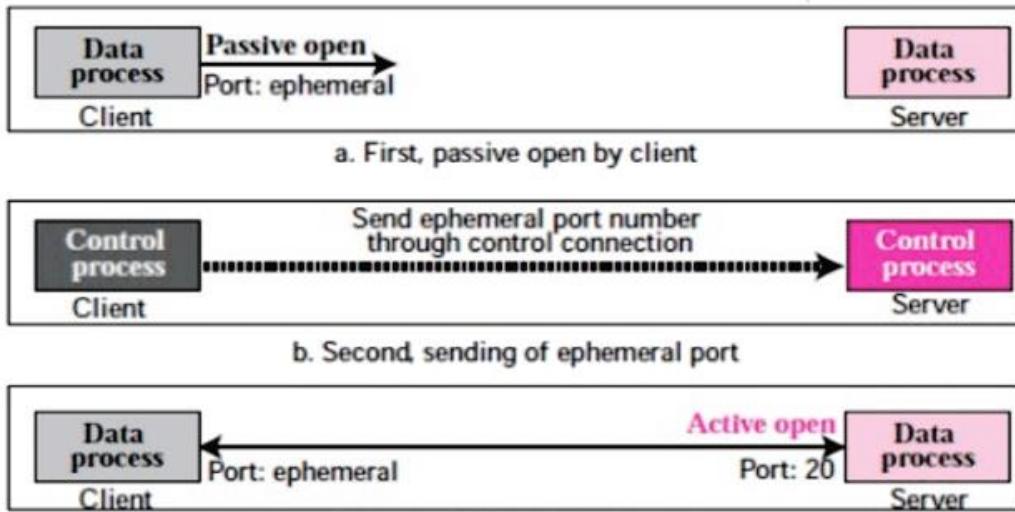


Figure 5.6: FTP Connection

### Control Connection

Data Connections: At the server site, the data connection uses well-known port 20. There are three steps to establish a data connection –

- Using ephemeral port client issues a passive open. This step must be done by the client not the server because the client wants to transform the file.
- Using the PORT command client sends this port number to the server.
- When the server receives this port number from the client, it issues active open using well-known port 20.



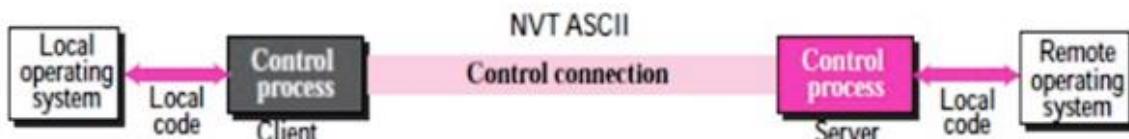
**Figure 5.7: FTP Control Connection**

## COMMUNICATION

Both the client and the server which runs on two different systems must be communicated for transforming data. For communication, it uses two approaches i.e. communication over control connection and communication over a data connection.

### Communication over the control connection

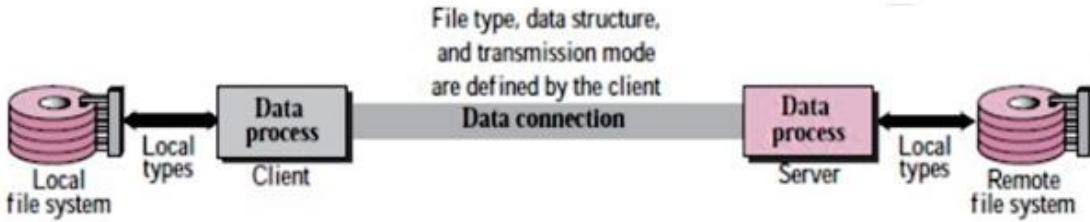
To communicate over control connection FTP uses TELNET or SMTP. Communication over control connection is done by commands and responses. The first command is sent over the connection and in return, a response is sent by another system. We can send a command or response at a time. There is only one-way communication.



**Figure 5.8: Communication over the control connection**

### Communication over the data connection

For transforming file over the data connection, the client must define the type of file which needs to be transformed, transmission mode, and the data structure. It solves the heterogeneity problem by defining these three attributes.



**Figure 5.9: Communication over the data connection**

## COMMAND PROCESSING

To establish communication between the client system and the server system FTP uses a control connection. During this process, the client sends commands to the server and in return, the server sends a response to the client.

### Types of FTP Transfers

It can transfer following file types across the internet connections

1. **ASCII file:** This is the default format for transforming a file from one to another. Each character is encoded by NVT ASCII i.e. Network Virtual Terminal ASCII character set. Both the sender and the receiver transform their file from its own representation into NVT ASCII.
2. **EBCDIC:** If sender or receiver connections use the EBCDIC encoding method, then for transforming file FTP uses EBCDIC encoding.
3. **Image File:** For transforming the binary file, the image file is the default mode. The file is transformed over the internet connections in the form of stream bits without encoding.

## TFTP

Trivial File Transfer Protocol, is a simple high-level protocol for transferring data servers use to boot diskless workstations, X-terminals, and routers by using User Data Protocol (UDP).

Although it may sound similar, TFTP works differently than FTP (File Transfer Protocol) and HTTP (HyperText Transfer Protocol). Although TFTP is also based in FTP technology, TFTP is an entirely different protocol. Among the differences is that TFTP's transport protocol uses UDP which is not secure while FTP uses Transmission Control Protocol (TCP) to secure information.

TFTP was primarily designed to read or write files by using a remote server. However, TFTP is a multi-purpose protocol that can be leveraged for an array of different tasks.

### TFTP Configuration Uses

TFTP configuration used for:

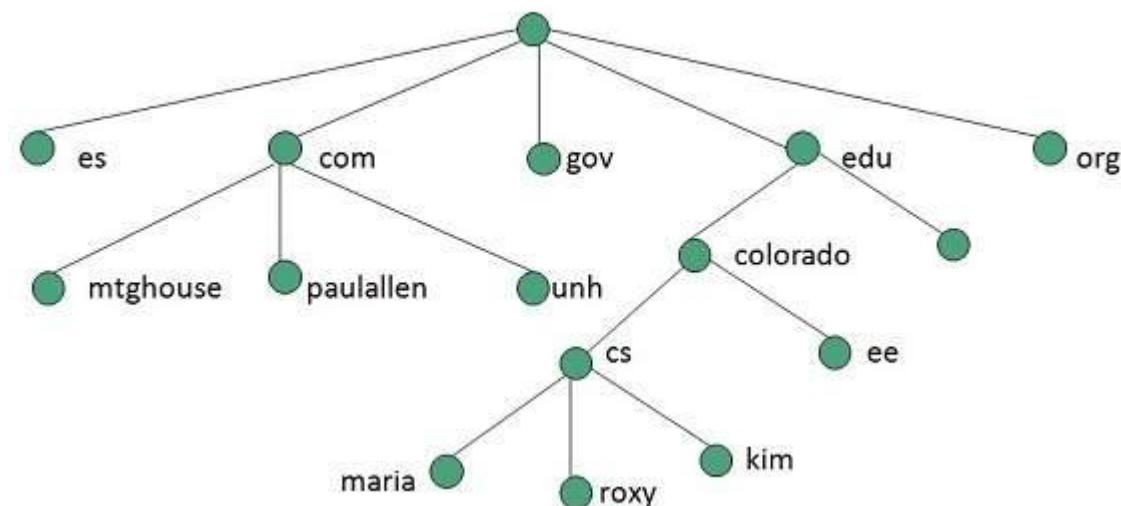
- Transferring files
- Remote-booting without hard drives
- Upgrading codes

- Backing up network configurations
- Backing up router configuration files
- Saving IOS images
- Booting PCs without a disk

After a workstation has been booted from a network card's ROM, the TFTP installation will download a program and then run it from a central server.

## DNS

The domain name space refers a hierarchy in the internet naming structure. This hierarchy has multiple levels (from 0 to 127), with a root at the top. The following diagram shows the domain name space hierarchy:



**Figure 5.5: Domain name space**

DNS is an industry-standard service used to locate computers on an Internet Protocol (IP)-based network. IP networks, such as the Internet and Windows 2000 networks, rely on number-based addresses, such as 207.46.131.137 to ferry information throughout the network. Network users rely on character-based names, such as www.microsoft.com. Therefore, it is necessary to translate character or user-friendly addresses (www.microsoft.com) into the number-based addresses (207.46.131.137) that the network can recognize.

DNS replaced the hosts file's flat name space with a hierarchical name space. With a hierarchical name space, information about host names and IP addresses can be partitioned and distributed, thus, scalability is achieved. For example, in the fictional widgets.products.microsoft.com domain, responsibility for name resolution can be partitioned so that various servers can handle name resolution for different parts of the name space:

- One server can be responsible for resolving the first part (microsoft.com), and can then forward the name-resolution request to the next DNS Server in the hierarchy.
- The next DNS Server can be responsible for resolving the next part of the name space.
- Finally, the request can be forwarded to a third DNS server that is responsible for resolving the last part of the name.

## DISTRIBUTION OF NAME SPACES

A namespace is a context within which the names of all objects must be unambiguously resolvable. For example, the internet is a single DNS name space, within which all network devices with a DNS name can be resolved to a particular address (for example, www.microsoft.com resolves to 207.46.131.13).

## DNS IN THE INTERNET

DNS translates the domain name into IP address automatically. Following are the steps of domain resolution process:

- When user type www.cdgi.com into the browser, it asks the local DNS Server for its IP address.
- When the local DNS does not find the IP address of requested domain name, it forwards the request to the root DNS server and again enquires about IP address of it.
- The root DNS server replies with delegation that user do not know the IP address of www.cdgi.com but know the IP address of DNS Server.
- The local DNS server then asks the com DNS Server the same question.
- The com DNS Server replies the same that it does not know the IP address of www.cdgi.com but knows the address of cdgi.com.
- Then the local DNS asks the cdgi.com DNS server the same question.
- Then cdgi.com DNS server replies with IP address of www.cdgi.com.
- Now, the local DNS sends the IP address of www.cdgi.com to the computer that sends the request.

## SSH Protocol

SSH stands for Secure Shell or Secure Socket Shell. It is a cryptographic network protocol that allows two computers to communicate and share the data over an insecure network such as the internet. It is used to login to a remote server to execute commands and data transfer from one machine to another machine.

A simple example can be understood, such as suppose you want to transfer a package to one of your friends. Without SSH protocol, it can be opened and read by anyone. But if you will send it using SSH protocol, it will be encrypted and secured with the public keys, and only the receiver can open it.

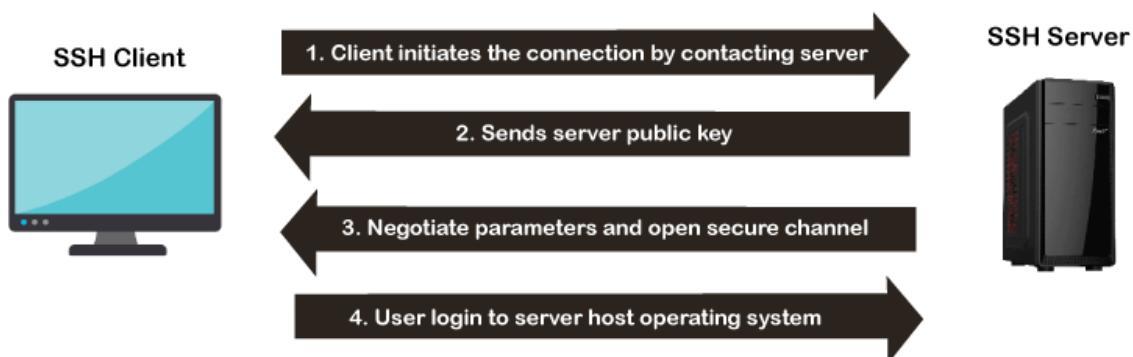
### Usages of SSH protocol

- It provides secure access to users and automated processes.
- It is an easy and secure way to transfer files from one system to another over an insecure network.
- It also issues remote commands to the users.
- It helps the users to manage the network infrastructure and other critical system components

- It is used to log in to shell on a remote system (Host), which replaces **Telnet** and **rlogin** and is used to execute a single command on the host, which replaces **rsh**.
- It combines with **rsync** utility to backup, copy, and mirror files with complete security and efficiency.
- It can be used for forwarding a port.
- By using SSH, we can set up the automatic login to a remote server such as OpenSSH.
- We can securely browse the web through the encrypted proxy connection with the SSH client, supporting the SOCKS protocol.

### How does SSH Works?

The SSH protocol works in a **client-server** model, which means it connects a secure shell client application (End where the session is displayed) with the SSH server (End where session executes).



## **E-MAIL**

### **SMTP**

SMTP stands for Simple Mail Transfer Protocol and it's the industry standard protocol for email sending.

With SMTP users are sending, relaying, or forwarding messages from a mail client (like Microsoft Outlook) to a receiving email server. A sender will use an SMTP server to carry out the process of transmitting an email message.

If user looking to enable email sending within the application, then he must use SMTP over IMAP.

### **MIME(Multipurpose Internet Mail Extension Protocol):**

Multipurpose Internet Mail Extension Protocol is an additional email protocol that allows non-ASCII data to be sent through SMTP. It allows users to send and receive different types of data like audio, images, videos and other application programs on the Internet. It allows to send multiple attachments with single message. It allows to send message of unlimited length.

### **POP**

POP stands for Post Office Protocol. And the number three stands for "version 3," which is the latest version and the most widely used — hence the term "POP3."

POP3 downloads the email from a server to a single computer, and then deletes the email from the server. On the other hand, IMAP stores the message on a server and synchronizes the message across multiple devices.

### **IMAP**

IMAP (Internet Access Message Protocol) is an email protocol that deals with managing and retrieving email messages from the receiving server. Since IMAP deals with message retrieval, user will not be able to use the IMAP protocol to send email. Instead, IMAP will be used for receiving messages.

### **SNMP**

Simple Network Management Protocol (SNMP) is a way for different devices on a network to share information with one another. It allows devices to communicate even if the devices are different hardware and run different software. Without a protocol like SNMP, there would be no way for network management tools to identify devices, monitor network performance, keep track of changes to the network, or determine the status of network devices in real time.

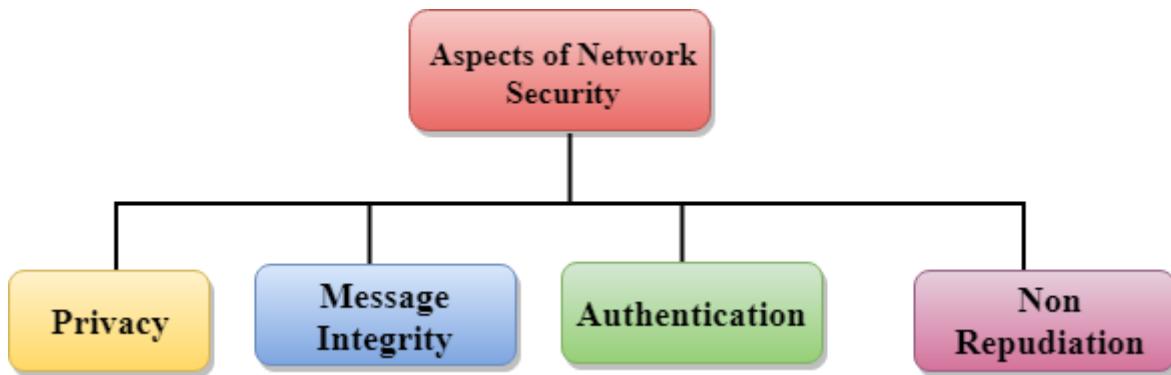
SNMP has a simple architecture based on a client-server model. The servers, called managers, collect and process information about devices on the network.

The clients, called agents, are any type of device or device component connected to the network. They can include not just computers but also network switches, phones, printers. Some devices may have multiple device components. For example, a laptop typically contains a wired as well as a wireless network interface.

All the measures used to safeguard a computer network's integrity and the data on it are collectively referred to as network security. Network security is crucial because it protects sensitive data from online threats and guarantees the network's dependability. Multiple security measures are used in successful network security plans to shield users and organizations from malware and online threats like distributed denial of service.

#### Aspects of Network Security

Following are the desirable properties to achieve secure communication:



#### Cryptography –

**Traditional Ciphers:-** Traditional ciphers encode messages in a way that only those with the secret key can decode it. Throughout history, it has been used for private communication.

These ciphers were used extensively before the advent of modern computer encryption techniques. Although relatively simple, when used properly, it can also be effective and provide interesting insights into the history of encryption.

Here are a few common types –

- Caesar Cipher
- Vigenere Cipher
- Simple Substitution Cipher
- Monoalphabetic and Polyalphabetic Cipher
- Transposition Cipher
- Playfair Cipher

#### Caesar Cipher

Caesar cipher is a type of substitution cipher in which every letter in the given plaintext is shifted with a certain number of places down or up the alphabet. The shift number is called the key. It is created by Julius Caesar. It is believed that this cipher is used to communicate secretly with his generals.

For this type of method both the sender and the receiver should agree on a 'secret shift key' for shifting the alphabet of the given plaintext. The number which can be between 0 and 25 is the key of encryption.

For example - If plaintext is shift of 4:

- 'A' will become 'E'
- 'B' will become 'F'
- 'C' will become 'G'
- and so on...

### **Vigenere Cipher**

Vigenere is also a traditional cipher algorithm which uses a text string (Let us say a word) as a key. Then it is used for shifting a number of shifts on the given plaintext.

For example – let us say the key is 'tutor'. In this each alphabet of the key is changed to its respective numeric value: In our case,

t → 20, u → 21, t → 20, o → 15, and r → 18.

Thus, the key is: 20 21 20 15 18.

### **Simple Substitution Cipher**

Simple substitution cipher is a form of encryption in which the letters in Plaintext is replaced by another character in cipher text according to a fixed order. In other words it is a method of encoding with any letter and characters are mapped to other characters.

For example, in Simple Substitution Cipher –

- 'A' can be replaced by 'X'
- 'B' can be replaced by 'K'
- 'C' can be replaced by 'Q'
- So on... until each character in the alphabet is replaced by another one.

### **Monoalphabetic and Polyalphabetic Cipher:**

A Monoalphabetic cipher is a type of substitution cipher in which fixed characters are always replaced in the ciphertext which means that every character in the plaintext is always replaced by the same corresponding character in the ciphertext. The replacement remains constant throughout the encryption process.

For example, if 'A' is replaced by 'E' in the encryption process, any occurrence of 'A' in plain text will be replaced by 'E' in ciphertext and if 'B' is replaced by 'F'. replace 'any B ' in plaintext with 'F' in ciphertext.

### Transposition Cipher

This is a new type of cipher in which the sequence of letters is rearranged in plain text to create ciphertext. They are not replaced by actual plaintext.

An example is a simple column substitution cipher in which simple increments are written in a specific alphabet width. The ciphertext is then read directly as indicated.

For example, the plain text is "The tajmahal is in agra" and the randomly chosen hidden key is "Four". We format this text directly in a table with a number of columns including key value. The resulting text is shown below.

T	h	e	t
a	j	m	a
h	a	l	i
s	i	n	a
g	r	a	

So now we can get the ciphertext just reading columns from first to last by vertically downward. And the ciphertext will be 'Tahsghjairemlnataia'.

### Playfair Cipher

The Playfair cipher uses a 5x5 grid of letters unless it is double and usually 'J' for encryption. Messages are encrypted by pairing letters and using rules: if on the same letter, turn right; If it is in one column, turn down; If there are rows and columns, make a triangle and replace them with opposite corners. Decryption follows the opposite rule. Playfair encrypts character pairs, making it more secure than single-character ciphers, but still vulnerable to an attack.

### Modern Ciphers - Base64 Encoding & Decoding

Details of Base64 encoding

Base64 refers to a group of related encoding techniques that encode binary data numerically and translate it into a base-64 representation. The word Base64 comes from a specific MIME-content transfer encoding.

Base64 has a specific set of characters –

- 26 Uppercase letters

- 26 Lowercase letters
- 10 Numbers
- Plus sign (+) and slash (/) for new lines

## Applications

Base64 encoding is often used in a variety of applications. It is widely used to encrypt binary data, especially if it needs to be communicated by email or used in other text fields. It is also used in a variety of web and internet protocols, as well as to encode digital signatures and certificates.

## Limitations

Here are the limitations of Base64 encoding –

- Base64 encoding usually maximizes the size of the data by about 33%. This can impact transmission and storage efficiency, mainly for large datasets.
- It is basically not a form of encryption. It simply converts data into a different format. As a result, sensitive information is not kept confidential or secure.
- Base64 adds padding characters ('=') to the end of the encoded data to ensure it aligns properly. This way it can complicate parsing and processing of the encoded data.
- It uses a limited set of characters (A-Z, a-z, 0-9, '+', '/') for encoding. This can lead to issues when the encoded data needs to be transmitted or processed in systems that have restrictions on certain characters.
- The encoding of Base64 does not compress data. It is only meant for representing binary data in a text-based format, not for reducing file size.

## Message Integrity in Cryptography:-

Message integrity in cryptography is the process of checking the message's authenticity. It ensures the message has not been altered or tampered with. Message integrity means that a message has not been tampered with or manipulated. Integrity is important because it ensures that the message has not been modified or tampered with.

Message integrity is commonly used in computing systems for integrity verification and information authentication. They are regarded cryptographically “weak” since they can be solved in polynomial time but are difficult to interpret.

Message integrity enhances traditional hash algorithms with security characteristics, making it more difficult to discover message content or receiver and sender information.

## Steps to Verify the Integrity of a Message

- **Message Authentication Codes:** Suppose two users, a sender, and a receiver, want to connect via messages. In MAC, or Message Authentication Codes, the transmitter and receiver use the same MAC algorithm or key.

- **Certificates:** A certificate is a digital document that validates a public key. The certificate provides information about the key, the owner's identity, and the organization's digital signature, which has verified the certificate's contents.
- **Nonrepudiation:** Nonrepudiation is the property of agreeing to adhere to an obligation. More specifically, it is the inability to refute responsibility.
- **Message Authentication Codes:** In the case of MAC, there is no public key. There is just one private key, which is known only to the sender and receiver. As a result, there is no interference from external parties. Even if a third-party user had access to the secret key, he could not guarantee that either the sender or the recipient signed the message because both can encrypt or decrypt it.

### Message authentication

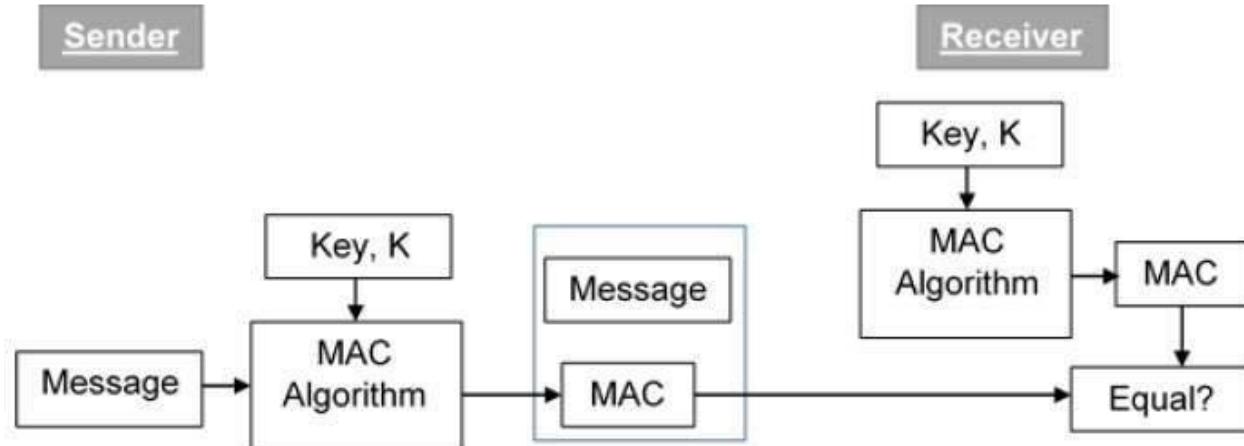
Message authentication can be provided using the cryptographic techniques that use secret keys as done in case of encryption.

Message Authentication Code (MAC):

MAC algorithm is a symmetric key cryptographic technique to provide message authentication. For establishing MAC process, the sender and receiver share a symmetric key K.

Essentially, a MAC is an encrypted checksum generated on the underlying message that is sent along with a message to ensure message authentication.

The process of using MAC for authentication is depicted in the following illustration –



Let us now try to understand the entire process in detail –

- The sender uses some publicly known MAC algorithm, inputs the message and the secret key K and produces a MAC value.
- Similar to hash, MAC function also compresses an arbitrary long input into a fixed length output. The major difference between hash and MAC is that MAC uses secret key during the compression.
- The sender forwards the message along with the MAC. Here, we assume that the message is sent in the clear, as we are concerned of providing message origin authentication, not confidentiality. If confidentiality is required then the message needs encryption.
- On receipt of the message and the MAC, the receiver feeds the received message and the shared secret key K into the MAC algorithm and re-computes the MAC value.
- The receiver now checks equality of freshly computed MAC with the MAC received from the sender. If they match, then the receiver accepts the message and assures himself that the message has been sent by the intended sender.
- If the computed MAC does not match the MAC sent by the sender, the receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified. As a bottom-line, a receiver safely assumes that the message is not the genuine.

**Chameli Devi Group of Institutions, Indore**

**Department of Information Technology**

**Subject: Computer Networks (IT502)**

## Unit V

Application Layer: WWW and HTTP, FTP, SSH, Email (SMTP, MIME, IMAP), DNS, Network Management (SNMP).

Network Security: Introduction to security, Traditional Ciphers, Modern Ciphers, Message Integrity and Authentication.

## **UNIT V APPLICATION LAYER**

### **World Wide Web (WWW)**

What is World Wide Web (WWW, W3)?

The World Wide Web -- also known as the web, WWW or W3 -- refers to all the public websites or pages that users can access on their local computers and other devices through the [internet](#). These pages and documents are interconnected by means of hyperlinks that users click on for information. This information can be in different formats, including text, images, audio and video.

The term *World Wide Web* isn't synonymous with the internet. Rather, the World Wide Web is part of the internet.

## **How does the World Wide Web work?**

Paving the way for an internet revolution that has transformed the world in only three decades, the World Wide Web consists of multiple components that enable users to access various resources, documents and web pages on the internet. Thus, the WWW is like a vast electronic book whose pages are stored or hosted on different servers worldwide.

These pages are the primary component or building blocks of the WWW and are linked through hyperlinks, which provide access from one specific spot in a hypertext or hypermedia document to another spot within that document or a different one. Hyperlinks are another defining concept of the WWW and provide its identity as a collection of interconnected documents.

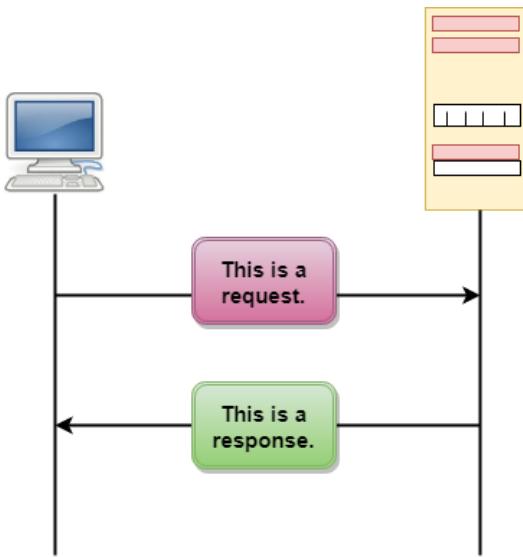
## **HTTP**

- HTTP stands for HyperText Transfer Protocol.
- It is a protocol used to access the data on the World Wide Web (www).
- The HTTP protocol can be used to transfer the data in the form of plain text, hypertext, audio, video, and so on.
- This protocol is known as HyperText Transfer Protocol because of its efficiency that allows us to use in a hypertext environment where there are rapid jumps from one document to another document.
- HTTP is similar to the FTP as it also transfers the files from one host to another host. But, HTTP is simpler than FTP as HTTP uses only one connection, i.e., no control connection to transfer the files.

## **Features of HTTP:**

- Connectionless protocol: HTTP is a connectionless protocol. HTTP client initiates a request and waits for a response from the server. When the server receives the request, the server processes the request and sends back the response to the HTTP client after which the client disconnects the connection. The connection between client and server exist only during the current request and response time only.
- Media independent: HTTP protocol is a media independent as data can be sent as long as both the client and server know how to handle the data content. It is required for both the client and server to specify the content type in MIME-type header.
- Stateless: HTTP is a stateless protocol as both the client and server know each other only during the current request. Due to this nature of the protocol, both the client and server do not retain the information between various requests of the web pages.

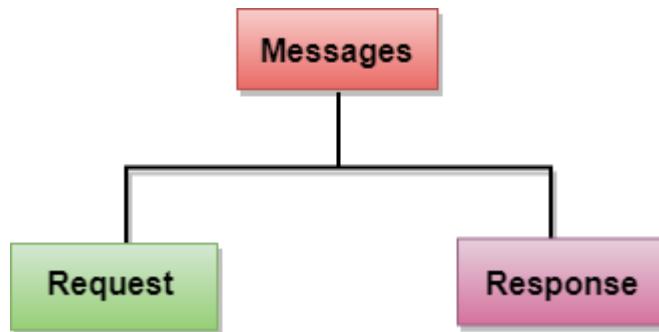
## **HTTP Transactions**



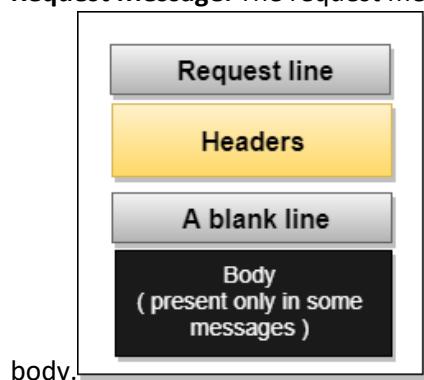
The above figure shows the HTTP transaction between client and server. The client initiates a transaction by sending a request message to the server. The server replies to the request message by sending a response message.

## Messages

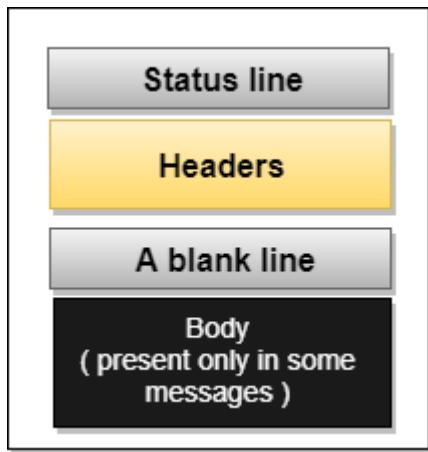
HTTP messages are of two types: request and response. Both the message types follow the same message format.



**Request Message:** The request message is sent by the client that consists of a request line, headers, and sometimes a



**Response Message:** The response message is sent by the server to the client that consists of a status line, headers, and sometimes a body.



## FTP

FTP stands for File Transfer Protocol, and it is basically the protocol, or procedure, for transferring files between computers. Files are not actually moved over to the source system, rather, they are copied from one computer to another. This exchange of files happens over Internet channels, formally referred to as a TCP/IP network. TCP/IP (Transmission Control Protocol/Internet Protocol) stands for the standard protocol of communication over the internet. FTP is a standard protocol, and any computer with the ability to use FTP can make use of the protocol.

## CONNECTION

For FTP to work there needs to be a client and a server. The client connects to the service. The client and server communicate back and forth, the server must validate that the client has access to parts of the server it is trying to access and then the client copies the files to the directories.

### Connection Modes: Active vs. Passive.

FTP may work in either active or passive mode, which sets how the data connection is setup. Each type involves the client creating a connection to the FTP server on a certain port (usually port 21). A port is a number assigned to servers in a TCP/IP network. They simply indicate the purpose of the data being transferred (e.g., web page, voice call). Servers will monitor a given port to know when data starts coming in. FTP servers usually watch port 21.

#### Active Mode

In active mode, the client connects, indicating that it will be monitoring port 21 for incoming data from the server. Behind the scenes, the client sends a command PORT to tell the server which port it's on. However, most clients are now behind firewalls, either personal or corporate, and the firewall doesn't let in incoming connections. For that purpose, passive mode will work better.

Active mode requires most of the setup to be completed on the client side. If using a firewall, it must be configured to allow ports for incoming connections.

#### Passive Mode

In passive mode, the client sends a PASV command to the server, the server then sends back an IP (Internet) address and port number. The client can use this connection to the server and data transfer/copying can occur.

For this connection mode, most of the work is on the server side. The server needs to setup the system so that it can accept incoming data, not only from port 21 but from a range of possible ports for incoming connections.

FTP uses TCP services. It needs two TCP connections. One is Control connection and another is Data connection. For control connection, it uses well-known port 21 and for data connection, it uses well-known port 20

### Control Connection

A server site control connection uses a well-known port 21. There are two steps to establish a control connection –

- Server issues a passive open on the well-known port 21 and waits for the client
- After severing issues passive open, the client issue active open using an ephemeral port.

This control connection remains open throughout the process. Since the user and the server uses the interactive connection for communication, their service used by internet protocol minimizes the delay. For communication, user types the command and in return, servers give responses without any delay.

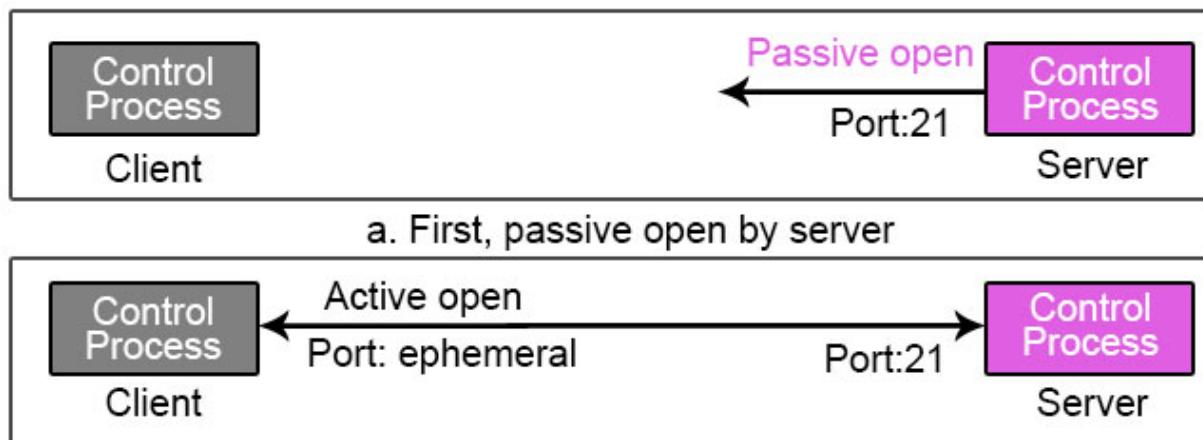
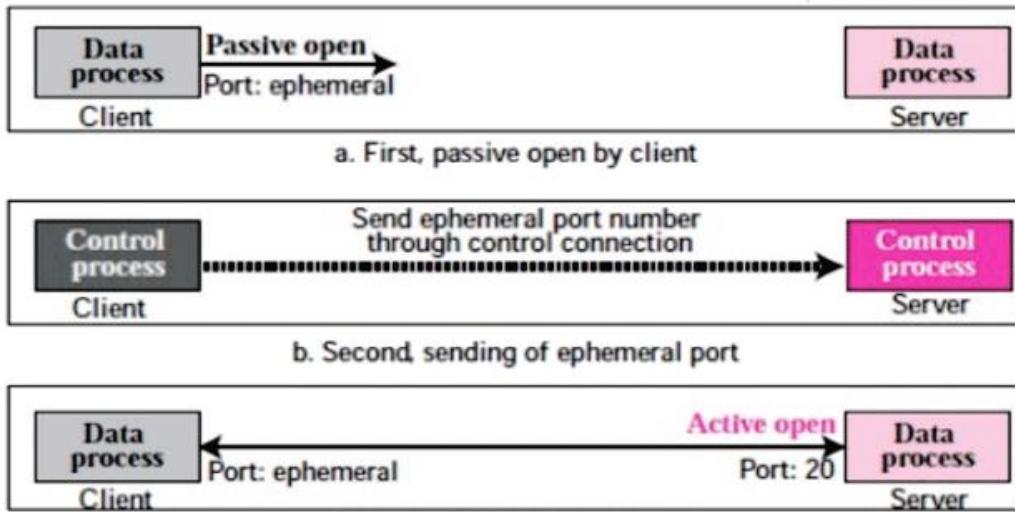


Figure 5.6: FTP Connection

### Control Connection

Data Connections: At the server site, the data connection uses well-known port 20. There are three steps to establish a data connection –

- Using ephemeral port client issues a passive open. This step must be done by the client not the server because the client wants to transform the file.
- Using the PORT command client sends this port number to the server.
- When the server receives this port number from the client, it issues active open using well-known port 20.



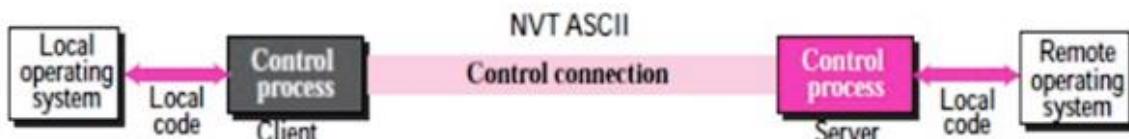
**Figure 5.7: FTP Control Connection**

## COMMUNICATION

Both the client and the server which runs on two different systems must be communicated for transforming data. For communication, it uses two approaches i.e. communication over control connection and communication over a data connection.

### Communication over the control connection

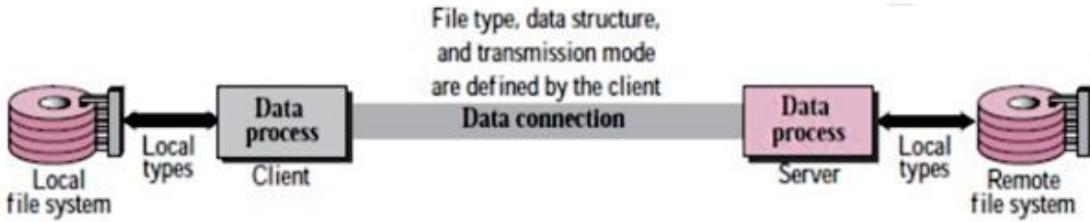
To communicate over control connection FTP uses TELNET or SMTP. Communication over control connection is done by commands and responses. The first command is sent over the connection and in return, a response is sent by another system. We can send a command or response at a time. There is only one-way communication.



**Figure 5.8: Communication over the control connection**

### Communication over the data connection

For transforming file over the data connection, the client must define the type of file which needs to be transformed, transmission mode, and the data structure. It solves the heterogeneity problem by defining these three attributes.



**Figure 5.9: Communication over the data connection**

## COMMAND PROCESSING

To establish communication between the client system and the server system FTP uses a control connection. During this process, the client sends commands to the server and in return, the server sends a response to the client.

### Types of FTP Transfers

It can transfer following file types across the internet connections

1. **ASCII file:** This is the default format for transforming a file from one to another. Each character is encoded by NVT ASCII i.e. Network Virtual Terminal ASCII character set. Both the sender and the receiver transform their file from its own representation into NVT ASCII.
2. **EBCDIC:** If sender or receiver connections use the EBCDIC encoding method, then for transforming file FTP uses EBCDIC encoding.
3. **Image File:** For transforming the binary file, the image file is the default mode. The file is transformed over the internet connections in the form of stream bits without encoding.

## TFTP

Trivial File Transfer Protocol, is a simple high-level protocol for transferring data servers use to boot diskless workstations, X-terminals, and routers by using User Data Protocol (UDP).

Although it may sound similar, TFTP works differently than FTP (File Transfer Protocol) and HTTP (HyperText Transfer Protocol). Although TFTP is also based in FTP technology, TFTP is an entirely different protocol. Among the differences is that TFTP's transport protocol uses UDP which is not secure while FTP uses Transmission Control Protocol (TCP) to secure information.

TFTP was primarily designed to read or write files by using a remote server. However, TFTP is a multi-purpose protocol that can be leveraged for an array of different tasks.

### TFTP Configuration Uses

TFTP configuration used for:

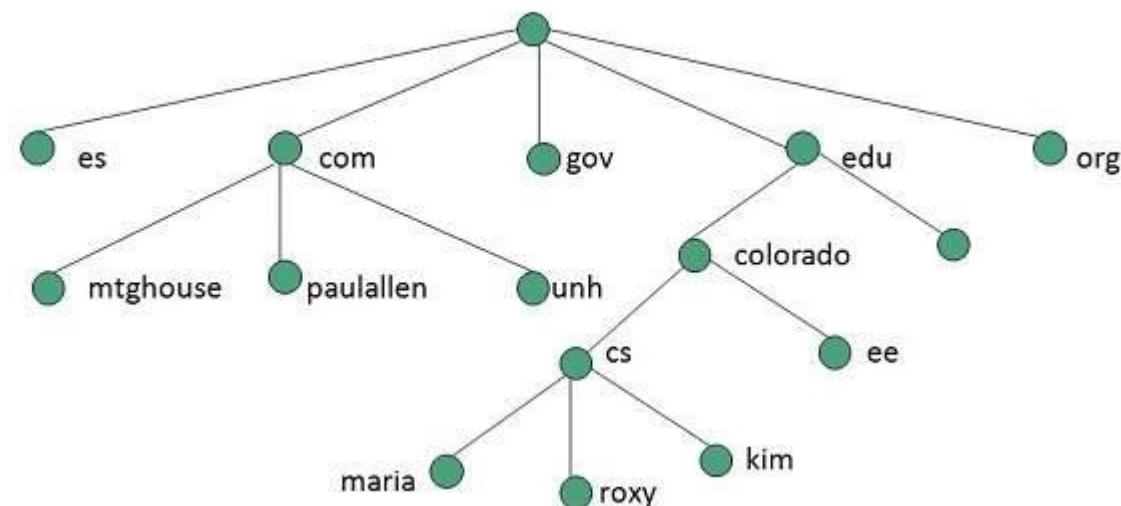
- Transferring files
- Remote-booting without hard drives
- Upgrading codes

- Backing up network configurations
- Backing up router configuration files
- Saving IOS images
- Booting PCs without a disk

After a workstation has been booted from a network card's ROM, the TFTP installation will download a program and then run it from a central server.

## DNS

The domain name space refers a hierarchy in the internet naming structure. This hierarchy has multiple levels (from 0 to 127), with a root at the top. The following diagram shows the domain name space hierarchy:



**Figure 5.5: Domain name space**

DNS is an industry-standard service used to locate computers on an Internet Protocol (IP)-based network. IP networks, such as the Internet and Windows 2000 networks, rely on number-based addresses, such as 207.46.131.137 to ferry information throughout the network. Network users rely on character-based names, such as www.microsoft.com. Therefore, it is necessary to translate character or user-friendly addresses (www.microsoft.com) into the number-based addresses (207.46.131.137) that the network can recognize.

DNS replaced the hosts file's flat name space with a hierarchical name space. With a hierarchical name space, information about host names and IP addresses can be partitioned and distributed, thus, scalability is achieved. For example, in the fictional widgets.products.microsoft.com domain, responsibility for name resolution can be partitioned so that various servers can handle name resolution for different parts of the name space:

- One server can be responsible for resolving the first part (microsoft.com), and can then forward the name-resolution request to the next DNS Server in the hierarchy.
- The next DNS Server can be responsible for resolving the next part of the name space.
- Finally, the request can be forwarded to a third DNS server that is responsible for resolving the last part of the name.

## DISTRIBUTION OF NAME SPACES

A namespace is a context within which the names of all objects must be unambiguously resolvable. For example, the internet is a single DNS name space, within which all network devices with a DNS name can be resolved to a particular address (for example, www.microsoft.com resolves to 207.46.131.13).

## DNS IN THE INTERNET

DNS translates the domain name into IP address automatically. Following are the steps of domain resolution process:

- When user type www.cdgi.com into the browser, it asks the local DNS Server for its IP address.
- When the local DNS does not find the IP address of requested domain name, it forwards the request to the root DNS server and again enquires about IP address of it.
- The root DNS server replies with delegation that user do not know the IP address of www.cdgi.com but know the IP address of DNS Server.
- The local DNS server then asks the com DNS Server the same question.
- The com DNS Server replies the same that it does not know the IP address of www.cdgi.com but knows the address of cdgi.com.
- Then the local DNS asks the cdgi.com DNS server the same question.
- Then cdgi.com DNS server replies with IP address of www.cdgi.com.
- Now, the local DNS sends the IP address of www.cdgi.com to the computer that sends the request.

## SSH Protocol

SSH stands for Secure Shell or Secure Socket Shell. It is a cryptographic network protocol that allows two computers to communicate and share the data over an insecure network such as the internet. It is used to login to a remote server to execute commands and data transfer from one machine to another machine.

A simple example can be understood, such as suppose you want to transfer a package to one of your friends. Without SSH protocol, it can be opened and read by anyone. But if you will send it using SSH protocol, it will be encrypted and secured with the public keys, and only the receiver can open it.

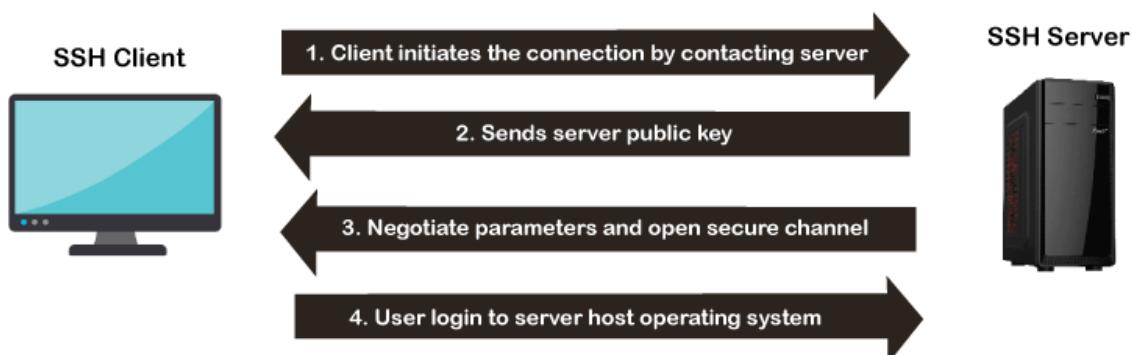
### Usages of SSH protocol

- It provides secure access to users and automated processes.
- It is an easy and secure way to transfer files from one system to another over an insecure network.
- It also issues remote commands to the users.
- It helps the users to manage the network infrastructure and other critical system components

- It is used to log in to shell on a remote system (Host), which replaces **Telnet** and **rlogin** and is used to execute a single command on the host, which replaces **rsh**.
- It combines with **rsync** utility to backup, copy, and mirror files with complete security and efficiency.
- It can be used for forwarding a port.
- By using SSH, we can set up the automatic login to a remote server such as OpenSSH.
- We can securely browse the web through the encrypted proxy connection with the SSH client, supporting the SOCKS protocol.

### How does SSH Works?

The SSH protocol works in a **client-server** model, which means it connects a secure shell client application (End where the session is displayed) with the SSH server (End where session executes).



## **E-MAIL**

### **SMTP**

SMTP stands for Simple Mail Transfer Protocol and it's the industry standard protocol for email sending.

With SMTP users are sending, relaying, or forwarding messages from a mail client (like Microsoft Outlook) to a receiving email server. A sender will use an SMTP server to carry out the process of transmitting an email message.

If user looking to enable email sending within the application, then he must use SMTP over IMAP.

### **MIME(Multipurpose Internet Mail Extension Protocol):**

Multipurpose Internet Mail Extension Protocol is an additional email protocol that allows non-ASCII data to be sent through SMTP. It allows users to send and receive different types of data like audio, images, videos and other application programs on the Internet. It allows to send multiple attachments with single message. It allows to send message of unlimited length.

### **POP**

POP stands for Post Office Protocol. And the number three stands for "version 3," which is the latest version and the most widely used — hence the term "POP3."

POP3 downloads the email from a server to a single computer, and then deletes the email from the server. On the other hand, IMAP stores the message on a server and synchronizes the message across multiple devices.

### **IMAP**

IMAP (Internet Access Message Protocol) is an email protocol that deals with managing and retrieving email messages from the receiving server. Since IMAP deals with message retrieval, user will not be able to use the IMAP protocol to send email. Instead, IMAP will be used for receiving messages.

### **SNMP**

Simple Network Management Protocol (SNMP) is a way for different devices on a network to share information with one another. It allows devices to communicate even if the devices are different hardware and run different software. Without a protocol like SNMP, there would be no way for network management tools to identify devices, monitor network performance, keep track of changes to the network, or determine the status of network devices in real time.

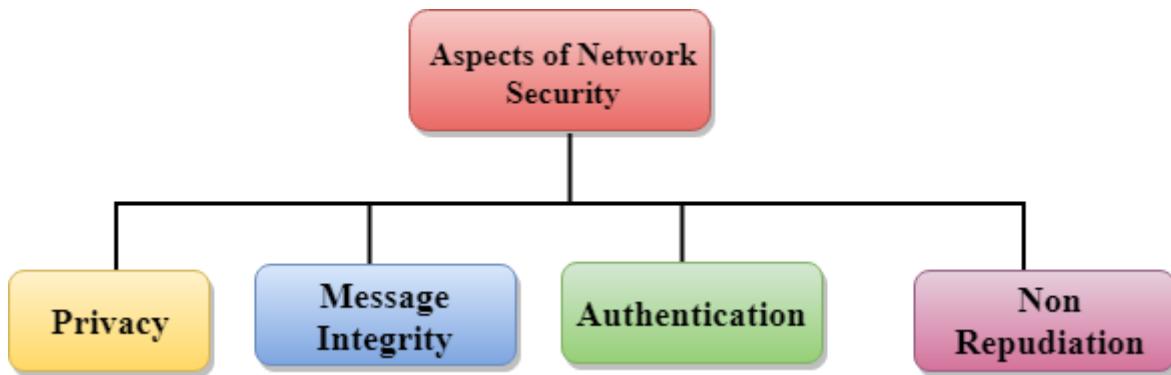
SNMP has a simple architecture based on a client-server model. The servers, called managers, collect and process information about devices on the network.

The clients, called agents, are any type of device or device component connected to the network. They can include not just computers but also network switches, phones, printers. Some devices may have multiple device components. For example, a laptop typically contains a wired as well as a wireless network interface.

All the measures used to safeguard a computer network's integrity and the data on it are collectively referred to as network security. Network security is crucial because it protects sensitive data from online threats and guarantees the network's dependability. Multiple security measures are used in successful network security plans to shield users and organizations from malware and online threats like distributed denial of service.

#### Aspects of Network Security

Following are the desirable properties to achieve secure communication:



#### Cryptography –

**Traditional Ciphers:-** Traditional ciphers encode messages in a way that only those with the secret key can decode it. Throughout history, it has been used for private communication.

These ciphers were used extensively before the advent of modern computer encryption techniques. Although relatively simple, when used properly, it can also be effective and provide interesting insights into the history of encryption.

#### Here are a few common types –

- Caesar Cipher
- Vigenere Cipher
- Simple Substitution Cipher
- Monoalphabetic and Polyalphabetic Cipher
- Transposition Cipher
- Playfair Cipher

#### Caesar Cipher

Caesar cipher is a type of substitution cipher in which every letter in the given plaintext is shifted with a certain number of places down or up the alphabet. The shift number is called the key. It is created by Julius Caesar. It is believed that this cipher is used to communicate secretly with his generals.

For this type of method both the sender and the receiver should agree on a 'secret shift key' for shifting the alphabet of the given plaintext. The number which can be between 0 and 25 is the key of encryption.

For example - If plaintext is shift of 4:

- 'A' will become 'E'
- 'B' will become 'F'
- 'C' will become 'G'
- and so on...

### **Vigenere Cipher**

Vigenere is also a traditional cipher algorithm which uses a text string (Let us say a word) as a key. Then it is used for shifting a number of shifts on the given plaintext.

For example – let us say the key is 'tutor'. In this each alphabet of the key is changed to its respective numeric value: In our case,

$t \rightarrow 20, u \rightarrow 21, t \rightarrow 20, o \rightarrow 15, \text{ and } r \rightarrow 18.$

Thus, the key is: 20 21 20 15 18.

### **Simple Substitution Cipher**

Simple substitution cipher is a form of encryption in which the letters in Plaintext is replaced by another character in cipher text according to a fixed order. In other words it is a method of encoding with any letter and characters are mapped to other characters.

For example, in Simple Substitution Cipher –

- 'A' can be replaced by 'X'
- 'B' can be replaced by 'K'
- 'C' can be replaced by 'Q'
- So on... until each character in the alphabet is replaced by another one.

### **Monoalphabetic and Polyalphabetic Cipher:**

A Monoalphabetic cipher is a type of substitution cipher in which fixed characters are always replaced in the ciphertext which means that every character in the plaintext is always replaced by the same corresponding character in the ciphertext. The replacement remains constant throughout the encryption process.

For example, if 'A' is replaced by 'E' in the encryption process, any occurrence of 'A' in plain text will be replaced by 'E' in ciphertext and if 'B' is replaced by 'F'. replace 'any B ' in plaintext with 'F' in ciphertext.

### Transposition Cipher

This is a new type of cipher in which the sequence of letters is rearranged in plain text to create ciphertext. They are not replaced by actual plaintext.

An example is a simple column substitution cipher in which simple increments are written in a specific alphabet width. The ciphertext is then read directly as indicated.

For example, the plain text is "The tajmahal is in agra" and the randomly chosen hidden key is "Four". We format this text directly in a table with a number of columns including key value. The resulting text is shown below.

T	h	e	t
a	j	m	a
h	a	l	i
s	i	n	a
g	r	a	

So now we can get the ciphertext just reading columns from first to last by vertically downward. And the ciphertext will be 'Tahsghjairemlnataia'.

### Playfair Cipher

The Playfair cipher uses a 5x5 grid of letters unless it is double and usually 'J' for encryption. Messages are encrypted by pairing letters and using rules: if on the same letter, turn right; If it is in one column, turn down; If there are rows and columns, make a triangle and replace them with opposite corners. Decryption follows the opposite rule. Playfair encrypts character pairs, making it more secure than single-character ciphers, but still vulnerable to an attack.

### Modern Ciphers - Base64 Encoding & Decoding

Details of Base64 encoding

Base64 refers to a group of related encoding techniques that encode binary data numerically and translate it into a base-64 representation. The word Base64 comes from a specific MIME-content transfer encoding.

Base64 has a specific set of characters –

- 26 Uppercase letters

- 26 Lowercase letters
- 10 Numbers
- Plus sign (+) and slash (/) for new lines

## Applications

Base64 encoding is often used in a variety of applications. It is widely used to encrypt binary data, especially if it needs to be communicated by email or used in other text fields. It is also used in a variety of web and internet protocols, as well as to encode digital signatures and certificates.

## Limitations

Here are the limitations of Base64 encoding –

- Base64 encoding usually maximizes the size of the data by about 33%. This can impact transmission and storage efficiency, mainly for large datasets.
- It is basically not a form of encryption. It simply converts data into a different format. As a result, sensitive information is not kept confidential or secure.
- Base64 adds padding characters ('=') to the end of the encoded data to ensure it aligns properly. This way it can complicate parsing and processing of the encoded data.
- It uses a limited set of characters (A-Z, a-z, 0-9, '+', '/') for encoding. This can lead to issues when the encoded data needs to be transmitted or processed in systems that have restrictions on certain characters.
- The encoding of Base64 does not compress data. It is only meant for representing binary data in a text-based format, not for reducing file size.

## Message Integrity in Cryptography:-

Message integrity in cryptography is the process of checking the message's authenticity. It ensures the message has not been altered or tampered with. Message integrity means that a message has not been tampered with or manipulated. Integrity is important because it ensures that the message has not been modified or tampered with.

Message integrity is commonly used in computing systems for integrity verification and information authentication. They are regarded cryptographically “weak” since they can be solved in polynomial time but are difficult to interpret.

Message integrity enhances traditional hash algorithms with security characteristics, making it more difficult to discover message content or receiver and sender information.

## Steps to Verify the Integrity of a Message

- **Message Authentication Codes:** Suppose two users, a sender, and a receiver, want to connect via messages. In MAC, or Message Authentication Codes, the transmitter and receiver use the same MAC algorithm or key.

- **Certificates:** A certificate is a digital document that validates a public key. The certificate provides information about the key, the owner's identity, and the organization's digital signature, which has verified the certificate's contents.
- **Nonrepudiation:** Nonrepudiation is the property of agreeing to adhere to an obligation. More specifically, it is the inability to refute responsibility.
- **Message Authentication Codes:** In the case of MAC, there is no public key. There is just one private key, which is known only to the sender and receiver. As a result, there is no interference from external parties. Even if a third-party user had access to the secret key, he could not guarantee that either the sender or the recipient signed the message because both can encrypt or decrypt it.

### Message authentication

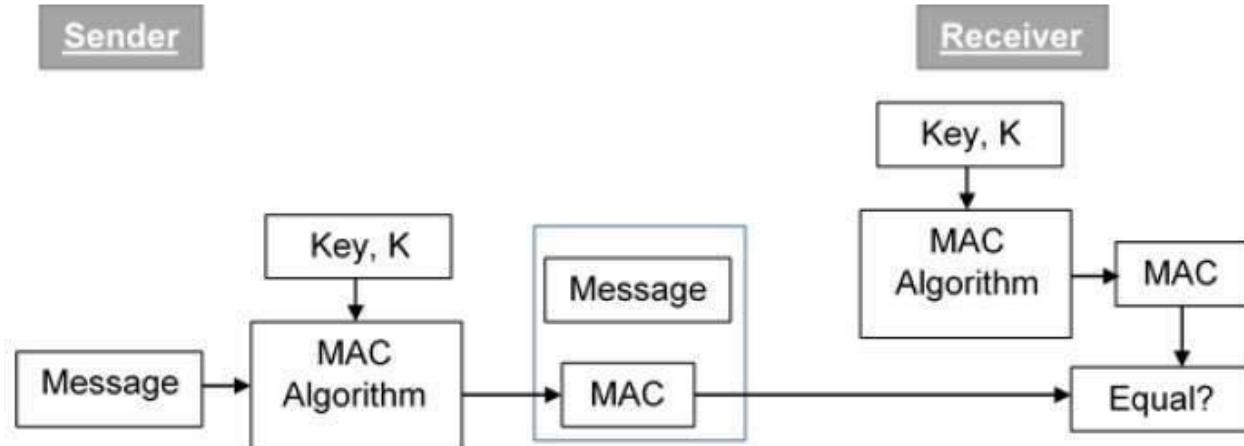
Message authentication can be provided using the cryptographic techniques that use secret keys as done in case of encryption.

Message Authentication Code (MAC):

MAC algorithm is a symmetric key cryptographic technique to provide message authentication. For establishing MAC process, the sender and receiver share a symmetric key K.

Essentially, a MAC is an encrypted checksum generated on the underlying message that is sent along with a message to ensure message authentication.

The process of using MAC for authentication is depicted in the following illustration –



Let us now try to understand the entire process in detail –

- The sender uses some publicly known MAC algorithm, inputs the message and the secret key K and produces a MAC value.
- Similar to hash, MAC function also compresses an arbitrary long input into a fixed length output. The major difference between hash and MAC is that MAC uses secret key during the compression.
- The sender forwards the message along with the MAC. Here, we assume that the message is sent in the clear, as we are concerned of providing message origin authentication, not confidentiality. If confidentiality is required then the message needs encryption.
- On receipt of the message and the MAC, the receiver feeds the received message and the shared secret key K into the MAC algorithm and re-computes the MAC value.
- The receiver now checks equality of freshly computed MAC with the MAC received from the sender. If they match, then the receiver accepts the message and assures himself that the message has been sent by the intended sender.
- If the computed MAC does not match the MAC sent by the sender, the receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified. As a bottom-line, a receiver safely assumes that the message is not the genuine.

**Chameli Devi Group of Institutions, Indore**

**Department of Information Technology**

**Subject: Computer Networks (IT502)**

## Unit V

Application Layer: WWW and HTTP, FTP, SSH, Email (SMTP, MIME, IMAP), DNS, Network Management (SNMP).

Network Security: Introduction to security, Traditional Ciphers, Modern Ciphers, Message Integrity and Authentication.

## UNIT V APPLICATION LAYER

### World Wide Web (WWW)

What is World Wide Web (WWW, W3)?

The World Wide Web -- also known as the web, WWW or W3 -- refers to all the public websites or pages that users can access on their local computers and other devices through the [internet](#). These pages and documents are interconnected by means of hyperlinks that users click on for information. This information can be in different formats, including text, images, audio and video.

The term *World Wide Web* isn't synonymous with the internet. Rather, the World Wide Web is part of the internet.

### How does the World Wide Web work?

Paving the way for an internet revolution that has transformed the world in only three decades, the World Wide Web consists of multiple components that enable users to access various resources, documents and web pages on the internet. Thus, the WWW is like a vast electronic book whose pages are stored or hosted on different servers worldwide.

These pages are the primary component or building blocks of the WWW and are linked through hyperlinks, which provide access from one specific spot in a hypertext or hypermedia document to another spot within that document or a different one. Hyperlinks are another defining concept of the WWW and provide its identity as a collection of interconnected documents.

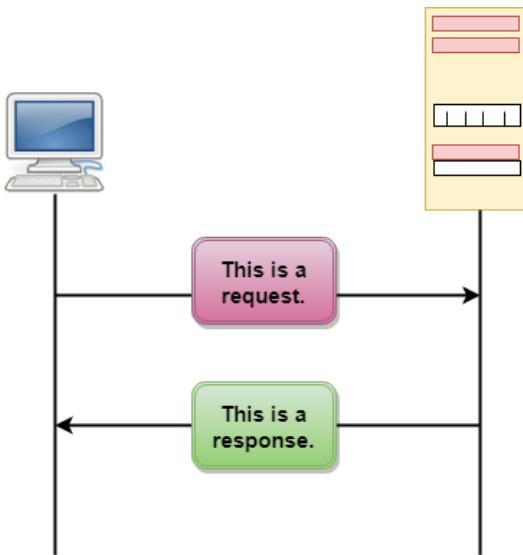
## **HTTP**

- HTTP stands for HyperText Transfer Protocol.
- It is a protocol used to access the data on the World Wide Web (www).
- The HTTP protocol can be used to transfer the data in the form of plain text, hypertext, audio, video, and so on.
- This protocol is known as HyperText Transfer Protocol because of its efficiency that allows us to use in a hypertext environment where there are rapid jumps from one document to another document.
- HTTP is similar to the FTP as it also transfers the files from one host to another host. But, HTTP is simpler than FTP as HTTP uses only one connection, i.e., no control connection to transfer the files.

## **Features of HTTP:**

- Connectionless protocol: HTTP is a connectionless protocol. HTTP client initiates a request and waits for a response from the server. When the server receives the request, the server processes the request and sends back the response to the HTTP client after which the client disconnects the connection. The connection between client and server exist only during the current request and response time only.
- Media independent: HTTP protocol is a media independent as data can be sent as long as both the client and server know how to handle the data content. It is required for both the client and server to specify the content type in MIME-type header.
- Stateless: HTTP is a stateless protocol as both the client and server know each other only during the current request. Due to this nature of the protocol, both the client and server do not retain the information between various requests of the web pages.

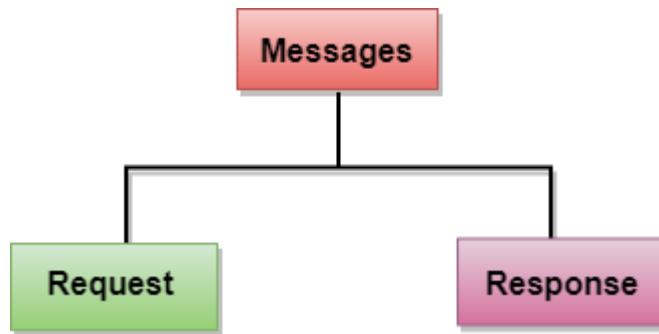
## **HTTP Transactions**



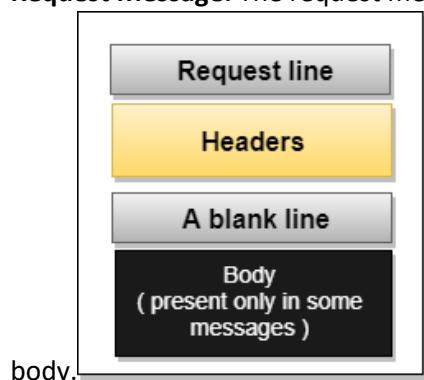
The above figure shows the HTTP transaction between client and server. The client initiates a transaction by sending a request message to the server. The server replies to the request message by sending a response message.

## Messages

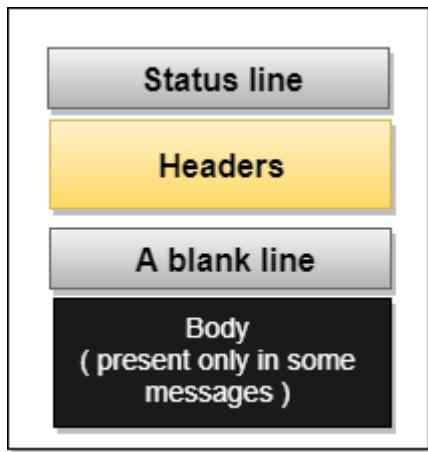
HTTP messages are of two types: request and response. Both the message types follow the same message format.



**Request Message:** The request message is sent by the client that consists of a request line, headers, and sometimes a



**Response Message:** The response message is sent by the server to the client that consists of a status line, headers, and sometimes a body.



## FTP

FTP stands for File Transfer Protocol, and it is basically the protocol, or procedure, for transferring files between computers. Files are not actually moved over to the source system, rather, they are copied from one computer to another. This exchange of files happens over Internet channels, formally referred to as a TCP/IP network. TCP/IP (Transmission Control Protocol/Internet Protocol) stands for the standard protocol of communication over the internet. FTP is a standard protocol, and any computer with the ability to use FTP can make use of the protocol.

## CONNECTION

For FTP to work there needs to be a client and a server. The client connects to the service. The client and server communicate back and forth, the server must validate that the client has access to parts of the server it is trying to access and then the client copies the files to the directories.

### Connection Modes: Active vs. Passive.

FTP may work in either active or passive mode, which sets how the data connection is setup. Each type involves the client creating a connection to the FTP server on a certain port (usually port 21). A port is a number assigned to servers in a TCP/IP network. They simply indicate the purpose of the data being transferred (e.g., web page, voice call). Servers will monitor a given port to know when data starts coming in. FTP servers usually watch port 21.

#### Active Mode

In active mode, the client connects, indicating that it will be monitoring port 21 for incoming data from the server. Behind the scenes, the client sends a command PORT to tell the server which port it's on. However, most clients are now behind firewalls, either personal or corporate, and the firewall doesn't let in incoming connections. For that purpose, passive mode will work better.

Active mode requires most of the setup to be completed on the client side. If using a firewall, it must be configured to allow ports for incoming connections.

#### Passive Mode

In passive mode, the client sends a PASV command to the server, the server then sends back an IP (Internet) address and port number. The client can use this connection to the server and data transfer/copying can occur.

For this connection mode, most of the work is on the server side. The server needs to setup the system so that it can accept incoming data, not only from port 21 but from a range of possible ports for incoming connections.

FTP uses TCP services. It needs two TCP connections. One is Control connection and another is Data connection. For control connection, it uses well-known port 21 and for data connection, it uses well-known port 20

### Control Connection

A server site control connection uses a well-known port 21. There are two steps to establish a control connection –

- Server issues a passive open on the well-known port 21 and waits for the client
- After severing issues passive open, the client issue active open using an ephemeral port.

This control connection remains open throughout the process. Since the user and the server uses the interactive connection for communication, their service used by internet protocol minimizes the delay. For communication, user types the command and in return, servers give responses without any delay.

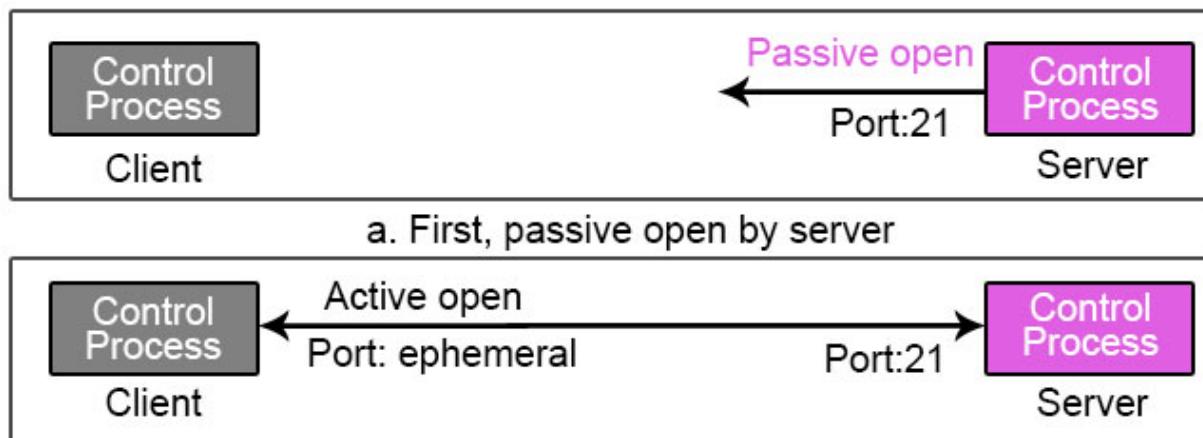
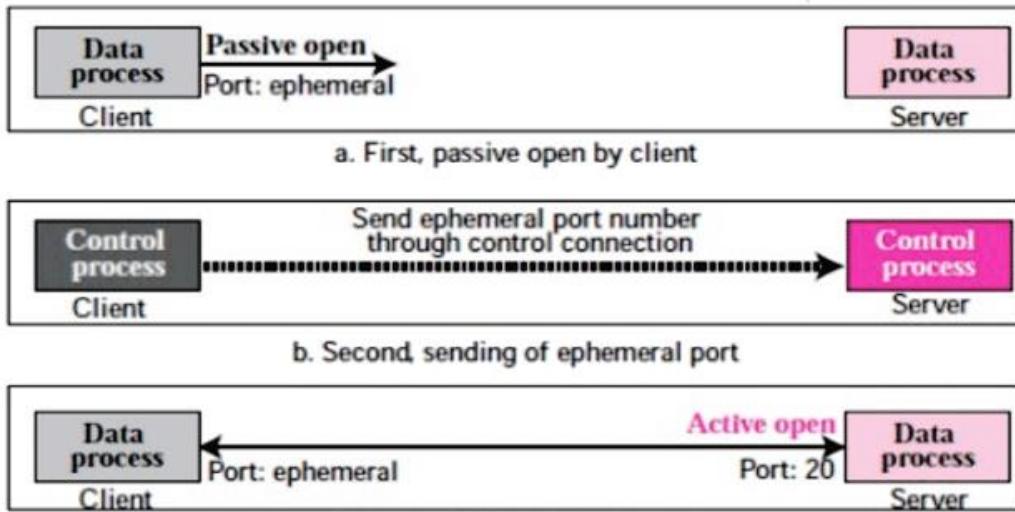


Figure 5.6: FTP Connection

### Control Connection

Data Connections: At the server site, the data connection uses well-known port 20. There are three steps to establish a data connection –

- Using ephemeral port client issues a passive open. This step must be done by the client not the server because the client wants to transform the file.
- Using the PORT command client sends this port number to the server.
- When the server receives this port number from the client, it issues active open using well-known port 20.



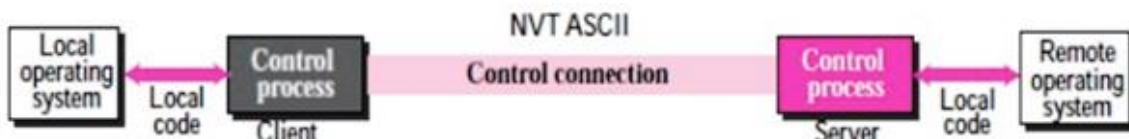
**Figure 5.7: FTP Control Connection**

## COMMUNICATION

Both the client and the server which runs on two different systems must be communicated for transforming data. For communication, it uses two approaches i.e. communication over control connection and communication over a data connection.

### Communication over the control connection

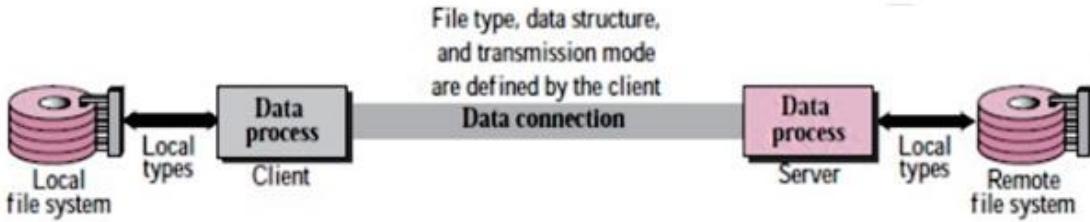
To communicate over control connection FTP uses TELNET or SMTP. Communication over control connection is done by commands and responses. The first command is sent over the connection and in return, a response is sent by another system. We can send a command or response at a time. There is only one-way communication.



**Figure 5.8: Communication over the control connection**

### Communication over the data connection

For transforming file over the data connection, the client must define the type of file which needs to be transformed, transmission mode, and the data structure. It solves the heterogeneity problem by defining these three attributes.



**Figure 5.9: Communication over the data connection**

## COMMAND PROCESSING

To establish communication between the client system and the server system FTP uses a control connection. During this process, the client sends commands to the server and in return, the server sends a response to the client.

### Types of FTP Transfers

It can transfer following file types across the internet connections

1. **ASCII file:** This is the default format for transforming a file from one to another. Each character is encoded by NVT ASCII i.e. Network Virtual Terminal ASCII character set. Both the sender and the receiver transform their file from its own representation into NVT ASCII.
2. **EBCDIC:** If sender or receiver connections use the EBCDIC encoding method, then for transforming file FTP uses EBCDIC encoding.
3. **Image File:** For transforming the binary file, the image file is the default mode. The file is transformed over the internet connections in the form of stream bits without encoding.

## TFTP

Trivial File Transfer Protocol, is a simple high-level protocol for transferring data servers use to boot diskless workstations, X-terminals, and routers by using User Data Protocol (UDP).

Although it may sound similar, TFTP works differently than FTP (File Transfer Protocol) and HTTP (HyperText Transfer Protocol). Although TFTP is also based in FTP technology, TFTP is an entirely different protocol. Among the differences is that TFTP's transport protocol uses UDP which is not secure while FTP uses Transmission Control Protocol (TCP) to secure information.

TFTP was primarily designed to read or write files by using a remote server. However, TFTP is a multi-purpose protocol that can be leveraged for an array of different tasks.

### TFTP Configuration Uses

TFTP configuration used for:

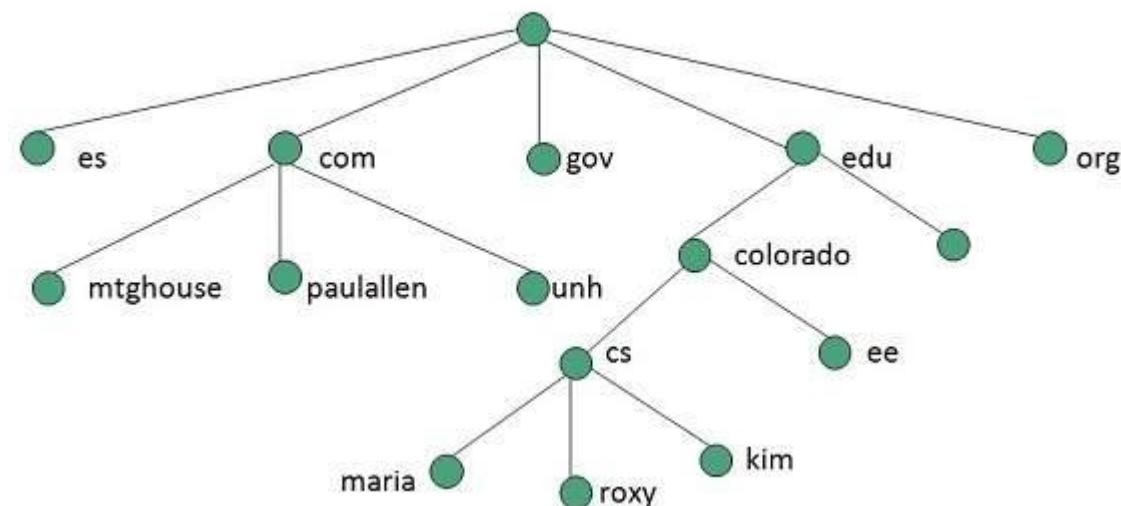
- Transferring files
- Remote-booting without hard drives
- Upgrading codes

- Backing up network configurations
- Backing up router configuration files
- Saving IOS images
- Booting PCs without a disk

After a workstation has been booted from a network card's ROM, the TFTP installation will download a program and then run it from a central server.

## DNS

The domain name space refers a hierarchy in the internet naming structure. This hierarchy has multiple levels (from 0 to 127), with a root at the top. The following diagram shows the domain name space hierarchy:



**Figure 5.5: Domain name space**

DNS is an industry-standard service used to locate computers on an Internet Protocol (IP)-based network. IP networks, such as the Internet and Windows 2000 networks, rely on number-based addresses, such as 207.46.131.137 to ferry information throughout the network. Network users rely on character-based names, such as www.microsoft.com. Therefore, it is necessary to translate character or user-friendly addresses (www.microsoft.com) into the number-based addresses (207.46.131.137) that the network can recognize.

DNS replaced the hosts file's flat name space with a hierarchical name space. With a hierarchical name space, information about host names and IP addresses can be partitioned and distributed, thus, scalability is achieved. For example, in the fictional widgets.products.microsoft.com domain, responsibility for name resolution can be partitioned so that various servers can handle name resolution for different parts of the name space:

- One server can be responsible for resolving the first part (microsoft.com), and can then forward the name-resolution request to the next DNS Server in the hierarchy.
- The next DNS Server can be responsible for resolving the next part of the name space.
- Finally, the request can be forwarded to a third DNS server that is responsible for resolving the last part of the name.

## DISTRIBUTION OF NAME SPACES

A namespace is a context within which the names of all objects must be unambiguously resolvable. For example, the internet is a single DNS name space, within which all network devices with a DNS name can be resolved to a particular address (for example, www.microsoft.com resolves to 207.46.131.13).

## DNS IN THE INTERNET

DNS translates the domain name into IP address automatically. Following are the steps of domain resolution process:

- When user type www.cdgi.com into the browser, it asks the local DNS Server for its IP address.
- When the local DNS does not find the IP address of requested domain name, it forwards the request to the root DNS server and again enquires about IP address of it.
- The root DNS server replies with delegation that user do not know the IP address of www.cdgi.com but know the IP address of DNS Server.
- The local DNS server then asks the com DNS Server the same question.
- The com DNS Server replies the same that it does not know the IP address of www.cdgi.com but knows the address of cdgi.com.
- Then the local DNS asks the cdgi.com DNS server the same question.
- Then cdgi.com DNS server replies with IP address of www.cdgi.com.
- Now, the local DNS sends the IP address of www.cdgi.com to the computer that sends the request.

## SSH Protocol

SSH stands for Secure Shell or Secure Socket Shell. It is a cryptographic network protocol that allows two computers to communicate and share the data over an insecure network such as the internet. It is used to login to a remote server to execute commands and data transfer from one machine to another machine.

A simple example can be understood, such as suppose you want to transfer a package to one of your friends. Without SSH protocol, it can be opened and read by anyone. But if you will send it using SSH protocol, it will be encrypted and secured with the public keys, and only the receiver can open it.

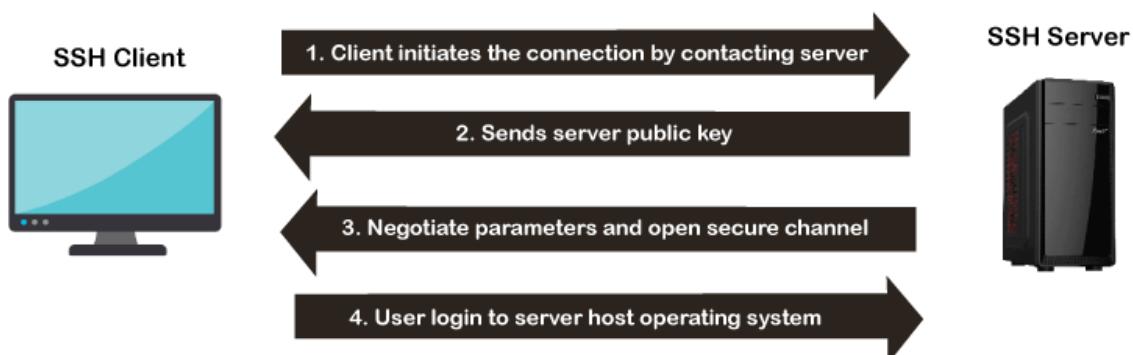
### Usages of SSH protocol

- It provides secure access to users and automated processes.
- It is an easy and secure way to transfer files from one system to another over an insecure network.
- It also issues remote commands to the users.
- It helps the users to manage the network infrastructure and other critical system components

- It is used to log in to shell on a remote system (Host), which replaces **Telnet** and **rlogin** and is used to execute a single command on the host, which replaces **rsh**.
- It combines with **rsync** utility to backup, copy, and mirror files with complete security and efficiency.
- It can be used for forwarding a port.
- By using SSH, we can set up the automatic login to a remote server such as OpenSSH.
- We can securely browse the web through the encrypted proxy connection with the SSH client, supporting the SOCKS protocol.

### How does SSH Works?

The SSH protocol works in a **client-server** model, which means it connects a secure shell client application (End where the session is displayed) with the SSH server (End where session executes).



## **E-MAIL**

### **SMTP**

SMTP stands for Simple Mail Transfer Protocol and it's the industry standard protocol for email sending.

With SMTP users are sending, relaying, or forwarding messages from a mail client (like Microsoft Outlook) to a receiving email server. A sender will use an SMTP server to carry out the process of transmitting an email message.

If user looking to enable email sending within the application, then he must use SMTP over IMAP.

### **MIME(Multipurpose Internet Mail Extension Protocol):**

Multipurpose Internet Mail Extension Protocol is an additional email protocol that allows non-ASCII data to be sent through SMTP. It allows users to send and receive different types of data like audio, images, videos and other application programs on the Internet. It allows to send multiple attachments with single message. It allows to send message of unlimited length.

### **POP**

POP stands for Post Office Protocol. And the number three stands for "version 3," which is the latest version and the most widely used — hence the term "POP3."

POP3 downloads the email from a server to a single computer, and then deletes the email from the server. On the other hand, IMAP stores the message on a server and synchronizes the message across multiple devices.

### **IMAP**

IMAP (Internet Access Message Protocol) is an email protocol that deals with managing and retrieving email messages from the receiving server. Since IMAP deals with message retrieval, user will not be able to use the IMAP protocol to send email. Instead, IMAP will be used for receiving messages.

### **SNMP**

Simple Network Management Protocol (SNMP) is a way for different devices on a network to share information with one another. It allows devices to communicate even if the devices are different hardware and run different software. Without a protocol like SNMP, there would be no way for network management tools to identify devices, monitor network performance, keep track of changes to the network, or determine the status of network devices in real time.

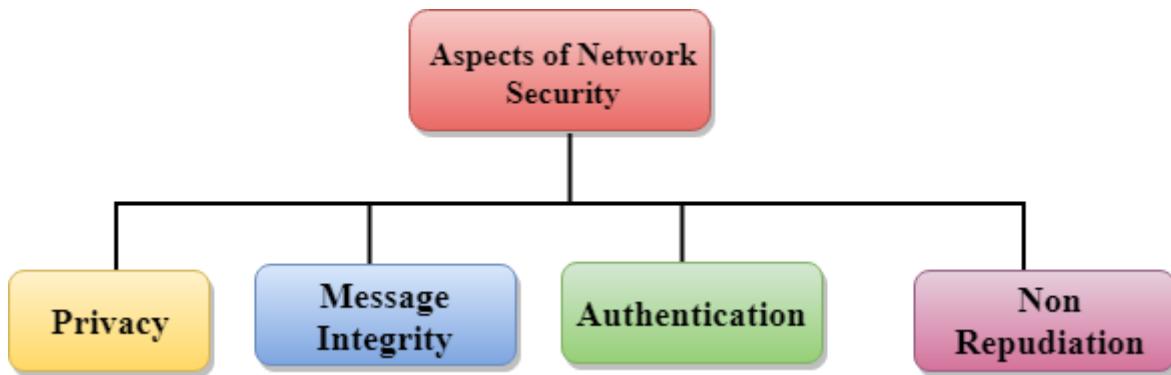
SNMP has a simple architecture based on a client-server model. The servers, called managers, collect and process information about devices on the network.

The clients, called agents, are any type of device or device component connected to the network. They can include not just computers but also network switches, phones, printers. Some devices may have multiple device components. For example, a laptop typically contains a wired as well as a wireless network interface.

All the measures used to safeguard a computer network's integrity and the data on it are collectively referred to as network security. Network security is crucial because it protects sensitive data from online threats and guarantees the network's dependability. Multiple security measures are used in successful network security plans to shield users and organizations from malware and online threats like distributed denial of service.

#### Aspects of Network Security

Following are the desirable properties to achieve secure communication:



#### Cryptography –

**Traditional Ciphers:-** Traditional ciphers encode messages in a way that only those with the secret key can decode it. Throughout history, it has been used for private communication.

These ciphers were used extensively before the advent of modern computer encryption techniques. Although relatively simple, when used properly, it can also be effective and provide interesting insights into the history of encryption.

Here are a few common types –

- Caesar Cipher
- Vigenere Cipher
- Simple Substitution Cipher
- Monoalphabetic and Polyalphabetic Cipher
- Transposition Cipher
- Playfair Cipher

#### Caesar Cipher

Caesar cipher is a type of substitution cipher in which every letter in the given plaintext is shifted with a certain number of places down or up the alphabet. The shift number is called the key. It is created by Julius Caesar. It is believed that this cipher is used to communicate secretly with his generals.

For this type of method both the sender and the receiver should agree on a 'secret shift key' for shifting the alphabet of the given plaintext. The number which can be between 0 and 25 is the key of encryption.

For example - If plaintext is shift of 4:

- 'A' will become 'E'
- 'B' will become 'F'
- 'C' will become 'G'
- and so on...

### **Vigenere Cipher**

Vigenere is also a traditional cipher algorithm which uses a text string (Let us say a word) as a key. Then it is used for shifting a number of shifts on the given plaintext.

For example – let us say the key is 'tutor'. In this each alphabet of the key is changed to its respective numeric value: In our case,

t → 20, u → 21, t → 20, o → 15, and r → 18.

Thus, the key is: 20 21 20 15 18.

### **Simple Substitution Cipher**

Simple substitution cipher is a form of encryption in which the letters in Plaintext is replaced by another character in cipher text according to a fixed order. In other words it is a method of encoding with any letter and characters are mapped to other characters.

For example, in Simple Substitution Cipher –

- 'A' can be replaced by 'X'
- 'B' can be replaced by 'K'
- 'C' can be replaced by 'Q'
- So on... until each character in the alphabet is replaced by another one.

### **Monoalphabetic and Polyalphabetic Cipher:**

A Monoalphabetic cipher is a type of substitution cipher in which fixed characters are always replaced in the ciphertext which means that every character in the plaintext is always replaced by the same corresponding character in the ciphertext. The replacement remains constant throughout the encryption process.

For example, if 'A' is replaced by 'E' in the encryption process, any occurrence of 'A' in plain text will be replaced by 'E' in ciphertext and if 'B' is replaced by 'F'. replace 'any B ' in plaintext with 'F' in ciphertext.

### Transposition Cipher

This is a new type of cipher in which the sequence of letters is rearranged in plain text to create ciphertext. They are not replaced by actual plaintext.

An example is a simple column substitution cipher in which simple increments are written in a specific alphabet width. The ciphertext is then read directly as indicated.

For example, the plain text is "The tajmahal is in agra" and the randomly chosen hidden key is "Four". We format this text directly in a table with a number of columns including key value. The resulting text is shown below.

T	h	e	t
a	j	m	a
h	a	l	i
s	i	n	a
g	r	a	

So now we can get the ciphertext just reading columns from first to last by vertically downward. And the ciphertext will be 'Tahsghjairemlnataia'.

### Playfair Cipher

The Playfair cipher uses a 5x5 grid of letters unless it is double and usually 'J' for encryption. Messages are encrypted by pairing letters and using rules: if on the same letter, turn right; If it is in one column, turn down; If there are rows and columns, make a triangle and replace them with opposite corners. Decryption follows the opposite rule. Playfair encrypts character pairs, making it more secure than single-character ciphers, but still vulnerable to an attack.

### Modern Ciphers - Base64 Encoding & Decoding

Details of Base64 encoding

Base64 refers to a group of related encoding techniques that encode binary data numerically and translate it into a base-64 representation. The word Base64 comes from a specific MIME-content transfer encoding.

Base64 has a specific set of characters –

- 26 Uppercase letters

- 26 Lowercase letters
- 10 Numbers
- Plus sign (+) and slash (/) for new lines

## Applications

Base64 encoding is often used in a variety of applications. It is widely used to encrypt binary data, especially if it needs to be communicated by email or used in other text fields. It is also used in a variety of web and internet protocols, as well as to encode digital signatures and certificates.

## Limitations

Here are the limitations of Base64 encoding –

- Base64 encoding usually maximizes the size of the data by about 33%. This can impact transmission and storage efficiency, mainly for large datasets.
- It is basically not a form of encryption. It simply converts data into a different format. As a result, sensitive information is not kept confidential or secure.
- Base64 adds padding characters ('=') to the end of the encoded data to ensure it aligns properly. This way it can complicate parsing and processing of the encoded data.
- It uses a limited set of characters (A-Z, a-z, 0-9, '+', '/') for encoding. This can lead to issues when the encoded data needs to be transmitted or processed in systems that have restrictions on certain characters.
- The encoding of Base64 does not compress data. It is only meant for representing binary data in a text-based format, not for reducing file size.

## Message Integrity in Cryptography:-

Message integrity in cryptography is the process of checking the message's authenticity. It ensures the message has not been altered or tampered with. Message integrity means that a message has not been tampered with or manipulated. Integrity is important because it ensures that the message has not been modified or tampered with.

Message integrity is commonly used in computing systems for integrity verification and information authentication. They are regarded cryptographically “weak” since they can be solved in polynomial time but are difficult to interpret.

Message integrity enhances traditional hash algorithms with security characteristics, making it more difficult to discover message content or receiver and sender information.

## Steps to Verify the Integrity of a Message

- **Message Authentication Codes:** Suppose two users, a sender, and a receiver, want to connect via messages. In MAC, or Message Authentication Codes, the transmitter and receiver use the same MAC algorithm or key.

- **Certificates:** A certificate is a digital document that validates a public key. The certificate provides information about the key, the owner's identity, and the organization's digital signature, which has verified the certificate's contents.
- **Nonrepudiation:** Nonrepudiation is the property of agreeing to adhere to an obligation. More specifically, it is the inability to refute responsibility.
- **Message Authentication Codes:** In the case of MAC, there is no public key. There is just one private key, which is known only to the sender and receiver. As a result, there is no interference from external parties. Even if a third-party user had access to the secret key, he could not guarantee that either the sender or the recipient signed the message because both can encrypt or decrypt it.

### Message authentication

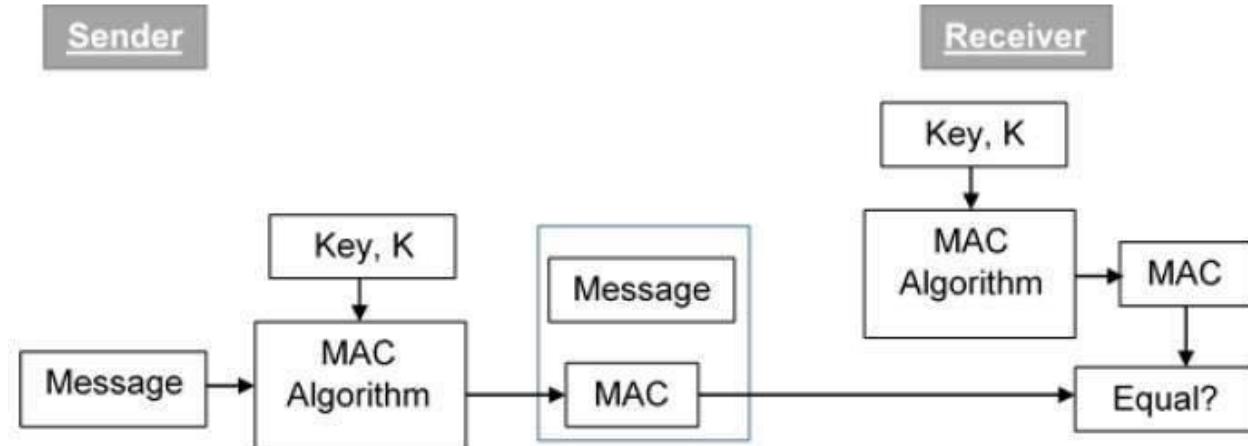
Message authentication can be provided using the cryptographic techniques that use secret keys as done in case of encryption.

Message Authentication Code (MAC):

MAC algorithm is a symmetric key cryptographic technique to provide message authentication. For establishing MAC process, the sender and receiver share a symmetric key K.

Essentially, a MAC is an encrypted checksum generated on the underlying message that is sent along with a message to ensure message authentication.

The process of using MAC for authentication is depicted in the following illustration –



Let us now try to understand the entire process in detail –

- The sender uses some publicly known MAC algorithm, inputs the message and the secret key K and produces a MAC value.
- Similar to hash, MAC function also compresses an arbitrary long input into a fixed length output. The major difference between hash and MAC is that MAC uses secret key during the compression.
- The sender forwards the message along with the MAC. Here, we assume that the message is sent in the clear, as we are concerned of providing message origin authentication, not confidentiality. If confidentiality is required then the message needs encryption.
- On receipt of the message and the MAC, the receiver feeds the received message and the shared secret key K into the MAC algorithm and re-computes the MAC value.
- The receiver now checks equality of freshly computed MAC with the MAC received from the sender. If they match, then the receiver accepts the message and assures himself that the message has been sent by the intended sender.
- If the computed MAC does not match the MAC sent by the sender, the receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified. As a bottom-line, a receiver safely assumes that the message is not the genuine.

**Chameli Devi Group of Institutions**

**Department: Artificial Intelligence and Data Science**

**Subject: Computer Networks (AD602)**

**UNIT I: Computer Network:** Definitions, goals, components, Architecture, Classifications & Types. **Layered Architecture:** Protocol hierarchy, Design Issues, Interfaces and Services, Connection Oriented & Connectionless Services, Service primitives, Design issues & its functionality. **ISOOSI Reference Model:** Principle, Model, Descriptions of various layers and its comparison with TCP/IP. **Principals of physical layer:** Media, Bandwidth, Data rate and Modulations.

---

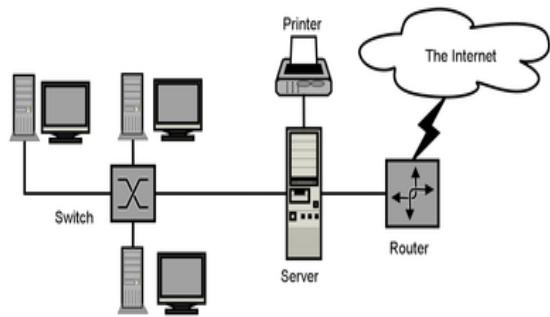
**Course Objective-** To provide students with an overview of the concepts and fundamentals of computer networks . To familiarize with the basic taxonomy and terminology of computer networking area.

**Course Outcome-** Characterize and appreciate computer networks from the view point of components and from the view point of services (Knowledge & design of networks)

**Definition-** Imagine you have several computers, and you want them to talk to each other and share information.

A computer network is like a digital neighborhood that connects these computers so they can communicate and share resources, such as files, printers, or an internet connection.

In simpler words, a computer network is a way for computers to connect and work together, allowing them to share and exchange information. It's like a digital community where devices can interact and collaborate.



### Key Elements:

- **Nodes:** Nodes are the individual devices connected to the network. Examples include computers, servers, printers, and other devices.
- **Links:** Links refer to the communication channels that connect nodes in a network. These can be wired, such as Ethernet cables, or wireless, like Wi-Fi connections.
- **Protocols:** Protocols are a set of rules and conventions that govern how data is transmitted and received across a network. Examples include TCP/IP (Transmission Control Protocol/Internet Protocol), which is fundamental to the functioning of the Internet.

**Topology:** Topology refers to the physical or logical layout of a network. Common topologies include star, bus, ring, and mesh.

### Goals of Computer Network:

The goals of a computer network can vary depending on the specific requirements and objectives of the organization or individuals implementing the network.

- **Resource Sharing:** One of the primary goals of computer networks is to facilitate the sharing of resources such as files, printers, and applications among connected devices. This enables efficient use of resources and reduces redundancy.
- **Data Communication:** Computer networks provide a means for seamless and efficient communication of data between devices. This includes the transmission of messages, files, and other forms of data.
- **Remote Access:** Networks enable remote access to resources and services. Users can access data and applications from different locations, promoting flexibility and mobility.
- **Reliability and Availability:** Networks are designed to provide reliable and available access to resources. Redundancy, fault tolerance, and other measures are implemented to ensure continuous operation even in the face of hardware failures or disruptions.

- **Cost Efficiency:** Sharing resources through a network can lead to cost savings. For example, multiple users can share a single printer or a centralized server can store and manage data for an entire organization.
- **Scalability:** Computer networks are designed to be scalable, allowing for the addition of new devices and resources as the organization or user's needs grow.
- **Data Security:** Ensuring the security of data is a key goal of computer networks. Measures such as encryption, firewalls, and access controls are implemented to protect sensitive information from unauthorized access or malicious activities.

### **Components Of Computer Network-**

A computer network consists of various components that work together to enable communication and resource sharing among connected devices. These components can be categorized into hardware and software components:

#### **Hardware Components:**

- **Devices/Nodes:** These are the individual devices that are part of the network. Examples include computers, servers, routers, switches, printers, and other networked devices.
- **Network Cables and Connectors:** Physical connections between devices are established using network cables, such as Ethernet cables. Connectors like RJ45 plugs are commonly used for wired connections.
- **Hubs:** While less common today, hubs were used to connect multiple devices in a network. Unlike switches, hubs do not intelligently manage data traffic.
- **Modems:** Modems (Modulator-Demodulator) are used to convert digital data from a computer into analog signals for transmission over telephone lines or other communication channels.
- **Firewalls:** Firewalls can be both hardware and software components. They are designed to protect a network from unauthorized access and ensure the security of data

#### **Software Components:**

**Network Operating System (NOS):** A network operating system is specialized software that provides network services, such as file sharing, printer sharing, and user authentication. Examples include Windows Server, Linux, and Novell NetWare.

**Device Drivers:** Device drivers enable the communication between the operating system and the network hardware (e.g., NIC drivers).

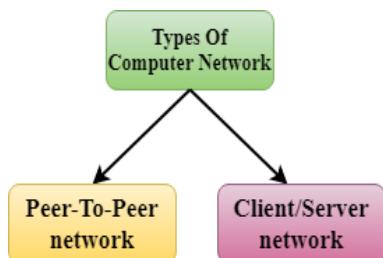
**Protocol Suites:** Protocols are sets of rules governing communication between devices in a network. Common protocol suites include TCP/IP (Transmission Control Protocol/Internet Protocol), which is foundational for the Internet.

**Network Services:** Various network services are provided by software components, including DNS (Domain Name System) for translating domain names to IP addresses, DHCP (Dynamic Host Configuration Protocol) for automatic IP address assignment, and more.

## Computer Network Architecture

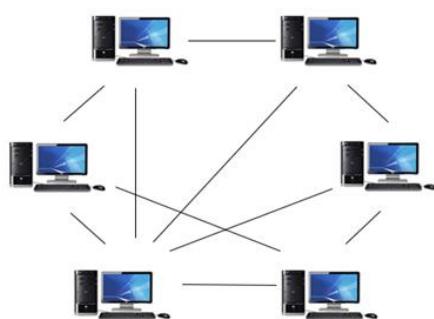
Computer Network Architecture is defined as the physical and logical design of the software, hardware, protocols, and media of the transmission of data. Simply we can say that how computers are organized and how tasks are allocated to the computer.

The two types of network architectures are used:



### Peer-To-Peer network

- Peer-To-Peer network is a network in which all the computers are linked together with equal privilege and responsibilities for processing the data.
- Peer-To-Peer network is useful for small environments, usually up to 10 computers.
- Peer-To-Peer network has no dedicated server.
- Special permissions are assigned to each computer for sharing the resources, but this can lead to a problem if the computer with the resource is down.



### Advantages of Peer-To-Peer Network:

- It is less costly as it does not contain any dedicated server.
- If one computer stops working but, other computers will not stop working.

### **Disadvantages of Peer-To-Peer Network:**

- In the case of Peer-To-Peer network, it does not contain the centralized system . Therefore, it cannot back up the data as the data is different in different locations.
- It has a security issue as the device is managed itself.

### **Client/Server Network:**

- Client/Server network is a network model designed for the end users called clients, to access the resources such as songs, video, etc. from a central computer known as Server.
- The central controller is known as a **server** while all other computers in the network are called **clients**.
- A server performs all the major operations such as security and network management.
- A server is responsible for managing all the resources such as files, directories, printer, etc.
- All the clients communicate with each other through a server. For example, if client1 wants to send some data to client 2, then it first sends the request to the server for the permission. The server sends the response to the client 1 to initiate its communication with the client 2.



### **Advantages of Client/Server network:**

- A Client/Server network contains the centralized system. Therefore we can back up the data easily.
- A Client/Server network has a dedicated server that improves the overall performance of the whole system.

### **Disadvantages of Client/Server network:**

- Client/Server network is expensive as it requires the server with large memory.
- A server has a Network Operating System(NOS) to provide the resources to the clients, but the cost of NOS is very high.

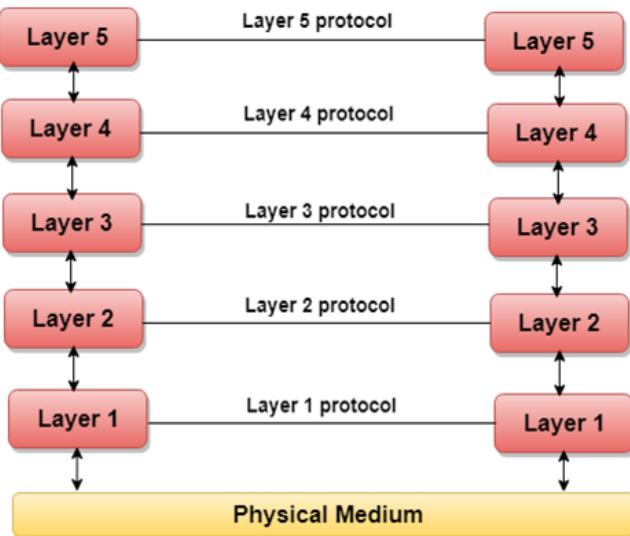
**Types of Computer Network-** There are computer networks that help in building social relationships. These networks depend upon the speed at which the internet is provided. All these three networks, i.e., LAN, MAN, and WAN, are used for providing internet services to people.

Let us discuss about them in detail:

Basis	LAN	MAN	WAN
Full-Form	<a href="#">LAN stands for local area network.</a>	<a href="#">MAN stands for metropolitan area network.</a>	<a href="#">WAN stands for wide area network.</a>
Geographic Span	Operates in small areas such as the same building or campus.	Operates in large areas such as a city.	Operates in larger areas such as country or continent.
Ownership	LAN's ownership is private.	MAN's ownership can be private or public.	While WAN also might not be owned by one organization.
Transmission Speed	The transmission speed of a LAN is high.	While the transmission speed of a MAN is average.	Whereas the transmission speed of a WAN is low.
Propagation delay	The propagation delay is short in a LAN.	There is a moderate propagation delay in a MAN.	Whereas, there is a long propagation delay in a WAN.
Congestion	There is less congestion in LAN.	While there is more congestion in MAN.	Whereas there is more congestion than MAN in WAN.
Design & Maintenance	LAN's design and maintenance are easy.	While MAN's design and maintenance are difficult than LAN.	Whereas WAN's design and maintenance are also difficult than LAN as well as MAN.
Fault tolerance	There is more fault tolerance in LAN.	While there is less fault tolerance.	In WAN, there is also less fault tolerance.

### Layered Architecture-

- The main aim of the layered architecture is to divide the design into small pieces.
- Each lower layer adds its services to the higher layer to provide a full set of services to manage communications and run the applications.
- It provides modularity and clear interfaces, i.e., provides interaction between subsystems.
- The basic elements of layered architecture are services, protocols, and interfaces.
- **Service:** It is a set of actions that a layer provides to the higher layer.
- **Protocol:** It defines a set of rules that a layer uses to exchange the information with peer entity. These rules mainly concern about both the contents and order of the messages used.
- **Interface:** It is a way through which the message is transferred from one layer to another layer.



- layer 1 is the physical medium through which the actual communication takes place.
- In a layered architecture, unmanageable tasks are divided into several small and manageable tasks.
- A set of layers and protocols is known as network architecture.

## CONNECTION ORIENTED AND CONNECTIONLESS SERVICES

### Connection Oriented Service:

It is a three-phase process which include-

- Connection Establishment
- Data Transfer
- Termination / disconnection

In this type of transmission, the receiving device sends an acknowledgement, back to the source after a packet or group of packet is received. This type of transmission is reliable and secure.

**Connection less service:** It is a one-phase process and includes Data Transfer. In this type of transmission, the receiver does not acknowledge receipt of a packet. This approach allows for much faster communication between devices. Connection-oriented service is more reliable than connectionless Service.

Criteria	Connection-Oriented	Connection-Less
<b>Connection</b>	Prior connection needs to be established.	No prior connection is established.
<b>Resource Allocation</b>	Resources need to be allocated.	No prior allocation of resource is required.
<b>Reliability</b>	It ensures reliable transfer of data.	Reliability is not guaranteed as it is a best effort service.
<b>Congestion</b>	Congestion is not at all possible.	Congestion can occur likely.
<b>Transfer mode</b>	It can be implemented either using Circuit Switching or VCs.	It is implemented using Packet Switching.
<b>Retransmission</b>	It is possible to retransmit the lost data bits.	It is not possible.
<b>Suitability</b>	It is suitable for long and steady communication.	It is suitable for bursty transmissions.
<b>Signaling</b>	Connection is established through process of signaling.	There is no concept of signaling.
<b>Packet travel</b>	In this packets travel to their destination node in a sequential manner.	In this packets reach the destination in a random manner.
<b>Delay</b>	There is more delay in transfer of information, but once connection established faster delivery.	There is no delay due absence of connection establishment phase.

## SERVICE PRIMITIVES

Service generally includes set of various primitives. A primitive simply means Operations.

A Service is specified by set of primitives that are available and given to user or other various entities to access the service. All these primitives simply tell the service to perform some action or to report on action that is taken by peer entity. Each of the protocol that communicates in layered architecture also communicates in peer-to-peer manner with some of its remote protocol entity.

Primitives are called calling functions between the layers that are used to manage communication among the adjacent protocol layers i.e., among the same communication node. The set of primitives that are available generally depends upon the nature of the service that is being provided.

### Classification of Service Primitives:

Primitive	Meaning
<b>Request</b>	It represents entity that wants or request service to perform some action or do some work (requesting for connection to remote computer).
<b>Indication</b>	It represent entity that is to be informed about event (receiver just have received request of connection).
<b>Response</b>	It represents entity that is responding to event (receiver is simply sending the permission or allowing to connect).

**Confirm**

It represent entity that acknowledges the response to earlier request that has come back (sender just acknowledge the permission to get connected to the remote host).

### Design issues-

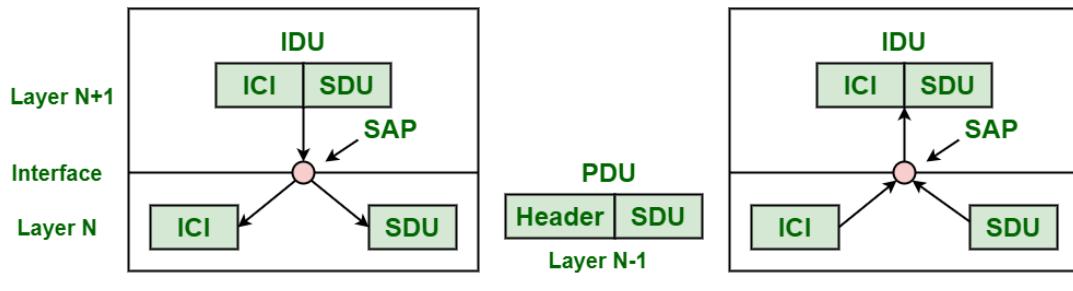
- **Topology:** Choose an appropriate network topology.
- **Size and Scale:** Determine the network's size and scale.
- **Scalability:** Design for future growth.
- **Performance:** Address bandwidth and latency issues.
- **Reliability:** Ensure continuous operation with redundancy.
- **Security:** Implement measures to protect against threats.
- **Cost:** Balance performance with budget constraints.
- **Management:** Design for easy configuration and troubleshooting.
- **Interoperability:** Ensure compatibility with different devices.
- **QoS:** Address specific application requirements for quality of service.
- **Addressing and Naming:** Develop strategies for IP addressing and domain naming.
- **Documentation:** Maintain comprehensive documentation.
- **Environmental Considerations:** Account for physical factors like temperature and power.
- **User Requirements:** Align the design with end-user and application needs.

## INTERFACES AND SERVICES

Interfaces and Services is a process that generally provides and gives a common technique for each layer to communicate with each other. Standard terminology basically required for layered networks to request and aim for the services are provided.

Service is defined as a set of primitive operations. Services are provided by layer to each of layers above it.

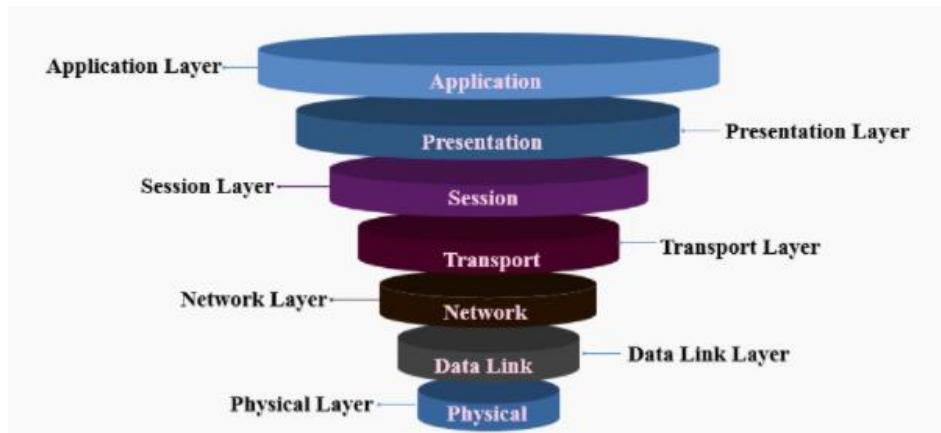
Below is diagram showing relation between layers at an interface. In diagram, layers N+1, N, and N-1 are involved and engaged in process of communication among each other.



**Relationship Between Layers at an Interface**

## ISO-OSI REFERENCE MODEL

OSI stands for Open Systems Interconnection. It has been developed by ISO – ‘International Organization of Standardization’, in the year 1984. It is a 7 layer architecture with each layer having specific functionality to perform.



**Figure 1.1: OSI reference model**

### Layer 1: Physical layer

The physical layer has the following major functions:

- It defines the electrical and physical specifications of the data connection. It defines the relationship between a device and a physical transmission medium (e.g., a copper or fiber optical cable). This includes the layout of pins, voltages, line impedance, cable specifications, signal timing, hubs, repeaters, network adapters, host bus adapters (HBA used in storage area networks) and more.
- It defines the protocol to establish and terminate a connection between two directly connected nodes over a communications medium.
- It may define the protocol for flow control.
- It defines transmission mode i.e. simplex, half duplex, full duplex.
- It defines the topology.
- It defines a protocol for the provision of a (not necessarily reliable) connection between two directly connected nodes, and the modulation or conversion between the representation of digital data in user Equipment and the corresponding signals transmitted over the physical communications channel.
- Cabling system components

- Adapters that connect media to physical interfaces
- Connector design and pin assignments
  - Hub, repeater, and patch panel specifications
  - Wireless system components
  - Parallel SCSI (Small Computer System Interface)
  - Network Interface Card (NIC)

## **Layer 2: Data link layer**

The data link layer provides node-to-node data transfer - A reliable link between two directly connected nodes, by detecting and possibly correcting errors that may occur in the physical layer. The data link layer is divided into two sublayers:

- Media Access Control (MAC) layer - Responsible for controlling how devices in a network gain access to data and permission to transmit it.
- Logical Link Control (LLC) layer - Controls error checking and packet synchronization.

### **Basic Functions**

- Allows a device to access the network to send and receive messages
- Offers a physical address so a device's data can be sent on the network
- Works with a device's networking software when sending and receiving messages
- Provides error-detection capability

### **Common networking components that function at layer 2 include:**

- Network interface cards
- Ethernet and Token Ring switches
- Bridges

## **Layer 3: Network layer**

The network layer provides the functional and procedural means of transferring variable length data sequences (called datagrams) from one node to another connected to the same network.

- It translates logical network address into physical machine address.
- Routing is also one of the main functions of the Network Layer, routing is the process of selecting paths in a network over which to send packets.
- Internet Control Message Protocol (ICMP) is network layer protocol and one of the main protocols of the Internet Protocol suite and is used for error handling and diagnostic purposes. Quality of Service (QOS) although not the primary function of the network layer is available in network layer protocols such as the Internet Protocol which allows certain traffic to be prioritized over other giving it preferential treatment.

#### **Layer 4: Transport layer**

The transport layer provides the functional and procedural means of transferring variable-length data sequences from a source to a destination host via one or more networks while maintaining the quality of service functions.

An example of a transport-layer protocol in the standard Internet stack is Transmission Control Protocol (TCP), usually built on top of the Internet Protocol (IP).

Some of the functions offered by the transport layer include:

- Application identification
- Client-side entity identification
- Confirmation that the entire message arrived intact
- Segmentation of data for network transport
- Control of data flow to prevent memory overruns
- Establishment and maintenance of both ends of virtual circuits
- Transmission-error detection
- Realignment of segmented data in the correct order on the receiving side
- Multiplexing or sharing of multiple sessions over a single physical link

#### **Layer 5: Session layer**

The session layer controls the dialogues (connections) between computers. It establishes, manages and terminates the connections between the local and remote application.

It provides for full-duplex, half-duplex, or simplex operation, and establishes check pointing, adjournment, termination, and restart procedures. This session layer allows applications functioning on devices to establish, manage, and terminate a dialog through a network. Session layer functionality includes:

- Virtual connection between application entities
- Synchronization of data flow
- Creation of dialog units
- Connection parameter negotiations
- Partitioning of services into functional groups
- Acknowledgements of data received during a session
- Retransmission of data if it is not received by a device

## **Layer 6: Presentation layer**

The presentation layer, is responsible for how an application formats the data to be sent out onto the network. The presentation layer basically allows an application to read (or understand) the message.

Examples of presentation layer functionality include:

- Encryption and decryption of a message for security
- Compression and expansion of a message so that it travels efficiently
- Graphics formatting
- Content translation
- System-specific translation

## **Layer 7: Application layer**

The application layer, provides an interface for the end user operating a device connected to a network.

This layer is what the user sees, in terms of loading an application (such as Web browser or e-mail).

Examples of application layer functionality include:

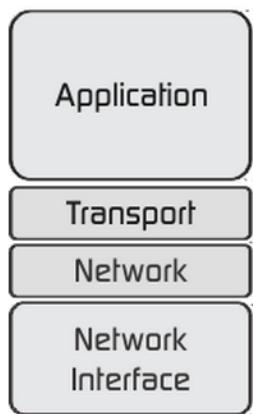
- Support for file transfers
- Ability to print on a network
- Electronic mail
- Electronic messaging
- Browsing the World Wide Web

## **TCP/IP MODEL**

The TCP/IP reference model is the network model used in the current Internet architecture. It is considered as the grandfather of the Internet the ARPANET. The reference model was named after two of its main protocols, TCP (Transmission control Protocol) and IP (Internet Protocol).

There are versions of this model with four layers and with five layers. The original four-layer version of the model is shown below.

## TCP/IP model



**Layer 4:** Process Layer or Application Layer: This is where the “higher level” protocols such as FTP, HTTP, etc. Operate. The original TCP/IP specification described many different applications that fit into the top layer of the protocol stack. These applications include Telnet, FTP, SMTP, and DNS.

**Layer 3:** Host-To-Host (Transport) Layer: This is where flow-control and connection protocols exist, such as TCP. This layer deals with opening and maintaining a connection, ensuring that packet is in fact received the transport layer is the interface between the application layer and the complex hardware of the.

Two modes are available, full-duplex and half-duplex. In full-duplex operation, both sides can transmit and receive data simultaneously, whereas, in half duplex, a side can only send or receive at one time.

**Layer 2:** Internet or Internetworking Layer: This layer defines IP addresses, with many routing schemes for navigating packets from one IP address to another. The job of the network layer is to inject packets into any network and have them travel independently to the destination. Packet routing is a major job of this protocol.

**Layer 1:** Networking Access Layer: This layer describes the physical equipment necessary for communications, such as twisted pair cables, the signaling used on that equipment, and the low-level protocols using that signaling. That Host-to-Network layer interfaces the TCP/IP protocol stack to the physical network.

### Principal of physical layer-

The Physical Layer is the first layer of the OSI (Open Systems Interconnection) model and is primarily concerned with the physical transmission of data bits over a physical medium.

It deals with the hardware aspects of transmitting raw binary data over a physical link or medium, without concern for the higher-layer protocols.

The physical layer converts the data frame received from the data link layer into bits, i.e., in terms of ones and zeros. It maintains the data quality by implementing the required protocols on different network modes and maintaining the bit rate through data transfer using a wired or wireless medium.

## **Attributes of the Physical Layer-**

**1. Signals:** The data is first converted to a signal for efficient data transmission. There are two kinds of signals:

- **Analog Signals:** These signals are continuous waveforms in nature and are represented by continuous electromagnetic waves for the transmission of data.
- **Digital Signals:** These signals are discrete in nature and represent network pulses and digital data from the upper layers.

**2. Transmission media:** Data is carried from source to destination with the help of transmission media. There are two sorts of transmission media:

- **Wired Media:** The connection is established with the help of cables. For example, fiber optic cables, coaxial cables, and twisted pair cables.
- **Wireless Media:** The connection is established using a wireless communication network. For example, Wi-Fi, Bluetooth, etc.

**Data Flow:** It describes the rate of data flow and the transmission time frame. The factors affecting the data flow are as follows:

- **Encoding:** Encoding data for transmission on the channel.
- **Error-Rate:** Receiving erroneous data due to noise in transmission.
- **Bandwidth:** The rate of transmission of data in the channel.

## **3.Modulation:**

- Amplitude Modulation (AM): Varies the amplitude of the carrier signal to represent data.
- Frequency Modulation (FM): Varies the frequency of the carrier signal.
- Phase Modulation (PM): Varies the phase of the carrier signal.

## **4. Transmission mode:**

It describes the direction of the data flow. Data can be transmitted in three sorts of transmission modes as follows:

- **Simplex mode:** This mode of communication is a one-way communication where a device can only send data. Examples are a mouse, keyboard, etc.
- **Half-duplex mode:** This mode of communication supports one-way communication, i.e., either data can be transmitted or received. An example is a walkie-talkie.
- **Full-duplex mode:** This mode of communication supports two-way communication, i.e., the device can send and receive data at the same time. An example is cellular communication.

## **Subject: Computer Networks (AD602)**

**UNIT I: Computer Network:** Definitions, goals, components, Architecture, Classifications & Types. **Layered Architecture:** Protocol hierarchy, Design Issues, Interfaces and Services, Connection Oriented & Connectionless Services, Service primitives, Design issues & its functionality. **ISOOSI Reference Model:** Principle, Model, Descriptions of various layers and its comparison with TCP/IP. **Principals of physical layer:** Media, Bandwidth, Data rate and Modulations.

**Course Objective-** To provide students with an overview of the concepts and fundamentals of computer networks . To familiarize with the basic taxonomy and terminology of computer networking area.

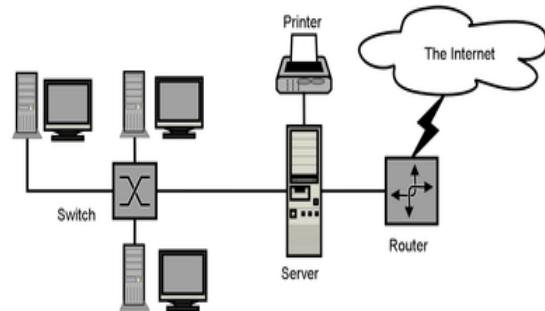
**Course Outcome-** Characterize and appreciate computer networks from the view point of components and from the view point of services (Knowledge & design of networks)

### **Computer Network:**

**Definition-** Imagine you have several computers, and you want them to talk to each other and share information.

A computer network is like a digital neighborhood that connects these computers so they can communicate and share resources, such as files, printers, or an internet connection.

In simpler words, a computer network is a way for computers to connect and work together, allowing them to share and exchange information. It's like a digital community where devices can interact and collaborate.



### **Key Elements:**

- **Nodes:** Nodes are the individual devices connected to the network. Examples include computers, servers, printers, and other devices.
- **Links:** Links refer to the communication channels that connect nodes in a network. These can be wired, such as Ethernet cables, or wireless, like Wi-Fi connections.
- **Protocols:** Protocols are a set of rules and conventions that govern how data is transmitted and received across a network. Examples include TCP/IP (Transmission Control Protocol/Internet Protocol), which is fundamental to the functioning of the Internet.

**Topology:** Topology refers to the physical or logical layout of a network. Common topologies include star, bus, ring, and mesh.

## **Goals of Computer Network:**

The goals of a computer network can vary depending on the specific requirements and objectives of the organization or individuals implementing the network.

- **Resource Sharing:** One of the primary goals of computer networks is to facilitate the sharing of resources such as files, printers, and applications among connected devices. This enables efficient use of resources and reduces redundancy.
- **Data Communication:** Computer networks provide a means for seamless and efficient communication of data between devices. This includes the transmission of messages, files, and other forms of data.
- **Remote Access:** Networks enable remote access to resources and services. Users can access data and applications from different locations, promoting flexibility and mobility.
- **Reliability and Availability:** Networks are designed to provide reliable and available access to resources. Redundancy, fault tolerance, and other measures are implemented to ensure continuous operation even in the face of hardware failures or disruptions.
- **Cost Efficiency:** Sharing resources through a network can lead to cost savings. For example, multiple users can share a single printer or a centralized server can store and manage data for an entire organization.
- **Scalability:** Computer networks are designed to be scalable, allowing for the addition of new devices and resources as the organization or user's needs grow.
- **Data Security:** Ensuring the security of data is a key goal of computer networks. Measures such as encryption, firewalls, and access controls are implemented to protect sensitive information from unauthorized access or malicious activities.

## **Components Of Computer Network-**

A computer network consists of various components that work together to enable communication and resource sharing among connected devices. These components can be categorized into hardware and software components:

### **Hardware Components:**

- **Devices/Nodes:** These are the individual devices that are part of the network. Examples include computers, servers, routers, switches, printers, and other networked devices.
- **Network Cables and Connectors:** Physical connections between devices are established using network cables, such as Ethernet cables. Connectors like RJ45 plugs are commonly used for wired connections.
- **Hubs:** While less common today, hubs were used to connect multiple devices in a network. Unlike switches, hubs do not intelligently manage data traffic.
- **Modems:** Modems (Modulator-Demodulator) are used to convert digital data from a computer into analog signals for transmission over telephone lines or other communication channels.

- **Firewalls:** Firewalls can be both hardware and software components. They are designed to protect a network from unauthorized access and ensure the security of data

### **Software Components:**

**Network Operating System (NOS):** A network operating system is specialized software that provides network services, such as file sharing, printer sharing, and user authentication. Examples include Windows Server, Linux, and Novell NetWare.

**Device Drivers:** Device drivers enable the communication between the operating system and the network hardware (e.g., NIC drivers).

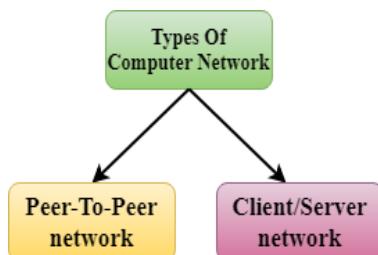
**Protocol Suites:** Protocols are sets of rules governing communication between devices in a network. Common protocol suites include TCP/IP (Transmission Control Protocol/Internet Protocol), which is foundational for the Internet.

**Network Services:** Various network services are provided by software components, including DNS (Domain Name System) for translating domain names to IP addresses, DHCP (Dynamic Host Configuration Protocol) for automatic IP address assignment, and more.

### **Computer Network Architecture**

Computer Network Architecture is defined as the physical and logical design of the software, hardware, protocols, and media of the transmission of data. Simply we can say that how computers are organized and how tasks are allocated to the computer.

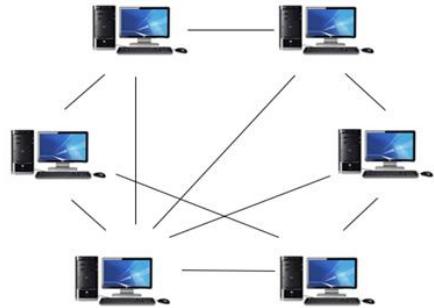
**The two types of network architectures are used:**



#### **Peer-To-Peer network**

- Peer-To-Peer network is a network in which all the computers are linked together with equal privilege and responsibilities for processing the data.
- Peer-To-Peer network is useful for small environments, usually up to 10 computers.
- Peer-To-Peer network has no dedicated server.

- Special permissions are assigned to each computer for sharing the resources, but this can lead to a problem if the computer with the resource is down.



#### **Advantages of Peer-To-Peer Network:**

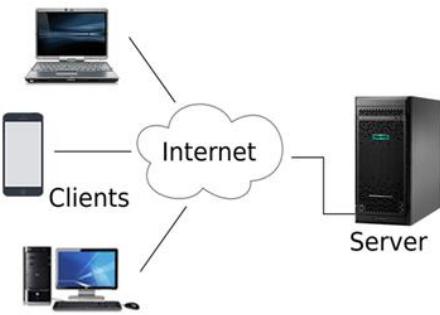
- It is less costly as it does not contain any dedicated server.
- If one computer stops working but, other computers will not stop working.

#### **Disadvantages of Peer-To-Peer Network:**

- In the case of Peer-To-Peer network, it does not contain the centralized system . Therefore, it cannot back up the data as the data is different in different locations.
- It has a security issue as the device is managed itself.

#### **Client/Server Network:**

- Client/Server network is a network model designed for the end users called clients, to access the resources such as songs, video, etc. from a central computer known as Server.
- The central controller is known as a **server** while all other computers in the network are called **clients**.
- A server performs all the major operations such as security and network management.
- A server is responsible for managing all the resources such as files, directories, printer, etc.
- All the clients communicate with each other through a server. For example, if client1 wants to send some data to client 2, then it first sends the request to the server for the permission. The server sends the response to the client 1 to initiate its communication with the client 2.



#### **Advantages of Client/Server network:**

- A Client/Server network contains the centralized system. Therefore we can back up the data easily.
- A Client/Server network has a dedicated server that improves the overall performance of the whole system.

#### **Disadvantages of Client/Server network:**

- Client/Server network is expensive as it requires the server with large memory.
- A server has a Network Operating System(NOS) to provide the resources to the clients, but the cost of NOS is very high.

**Types of Computer Network-** There are computer networks that help in building social relationships. These networks depend upon the speed at which the internet is provided. All these three networks, i.e., LAN, MAN, and WAN, are used for providing internet services to people.

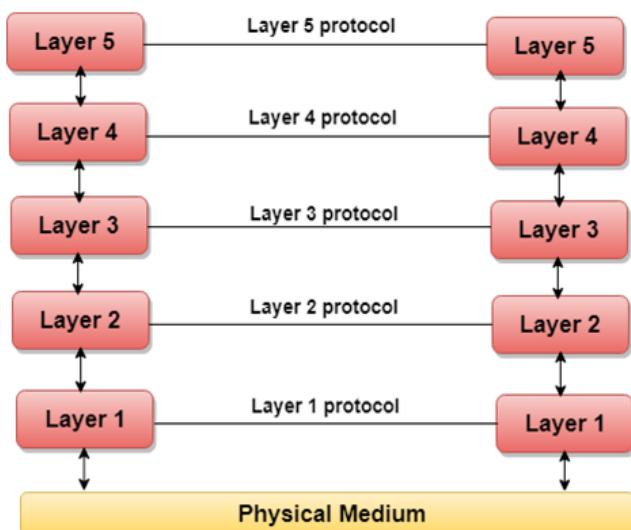
Let us discuss about them in detail:

Basis	LAN	MAN	WAN
<b>Full-Form</b>	<a href="#">LAN stands for local area network.</a>	<a href="#">MAN stands for metropolitan area network.</a>	<a href="#">WAN stands for wide area network.</a>
<b>Geographic Span</b>	Operates in small areas such as the same building or campus.	Operates in large areas such as a city.	Operates in larger areas such as country or continent.
<b>Ownership</b>	LAN's ownership is private.	MAN's ownership can be private or public.	While WAN also might not be owned by one organization.
<b>Transmission Speed</b>	The transmission speed of a LAN is high.	While the transmission speed of a MAN is average.	Whereas the transmission speed of a WAN is low.
<b>Propagation delay</b>	The propagation delay is short in a LAN.	There is a moderate propagation delay in a MAN.	Whereas, there is a long propagation delay in a WAN.
<b>Congestion</b>	There is less congestion in LAN.	While there is more congestion in MAN.	Whereas there is more congestion than MAN in WAN.

<b>Design &amp; Maintenance</b>	LAN's design and maintenance are easy.	While MAN's design and maintenance are difficult than LAN.	Whereas WAN's design and maintenance are also difficult than LAN as well as MAN.
<b>Fault tolerance</b>	There is more fault tolerance in LAN.	While there is less fault tolerance.	In WAN, there is also less fault tolerance.

### Layered Architecture-

- The main aim of the layered architecture is to divide the design into small pieces.
- Each lower layer adds its services to the higher layer to provide a full set of services to manage communications and run the applications.
- It provides modularity and clear interfaces, i.e., provides interaction between subsystems.
- The basic elements of layered architecture are services, protocols, and interfaces.
- **Service:** It is a set of actions that a layer provides to the higher layer.
- **Protocol:** It defines a set of rules that a layer uses to exchange the information with peer entity. These rules mainly concern about both the contents and order of the messages used.
- **Interface:** It is a way through which the message is transferred from one layer to another layer.



- layer 1 is the physical medium through which the actual communication takes place.
- In a layered architecture, unmanageable tasks are divided into several small and manageable tasks.
- A set of layers and protocols is known as network architecture.

### CONNECTION ORIENTED AND CONNECTIONLESS SERVICES

#### Connection Oriented Service:

It is a three-phase process which include-

- Connection Establishment
- Data Transfer
- Termination / disconnection

In this type of transmission, the receiving device sends an acknowledgement, back to the source after a packet or group of packet is received. This type of transmission is reliable and secure.

**Connection less service:** It is a one-phase process and includes Data Transfer. In this type of transmission, the receiver does not acknowledge receipt of a packet. This approach allows for much faster communication between devices. Connection-oriented service is more reliable than connectionless Service.

Criteria	Connection-Oriented	Connection-Less
<b>Connection</b>	Prior connection needs to be established.	No prior connection is established.
<b>Resource Allocation</b>	Resources need to be allocated.	No prior allocation of resource is required.
<b>Reliability</b>	It ensures reliable transfer of data.	Reliability is not guaranteed as it is a best effort service.
<b>Congestion</b>	Congestion is not at all possible.	Congestion can occur likely.
<b>Transfer mode</b>	It can be implemented either using Circuit Switching or VCs.	It is implemented using Packet Switching.
<b>Retransmission</b>	It is possible to retransmit the lost data bits.	It is not possible.
<b>Suitability</b>	It is suitable for long and steady communication.	It is suitable for bursty transmissions.
<b>Signaling</b>	Connection is established through process of signaling.	There is no concept of signaling.
<b>Packet travel</b>	In this packets travel to their destination node in a sequential manner.	In this packets reach the destination in a random manner.
<b>Delay</b>	There is more delay in transfer of information, but once connection established faster delivery.	There is no delay due absence of connection establishment phase.

## SERVICE PRIMITIVES

Service generally includes set of various primitives. A primitive simply means Operations.

A Service is specified by set of primitives that are available and given to user or other various entities to access the service. All these primitives simply tell the service to perform some action or to report on action that is taken by peer entity. Each of the protocol that communicates in layered architecture also communicates in peer-to-peer manner with some of its remote protocol entity.

Primitives are called calling functions between the layers that are used to manage communication among the adjacent protocol layers i.e., among the same communication node. The set of primitives that are available generally depends upon the nature of the service that is being provided.

### Classification of Service Primitives:

Primitive	Meaning
<b>Request</b>	It represents entity that wants or request service to perform some action or do some work (requesting for connection to remote computer).
<b>Indication</b>	It represent entity that is to be informed about event (receiver just have received request of connection).
<b>Response</b>	It represents entity that is responding to event (receiver is simply sending the permission or allowing to connect).
<b>Confirm</b>	It represent entity that acknowledges the response to earlier request that has come back (sender just acknowledge the permission to get connected to the remote host).

### Design issues-

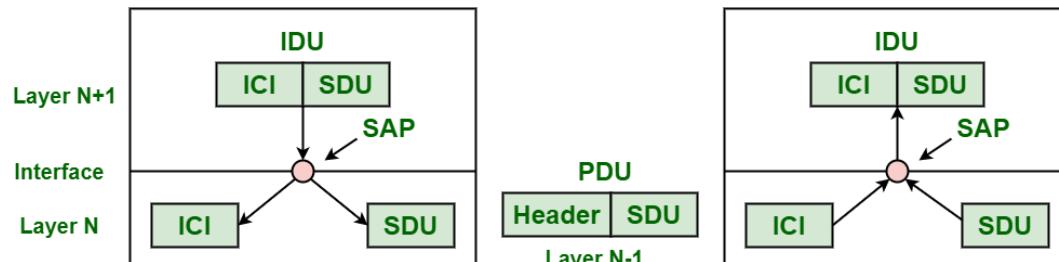
- **Topology:** Choose an appropriate network topology.
- **Size and Scale:** Determine the network's size and scale.
- **Scalability:** Design for future growth.
- **Performance:** Address bandwidth and latency issues.
- **Reliability:** Ensure continuous operation with redundancy.
- **Security:** Implement measures to protect against threats.
- **Cost:** Balance performance with budget constraints.
- **Management:** Design for easy configuration and troubleshooting.
- **Interoperability:** Ensure compatibility with different devices.
- **QoS:** Address specific application requirements for quality of service.
- **Addressing and Naming:** Develop strategies for IP addressing and domain naming.
- **Documentation:** Maintain comprehensive documentation.
- **Environmental Considerations:** Account for physical factors like temperature and power.
- **User Requirements:** Align the design with end-user and application needs.

### INTERFACES AND SERVICES

Interfaces and Services is a process that generally provides and gives a common technique for each layer to communicate with each other. Standard terminology basically required for layered networks to request and aim for the services are provided.

Service is defined as a set of primitive operations. Services are provided by layer to each of layers above it.

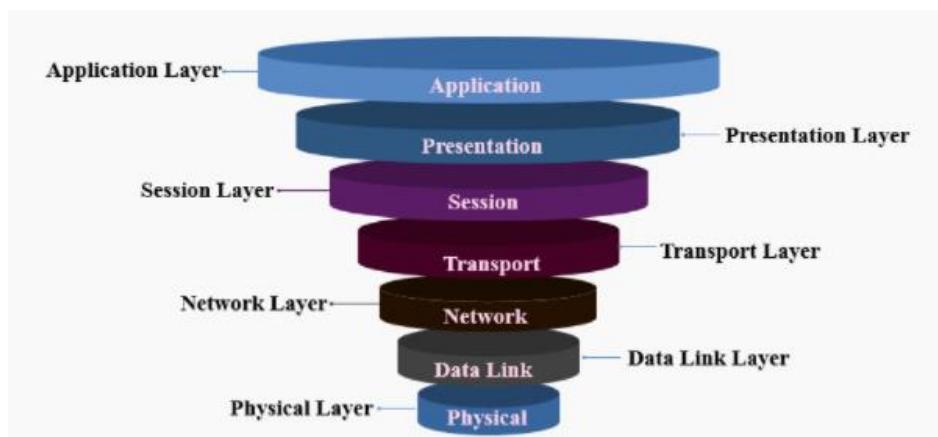
Below is diagram showing relation between layers at an interface. In diagram, layers N+1, N, and N-1 are involved and engaged in process of communication among each other.



**Relationship Between Layers at an Interface**

## ISO-OSI REFERENCE MODEL

OSI stands for Open Systems Interconnection. It has been developed by ISO – ‘International Organization of Standardization’, in the year 1984. It is a 7 layer architecture with each layer having specific functionality to perform.



**Figure 1.1: OSI reference model**

### Layer 1: Physical layer

The physical layer has the following major functions:

- It defines the electrical and physical specifications of the data connection. It defines the relationship between a device and a physical transmission medium (e.g., a copper or fiber optical cable). This includes the layout of pins, voltages, line impedance, cable specifications, signal timing, hubs, repeaters, network adapters, host bus adapters (HBA used in storage area networks) and more.
- It defines the protocol to establish and terminate a connection between two directly connected nodes over a communications medium.
- It may define the protocol for flow control.
- It defines transmission mode i.e. simplex, half duplex, full duplex.
- It defines the topology.

- It defines a protocol for the provision of a (not necessarily reliable) connection between two directly connected nodes, and the modulation or conversion between the representation of digital data in user Equipment and the corresponding signals transmitted over the physical communications channel.
- Cabling system components
- Adapters that connect media to physical interfaces
- Connector design and pin assignments
  - Hub, repeater, and patch panel specifications
  - Wireless system components
  - Parallel SCSI (Small Computer System Interface)
  - Network Interface Card (NIC)

## **Layer 2: Data link layer**

The data link layer provides node-to-node data transfer - A reliable link between two directly connected nodes, by detecting and possibly correcting errors that may occur in the physical layer. The data link layer is divided into two sublayers:

- Media Access Control (MAC) layer - Responsible for controlling how devices in a network gain access to data and permission to transmit it.
- Logical Link Control (LLC) layer - Controls error checking and packet synchronization.

## **Basic Functions**

- Allows a device to access the network to send and receive messages
- Offers a physical address so a device's data can be sent on the network
- Works with a device's networking software when sending and receiving messages
- Provides error-detection capability

## **Common networking components that function at layer 2 include:**

- Network interface cards
- Ethernet and Token Ring switches
- Bridges

## **Layer 3: Network layer**

The network layer provides the functional and procedural means of transferring variable length data sequences (called datagrams) from one node to another connected to the same network.

- It translates logical network address into physical machine address.

- Routing is also one of the main functions of the Network Layer, routing is the process of selecting paths in a network over which to send packets.
- Internet Control Message Protocol (ICMP) is network layer protocol and one of the main protocols of the Internet Protocol suite and is used for error handling and diagnostic purposes. Quality of Service (QoS) although not the primary function of the network layer is available in network layer protocols such as the Internet Protocol which allows certain traffic to be prioritized over other giving it preferential treatment.

#### **Layer 4: Transport layer**

The transport layer provides the functional and procedural means of transferring variable-length data sequences from a source to a destination host via one or more networks while maintaining the quality of service functions.

An example of a transport-layer protocol in the standard Internet stack is Transmission Control Protocol (TCP), usually built on top of the Internet Protocol (IP).

Some of the functions offered by the transport layer include:

- Application identification
- Client-side entity identification
- Confirmation that the entire message arrived intact
- Segmentation of data for network transport
- Control of data flow to prevent memory overruns
- Establishment and maintenance of both ends of virtual circuits
- Transmission-error detection
- Realignment of segmented data in the correct order on the receiving side
- Multiplexing or sharing of multiple sessions over a single physical link

#### **Layer 5: Session layer**

The session layer controls the dialogues (connections) between computers. It establishes, manages and terminates the connections between the local and remote application.

It provides for full-duplex, half-duplex, or simplex operation, and establishes check pointing, adjournment, termination, and restart procedures. This session layer allows applications functioning on devices to establish, manage, and terminate a dialog through a network. Session layer functionality includes:

- Virtual connection between application entities
- Synchronization of data flow
- Creation of dialog units

- Connection parameter negotiations
- Partitioning of services into functional groups
- Acknowledgements of data received during a session
- Retransmission of data if it is not received by a device

### **Layer 6: Presentation layer**

The presentation layer, is responsible for how an application formats the data to be sent out onto the network. The presentation layer basically allows an application to read (or understand) the message.

Examples of presentation layer functionality include:

- Encryption and decryption of a message for security
- Compression and expansion of a message so that it travels efficiently
- Graphics formatting
- Content translation
- System-specific translation

### **Layer 7: Application layer**

The application layer, provides an interface for the end user operating a device connected to a network.

This layer is what the user sees, in terms of loading an application (such as Web browser or e-mail).

Examples of application layer functionality include:

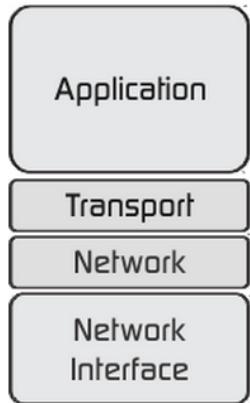
- Support for file transfers
- Ability to print on a network
- Electronic mail
- Electronic messaging
- Browsing the World Wide Web

### **TCP/IP MODEL**

The TCP/IP reference model is the network model used in the current Internet architecture. It is considered as the grandfather of the Internet the ARPANET. The reference model was named after two of its main protocols, TCP (Transmission control Protocol) and IP (Internet Protocol).

There are versions of this model with four layers and with five layers. The original four-layer version of the model is shown below.

## TCP/IP model



**Layer 4:** Process Layer or Application Layer: This is where the “higher level” protocols such as FTP, HTTP, etc. Operate. The original TCP/IP specification described many different applications that fit into the top layer of the protocol stack. These applications include Telnet, FTP, SMTP, and DNS.

**Layer 3:** Host-To-Host (Transport) Layer: This is where flow-control and connection protocols exist, such as TCP. This layer deals with opening and maintaining a connection, ensuring that packet is in fact received. The transport layer is the interface between the application layer and the complex hardware of the.

Two modes are available, full-duplex and half-duplex. In full-duplex operation, both sides can transmit and receive data simultaneously, whereas, in half duplex, a side can only send or receive at one time.

**Layer 2:** Internet or Internetworking Layer: This layer defines IP addresses, with many routing schemes for navigating packets from one IP address to another. The job of the network layer is to inject packets into any network and have them travel independently to the destination. Packet routing is a major job of this protocol.

**Layer 1:** Networking Access Layer: This layer describes the physical equipment necessary for communications, such as twisted pair cables, the signaling used on that equipment, and the low-level protocols using that signaling. That Host-to-Network layer interfaces the TCP/IP protocol stack to the physical network.

### Principal of physical layer-

The Physical Layer is the first layer of the OSI (Open Systems Interconnection) model and is primarily concerned with the physical transmission of data bits over a physical medium.

It deals with the hardware aspects of transmitting raw binary data over a physical link or medium, without concern for the higher-layer protocols.

The physical layer converts the data frame received from the data link layer into bits, i.e., in terms of ones and zeros. It maintains the data quality by implementing the required protocols on different network modes and maintaining the bit rate through data transfer using a wired or wireless medium.

## **Attributes of the Physical Layer-**

**1. Signals:** The data is first converted to a signal for efficient data transmission. There are two kinds of signals:

- **Analog Signals:** These signals are continuous waveforms in nature and are represented by continuous electromagnetic waves for the transmission of data.
- **Digital Signals:** These signals are discrete in nature and represent network pulses and digital data from the upper layers.

**2. Transmission media:** Data is carried from source to destination with the help of transmission media. There are two sorts of transmission media:

- **Wired Media:** The connection is established with the help of cables. For example, fiber optic cables, coaxial cables, and twisted pair cables.
- **Wireless Media:** The connection is established using a wireless communication network. For example, Wi-Fi, Bluetooth, etc.

**Data Flow:** It describes the rate of data flow and the transmission time frame. The factors affecting the data flow are as follows:

- **Encoding:** Encoding data for transmission on the channel.
- **Error-Rate:** Receiving erroneous data due to noise in transmission.
- **Bandwidth:** The rate of transmission of data in the channel.

## **3.Modulation:**

- Amplitude Modulation (AM): Varies the amplitude of the carrier signal to represent data.
- Frequency Modulation (FM): Varies the frequency of the carrier signal.
- Phase Modulation (PM): Varies the phase of the carrier signal.

## **4. Transmission mode:**

It describes the direction of the data flow. Data can be transmitted in three sorts of transmission modes as follows:

- **Simplex mode:** This mode of communication is a one-way communication where a device can only send data. Examples are a mouse, keyboard, etc.
- **Half-duplex mode:** This mode of communication supports one-way communication, i.e., either data can be transmitted or received. An example is a walkie-talkie.
- **Full-duplex mode:** This mode of communication supports two-way communication, i.e., the device can send and receive data at the same time. An example is cellular communication.

## Subject Notes: Unit I

### AD 501- Theory of Computation

#### B.Tech, AI&DS-5<sup>th</sup> Semester

**Syllabus: Introduction of Automata Theory:** Examples of automata machines, Finite Automata as a language acceptor and translator, Moore machines and mealy machines, composite machine, Conversion from Mealy to Moore and vice versa.

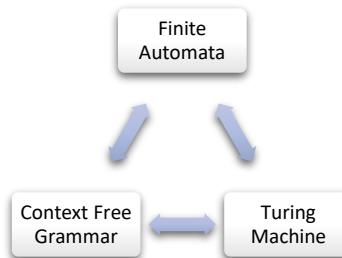
#### Unit-I: Introduction of Automata Theory

##### Automata

The term "Automata" is derived from the Greek word "αὐτόματα" which means "self-acting". An automaton (Automata in plural) is an abstract self-propelled computing device which follows a predetermined sequence of operations automatically.

Automata are computational devices to solve language recognition problems. Language recognition problem is to determine whether a word belongs to a language.

Automaton = abstract computing device, or “machine”.



**Figure 1.1: Computational Model of Automata Theory**

##### Examples of automata machines:

- **Sequential machine:** A sequential machine is a mathematical model of a certain type of simple computational structure. Its behavior represents the working process of finite Automata.
- **Vending Machines:** A vending machine is an automated machine that dispenses numerous items such as cold drinks, snacks, beverages, alcohol etc. Vending machine is works on finite state automate to control the functions process.
- **Traffic Lights:** The optimization of traffic light controllers in a city is a systematic representation of handling the instructions of traffic rules. Its process depends on a set of instruction works in a loop with switching among instruction to control traffic.
- **Video Games:** Video games levels represent the states of automata. In which a sequence of instructions is followed by the players to accomplish the task.
- **Text Parsing:** Text parsing is a technique which is used to derive a text string using the production rules of a grammar to check the acceptability of a string.
- **Regular Expression Matching:** It is a technique to checking the two or more regular expression are similar to each other or not. The finite state machine is useful to checking out that the expressions are acceptable or not by a machine or not.

- **Speech Recognition:** Speech recognition via machine is the technology enhancement that is capable to identify words and phrases in spoken language and convert them to a machine-readable format. Receiving words and phrases from real world and then converted it into machine readable language automatically is effectively solved by using finite state machine.

#### **Characteristics:**

- FA is having only a finite number of states.
- Finite Automata can only "count" (that is, maintain a counter, where different states correspond to different values of the counter) a finite number of input scenarios.
- Able to solve limited set of problem
- Outcome in the form of "yes" or "No" (Accept / Reject)

#### **Limitation:**

There is no finite automaton that recognizes these strings:

- The set of binary strings consisting of an equal number of 1's and 0's
- The set of strings over '(' and ')' that have "balanced" parentheses.

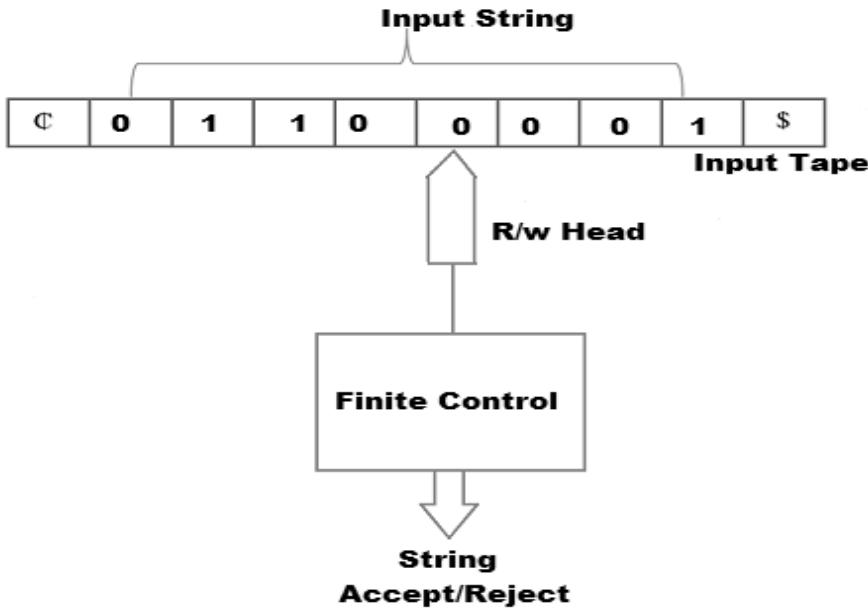
#### **Applications of TOC:**

- Finite State Programming
- Event Driven Finite State Machine (FSM)
- Virtual FSM
- DFA based text filter in Java
- Acceptors and Recognizers
- Transducers
- UML state diagrams
- Hardware Application

#### **Finite Automata:**

An automaton with a finite number of states is called a Finite Automaton (FA) or Finite State Machine (FSM).

An automaton has a mechanism to read input, which is string over a given alphabet. This input is actually written on an input tape /file, which can be read by automaton but cannot change it. Input file is divided into cells each of which can hold one symbol. Automaton has a control unit which is said to be in one of finite number of internal states.



**Figure 1.2: Finite Automata**

**Formal definition of Finite Automata:**

A finite automaton is a 5-tuple  $M = (Q, \Sigma, \delta, q, F)$ , where

1.  $Q$  is a finite set, whose elements are called states,
2.  $\Sigma$  is a finite set, called the alphabet; the elements of  $\Sigma$  are called symbols,
3.  $\delta: Q \times \Sigma \rightarrow Q$  is a function, called the transition function,
4.  $q$  is an element of  $Q$ ; it is called the start state or initial state,
  
5.  $F$  is a subset of  $Q$ ; the elements of  $F$  are called accept states or final state.

**Related Terminologies:**

**Alphabet:** An alphabet is any finite set of symbols.

Example:  $\Sigma = \{a, b, c, d\}$  is an alphabet set where 'a', 'b', 'c', and 'd' are symbols.

**String:** A string is a finite sequence of symbols taken from  $\Sigma$ .

Example: 'cabcad' is a valid string on the alphabet set  $\Sigma = \{a, b, c, d\}$

**Length of a String:** It is the number of symbols present in a string. (Denoted by  $|S|$ ).

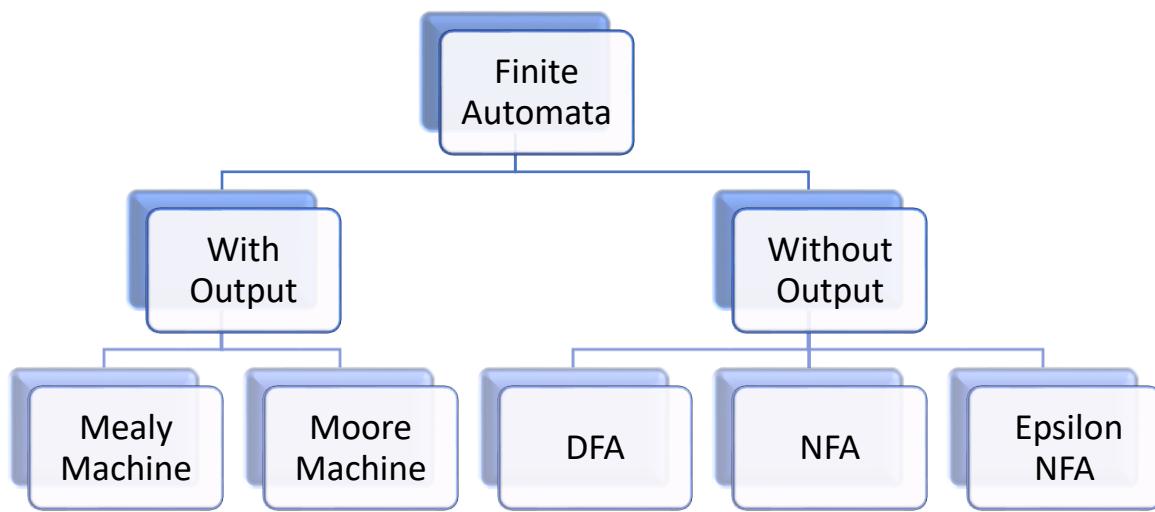
Examples: If  $S = \text{'cabcad'}$ ,  $|S| = 6$

If  $|S| = 0$ , it is called an empty string (Denoted by  $\lambda$  or  $\epsilon$ )

**Language:** A language is a subset of  $\Sigma^*$  for some alphabet  $\Sigma$ . It can be finite or infinite.

Example: If the language takes all possible strings of length 2 over  $\Sigma = \{a, b\}$ , then  $L = \{ab, bb, ba, aa\}$

**Classification of Finite Automata:**



**Figure 1.3: Classification of Finite Automata**

#### An example of finite automata

Let  $A = \{w : w \text{ is a binary string containing an odd number of } 1\text{'s}\}$ .

We claim that this language  $A$  is regular. In order to prove this, we have to construct a finite automaton  $M$  such that  $A = L(M)$ .

Steps to construct finite automata:

- The FA reads the input string  $w$  from left to right and keep scanning on the number of 1's
- After scanning the entire string, it counts the number of 1's to check whether it is odd or even
- If the number of 1's is odd then the string is acceptable otherwise it is rejected

Using this approach, the finite automaton needs a state for every integer  $i \geq 0$ ; indicating that the number of 1s read so far is equal to  $i$ . Hence, to design a finite automaton that follows this approach, we need an infinite number of states. But, the definition of finite automaton requires the number of states to be finite. A better, and correct approach, is to keep track of whether the number of 1s read so far is even or odd. This leads to the following finite automaton:

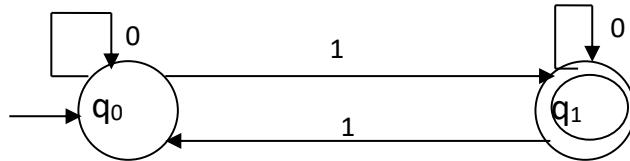
- The set of states is  $Q = \{q_0, q_1\}$ . If the finite automaton is in state  $q_1$ , then it has an even number of 1's; if it is in state  $q_0$ , then it has an odd number of 1's.
- The alphabet is  $\Sigma = \{0, 1\}$ .
- The start state is  $q_0$ , because at the start, the number of 1's read by the automaton is equal to 0, and 0 is even.
- The set  $F$  of accept states is  $F = \{q_1\}$ .
- The **transition function  $\delta$**  is given by the following table:

State	Input 0	Input 1
-------	---------	---------

$q_0$	$q_0$	$q_1$
$q_1$	$q_1$	$q_0$

**Table 1.1: Transition Table/Matrix**

This finite automaton  $M = (Q, \Sigma, \delta, q_0, F)$  can also be represented by its state diagram.



**Figure 1.4: Transition Graph**

**Transition Graph:** it is a finite directed labeled graph in which each vertex (or node) represents a state and the directed edges indicate the transition of state. Edges are labeled with input symbol.

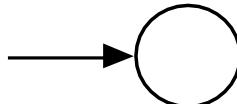
**Transition Matrix:** It is two-dimension matrixes between states of automata and Input symbol. Elements of matrix are state form mapping ( $\Sigma \times Q$ ) into  $Q$ .

**Finite state acceptor** is a finite state machine with no outputs. The user of a finite state acceptor cares only about the final state: if the machine ends in an **accepting** state after processing a series of inputs, the machine is said to have **accepted** the input; otherwise, it is said to have **rejected** the input.

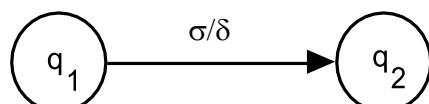
### Description of Finite-State Machines using Graphs

Any finite-state machine can be shown as a graph with a finite set of nodes. The nodes correspond to the states. There is no other memory implied other than the state shown. The start state is designated with an arrow directed into the corresponding node, but otherwise unconnected.

**Figure 1.5: An unconnected in-going arc indicates that the node is the start state**

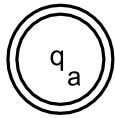


The arcs and nodes are labeled differently, depending on whether we are representing a transducer, a classifier, or an acceptor. In the case of a **transducer**, the arcs are labeled as shown below, where  $q_1$  is the input symbol and  $q_2$  is the output symbol. The state- transition is designated by virtue of the arrow going from one node to another.



**Figure 1.6: Transducer transition from  $q_1$  to  $q_2$ , based on input  $\sigma$ , giving output  $\delta$**

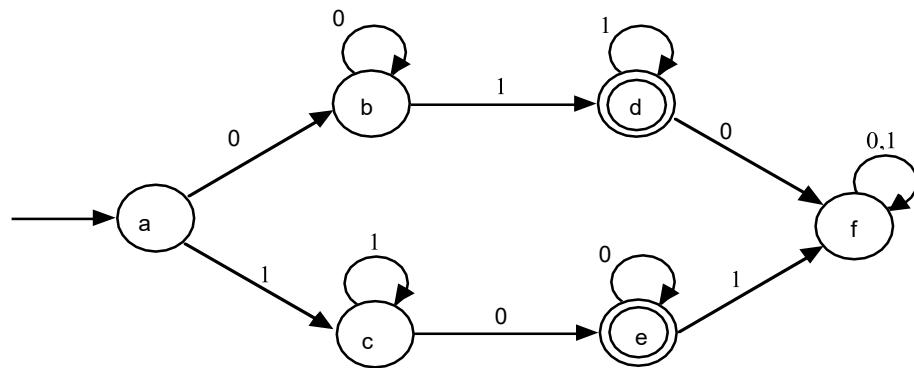
In the case of an **acceptor**, instead of labeling the states with categories 0 and 1, we sometimes use a double-lined node for an accepting state and a single-lined node for a rejecting state.



**Figure 1.7: Acceptor, an accepting state**

#### Acceptor Example

Let us give an acceptor that accepts those strings with exactly one edge. We can use the state transitions from the previous classifier. We need only designate those states that categorize there being one edge as accepting states and the others as rejecting states.



**Figure 1.8: Acceptor for strings with exactly one edge. Accepting states are d and e**

#### Finite automata as Acceptor (Recognizer)

An automaton that computes a Boolean function is called an acceptor. All the states of an acceptor is either accepting or rejecting the inputs given to it.

#### Finite automata as Classifier

A classifier has more than two final states and it gives a single output when it terminates.

## Finite automata as Classifier Translator (Transducer)

An automaton that produces outputs based on current input and/or previous state is called a transducer or translator.

Transducers can be of two types:

- Mealy Machine: The output depends both on the current state and the current input.
- Moore Machine: The output depends only on the current state.

## Examples for Practice

**Problem-01:** Draw a DFA for the language accepting strings starting with 'ab' over input alphabets  $\Sigma = \{a, b\}$

**Solution-**Regular expression for the given language =  $ab (a + b)^*$

Step-01: All strings of the language start with substring "ab".

So, length of substring = 2.

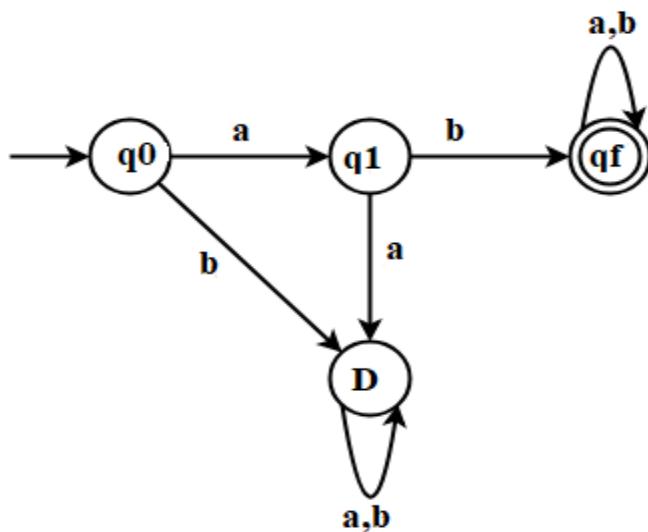
Thus, Minimum number of states required in the DFA =  $2 + 2 = 4$ .

It suggests that minimized DFA will have 4 states

Step-02: We will construct DFA for the following strings-

Ab, aba, abab

Step-03: The required DFA is-



**Figure 1.9: Transition Graph of required DFA**

**Problem-02:** Draw a DFA for the language accepting strings starting with 'a' over input alphabets  $\Sigma = \{a, b\}$

**Solution-**Regular expression for the given language =  $a(a + b)^*$

Step-01: All strings of the language starts with substring "a".

So, length of substring = 1.

Thus, Minimum number of states required in the DFA =  $1 + 2 = 3$ .

It suggests that minimized DFA will have 3 states.

Step-02: We will construct DFA for the following strings-

A,aa

Step-03: The required DFA is-

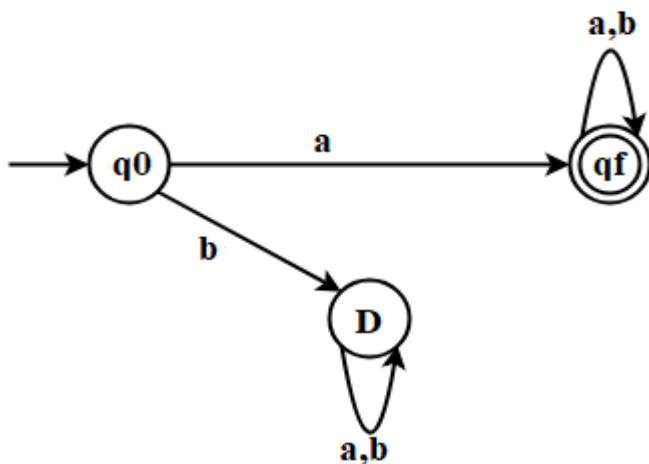


Figure 1.10: Transition Graph of required DFA

**Problem 3:** Construct a minimal DFA, which accepts set of all strings over {0, 1}, which when interpreted as binary number is divisible by '3'. Means 110 in binary is equivalent to 6 in decimal and 6 is divisible by 3.

**Solution-**So if you think in the way of considering remainders if you divide by 3 that is {0, 1, 2}

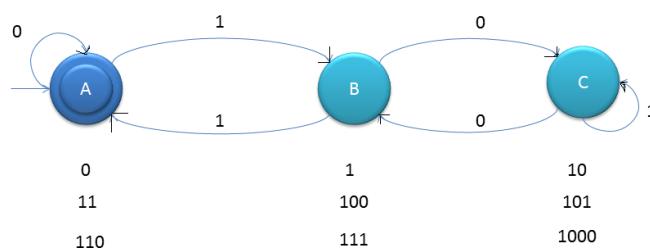


Figure 1.11: Transition Graph

As you can see that binary number which is divisible by 2 is appearing on left state. Short Trick to create such DFAs

Create Transition Table as below

Table creation rules

State	0	1
-------	---	---

<b>q0</b>	q0	q1
<b>q1</b>	q2	q0
<b>q2</b>	q1	q2

**Table 1.2: Transition Table for required DFA**

- First write the input alphabets, example 0, 1
- There will n states for given number which is divisor.
- Start writing states, as for n=3: q0 under 0, q1 under 1
- Q2 under 0 and q0 under 1
- Q1 under 0 and q2 under 1

If the input alphabets are 0, 1, 2 then table will expand accordingly with same rules above.

Example

For ternary pattern and n=4, table will be as follows:

State	0	1	2
<b>q0</b>	q0	q1	q2
<b>q1</b>	q3	q0	q1
<b>q2</b>	q2	q3	q0
<b>q3</b>	q1	q2	q3

**Table 1.3: Transition Table for above DFA**

#### **Mealy and Moore Machine:**

These machines are modeled to show transition and output symbol. These machines do not define a language by accepting or rejecting input string, so there is no existence of final state. Finite automata generate outputs related to each act. While two types of finite state machines create output –

- **Mealy Machine**
- **Moore machine**

#### **Mealy Machine:**

A Mealy Machine is considered as an FSM, the output will be based on the present state and the present input. **Mealy machine is explained as a 6 tuple ( $Q, \Sigma, O, \delta, X, q_0$ ) where –**

- $Q$  is a finite set of states.
- $\Sigma$  is a finite set of symbols called the input alphabet.
- $O$  is a finite set of symbols called the output alphabet.
- $\delta$  is the input transition function where  $\delta: Q \times \Sigma \rightarrow O$

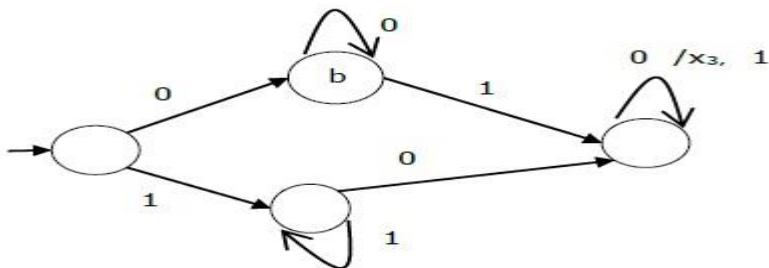
- $X$  is the output transition function where  $X: Q \times \Sigma \rightarrow O$
- $q_0$  is the initial state from where any input is processed ( $q_0 \in Q$ ).

The state table of a Mealy Machine is mentioned below –

Present state	Next state			
	input = 0		input = 1	
	State	Output	State	Output
$\rightarrow a$	B	x1	c	x1
b	B	x2	d	x3
c	D	x3	c	x1
d	D	x3	d	x2

**Table 1.4: Transition Table for Mealy Machine**

The state diagram of the above Mealy Machine is –



**Figure 1.12: Transition Graph**

### Moore Machine

Moore machine is also considered as an FSM and the outputs depend on the present state.

A Moore machine is also explained by a 6 tuple  $(Q, \Sigma, O, \delta, X, q_0)$  where –

- $Q$  is a finite set of states.
- $\Sigma$  is a finite set of symbols called the input alphabet.
- $O$  is a finite set of symbols called the output alphabet.
- $\delta$  is the input transition function where  $\delta: Q \times \Sigma \rightarrow Q$
- $X$  is the output transition function where  $X: Q \rightarrow O$
- $q_0$  is the initial state from where any input is processed ( $q_0 \in Q$ ).

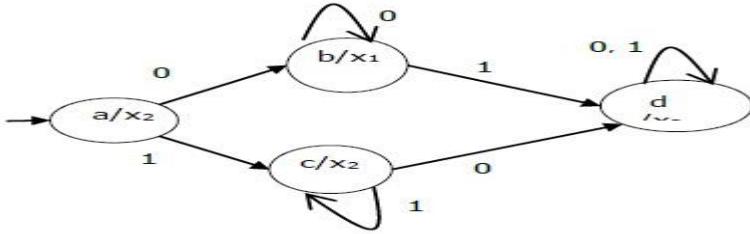
The state table of a Moore Machine is shown below –

Present state	Next State		Output
	Input = 0	Input = 1	

$\rightarrow a$	B	c	x2
b	B	d	x1
c	C	d	x2
d	D	d	x3

**Table 1.5: Transition Table for Moore Machine**

The state diagram of the above Moore Machine is –



**Figure 1.13: Transition Graph**

#### Composite finite-state machines:

A finite state machine, FSM, is a box with C input/output channels, and S states, and a fixed map  $f:S \times C \rightarrow S \times C$ . If a state  $(ci, sj)$  is mapped to the 0 element it means it enters a loop and can't exit. if we join the channels of several FSM's pairwise, we obtain a new FSM, i.e. we get a system with some number of unjoined channels, C, and internal states given by the product of the number of internal states of each component FSM, S. A composite FSM is allowed to have internal loops, i.e. we may never exit through an unjoined channel.

#### Conversion from Moore Machine to Mealy Machine

##### Algorithm

**Input** – Moore Machine

**Output** – Mealy Machine

**Step 1** – Take a blank Mealy Machine transition table format.

**Step 2** – Copy all the Moore Machine transition states into this table format.

**Step 3** – Now check the present states and their corresponding outputs in the Moore Machine state table; if for a state  $Qi$  output is m, copy it into the output columns of the Mealy Machine state table wherever  $Qi$  appears in the next state.

##### Example 1

Let's see the following Moore machine –

Present State	Next State		Output
	$a = 0$	$a = 1$	
$\rightarrow a$	D	b	1

<b>B</b>	A	d	0
<b>C</b>	C	c	0
<b>D</b>	B	a	1

**Table 1.6: Transition Table for given Moore Machine**

Now we apply Algorithm to convert it to Mealy Machine.

**Step 1 & 2 –**

Present State	Next State			
	a = 0		a = 1	
	State	Output	State	Output
→ a	D		B	
B	A		D	
C	C		C	
D	B		A	

**Table 1.7: Intermediate Table**

**Step 3 –**

Present State	Next State			
	a = 0		a = 1	
	State	Output	State	Output
=> a	D	1	b	0
b	A	1	d	1
c	C	0	c	0
d	B	0	a	1

**Table 1.8: Required Mealy Machine**

## Example 2

Let's see the following Moore machine –

	Next State	Output
--	------------	--------

Present State	a = 0	a = 1	
→ q0	q1	q0	0
q1	q1	q2	0
q2	q1	q0	1

Table 1.9: Transition Table for given Moore Machine

Now we apply Algorithm to convert it to Mealy Machine.

**Step 1 & 2 –**

Present State	Next State			
	a = 0		a = 1	
	State	Output	State	Output
→ q0	q1		q0	
q1	q1		q2	
q2	q1		q0	

Table 1.10: Intermediate Table

**Step 3 –**

Present State	Next State			
	a = 0		a = 1	
	State	Output	State	Output
→ q0	q1	0	q0	0
q1	q1	0	q2	1
q2	q1	0	q0	0

Table 1.11: Required Mealy Machine

### Conversion from Mealy Machine to Moore Machine

Algorithm

**Input – Mealy Machine**

**Output – Moore Machine**

**Step 1 –** Here measure the number of different outputs for each state ( $Q_i$ ) that are available in the state table of the Mealy machine.

Step 2 – Incase if all the outputs of  $Q_i$  are same, copy state  $Q_i$ . If it has  $n$  distinct outputs, break  $Q_i$  into  $n$  states as  $Q_{in}$  where  $n = 0, 1, 2, \dots$

**Step 3 –** If the output of the initial state is 1, insert a new initial state at the beginning which gives 0 output.

### Example 1

Let's see the following Mealy Machine –

Present State	Next State			
	a = 0		a = 1	
	Next State	Output	Next State	Output
→ a	D	0	b	1
b	A	1	d	0
c	C	1	c	0
d	B	0	a	1

Table 1.12: Given Mealy Machine

While the states 'a' and 'd' provide only 1 and 0 outputs respectively, so we create states 'a' and 'd'. But states 'b' and 'c' deliver different outputs (1 and 0). So, we divide b into b0, b1 and c into c0, c1.

Present State	Next State		Output
	a = 0	a = 1	
→ a	D	b1	1
b0	A	d	0
b1	A	d	1
c0	c1	c0	0
c1	c1	c0	1
D	b0	a	0

Table 1.13: Required Moore Machine

### Example 2

Consider the following Mealy Machine

Present State	Next State			
	a = 0		a = 1	
	Next State	Output	Next State	Output
→ q1	q1	1	q2	0

<b>q2</b>	<b>q4</b>	1	<b>q4</b>	1
<b>q3</b>	<b>q2</b>	1	<b>q3</b>	1
<b>q4</b>	<b>q3</b>	0	<b>q1</b>	1

**Table 1.14: Given Mealy Machine**

- For state q1, there is only one incident edge with output 0. So, we don't need to split this state in Moore machine.
- For state q2, there is 2 incident edge with output 0 and 1. So, we will split this state into two states q20( state with output 0) and q21(with output 1).
- For state q3, there is 2 incident edge with output 0 and 1. So, we will split this state into two states q30( state with output 0) and q31( state with output 1).
- For state q4, there is only one incident edge with output 0. So, we don't need to split this state in Moore machine.

Transition table for Moore machine will be:

Present State	Next State		Output
	a = 0	a = 1	
→ q1	q1	q20	1
q20	q4	q4	0
q21	q4	q4	1
q30	q21	q31	0
q31	q21	q31	1
q4	q30	q1	1

**Table 1.15: Required Moore Machine**

**Difference between Mealy and Moore Machine:**

Mealy Machine	Moore Machine
Output depends both upon present state and present input	Output depends only upon the present state
Generally, it has fewer states than Moore machine	Generally, it has more states than Mealy machine
Output changes at clock edges	Input change can cause change in output as soon as logic is done.
Mealy machine react faster to inputs.	In Moore machines, more logic is needed to decode the outputs since it has more circuit delays.

**Table 1.16: Difference between Mealy & Moore Machine**

## **Unit II**

**Syllabus: Types of Finite Automata:** Non-Deterministic Finite Automata (NDFA), Deterministic finite automata machines, conversion of NDFA to DFA, minimization of automata machines, regular expression, Arden's theorem, Meaning of union, intersection, concatenation and closure, 2-wayDFA.

**Unit Objective:** Relate practical problems to languages, automata, computability and complexity. Constructs abstract models of computing and check their power to recognize the language.

**Finite Automata:** An automaton with a finite number of states is called a Finite Automaton (FA) or Finite State Machine (FSM).

An automaton has a mechanism to read input, which is string over a given alphabet. This input is actually written on an input tape /file, which can be read by automaton but cannot change it. Input file is divided into cells each of which can hold one's symbol. Automaton has a control unit which is said to be in one of finite number of internal states.

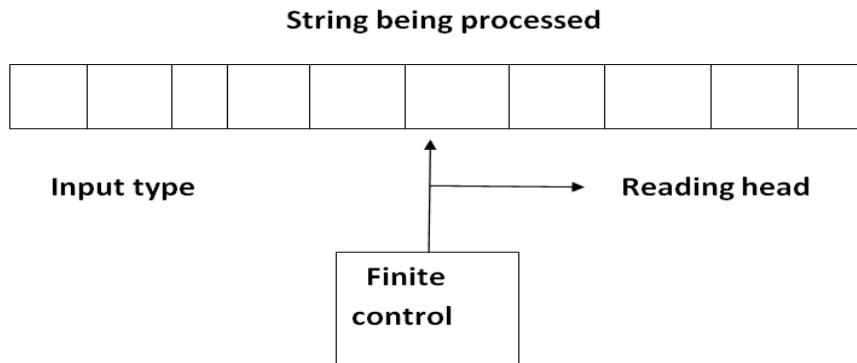


Figure 2.1: Finite Automata

#### Formal definition of Finite Automata:

A finite automaton is a 5-tuple  $M = (Q, \Sigma, \delta, q, F)$ , where

1.  $Q$  is a finite set, whose elements are called states,
2.  $\Sigma$  is a finite set, called the alphabet; the elements of  $\Sigma$  are called symbols,
3.  $\delta: Q \times \Sigma \rightarrow Q$  is a function, called the transition function,
4.  $q$  is an element of  $Q$ ; it is called the start state or initial state,
5.  $F$  is a subset of  $Q$ ; the elements of  $F$  are called accept states or final state.

#### Related Terminologies:

**Alphabet:** An alphabet is any finite set of symbols.

Example:  $\Sigma = \{a, b, c, d\}$  is an alphabet set where 'a', 'b', 'c', and 'd' are symbols.

**String:** A string is a finite sequence of symbols taken from  $\Sigma$ . Example: 'cabcad' is a valid string on the alphabet set  $\Sigma = \{a, b, c, d\}$

**Length of a String:** It is the number of symbols present in a string. (Denoted by  $|S|$ ). Examples: If  $S = 'cabcad'$ ,  $|S| = 6$ . If  $|S| = 0$ , it is called an empty string (Denoted by  $\lambda$  or  $\epsilon$ )

**Language:** A language is a subset of  $\Sigma^*$  for some alphabet  $\Sigma$ . It can be finite or infinite.

Example: If the language takes all possible strings of length 2 over  $\Sigma = \{a, b\}$ , then  $L = \{ab, bb, ba, aa\}$

Types of Finite Automata:

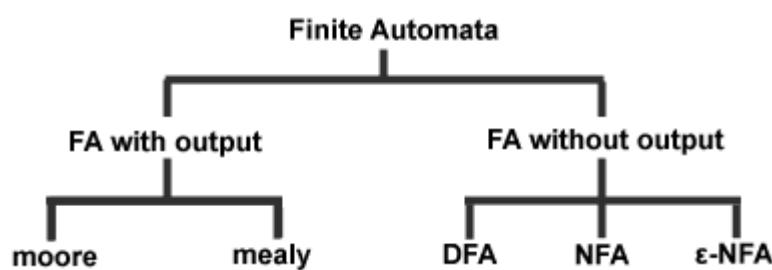


Figure 2.2: Classification of Finite Automata

### An example of finite automata

Let  $A = \{w : w \text{ is a binary string containing an odd number of } 1's\}$ .

We claim that this language  $A$  is regular. In order to prove this, we have to construct a finite automaton  $M$  such that  $A = L(M)$ .

Steps to construct finite automata:

- The FA reads the input string  $w$  from left to right and keep scanning on the number of 1's
- After scanning the entire string, it counts the number of 1's to check whether it is odd or even
- If the number of 1's is odd then the string is acceptable otherwise it is rejected

Using this approach, the finite automaton needs a state for every integer  $i \geq 0$ ; indicating that the number of 1's read so far is equal to  $i$ . Hence, to design a finite automaton that follows this approach, we need an infinite number of states. But, the definition of finite automaton requires the number of states to be finite. A better, and correct approach, is to keep track of whether the number of 1's read so far is even or odd. This leads to the following finite automaton:

- The set of states is  $Q = \{q_0, q_1\}$ . If the finite automaton is in state  $q_1$ , then it has an even number of 1's; if it is in state  $q_0$ , then it has an odd number of 1's.
- The alphabet is  $\Sigma = \{0, 1\}$ .
- The start state is  $q_0$ , because at the start, the number of 1's read by the automaton is equal to 0, and 0 is even.
- The set  $F$  of accept state is  $F = \{q_1\}$ .

- The **transition function**  $\delta$  is given by the following table:

State	Input 0	Input 1
q0	q0	q1
q1	q1	q0

Table 2.1: Transition Table/Matrix

This finite automaton  $M = (Q, \Sigma, \delta, q_0, F)$  can also be represented by its state diagram.

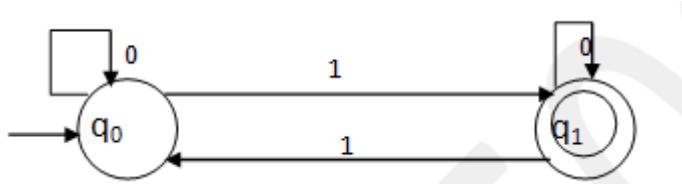


Figure 2.3: Transition Graph

**Transition Graph:** it is a finite directed labeled graph in which each vertex (or node) represents a state and the directed edges indicate the transition of state. Edges are labeled with input symbol.

**Transition Matrix:** It is two-dimension matrixes between states of automata and Input symbol. Elements of matrix are state form mapping ( $\Sigma \times Q$ ) into  $Q$ .

Deterministic Finite Automata (DFA):

Deterministic automaton is one in which each move (transition from one state to another) is uniquely determined by the current configuration. If the internal states input and contents of the storage are known it is possible to predict the next (future) behavior of the automaton.

Formal Definition of a DFA

A DFA can be represented by a **5-tuple**  $(Q, \Sigma, \delta, q_0, F)$  where:

1.  $Q$  is a finite set of states.
2.  $\Sigma$  is a finite set of symbols called the alphabet.
3.  $\delta$  is the transition function where  $\delta: Q \times \Sigma \rightarrow Q$
4.  $q_0$  is the initial state from where any input is processed ( $q_0 \in Q$ ).
5.  $F$  is a set of final state/states of  $Q$  ( $F \subseteq Q$ ).

Non-Deterministic Finite Automata (NDFA):

In NDFA, for a particular input symbol, the machine can move to any combination of the states. In other words, the exact state to which the machine moves cannot be determined. Hence, it is called Non-deterministic Automaton.

Formal Definition of NDFA

An NDFA can be represented by a **5-tuple ( $Q, \Sigma, \delta, q_0, F$ )** where:

1.  $Q$  is a finite set of states.
2.  $\Sigma$  is a finite set of symbols called the alphabets.
3.  $\delta$  is the transition function where  $\delta: Q \times \Sigma \rightarrow 2^Q$

(Here the power set of  $Q$ 's ( $2^Q$ ) has been taken because in case of NDFA, from a state, transition can occur to any combination of  $Q$  states)

4.  $q_0$  is the initial state from where any input is processed ( $q_0 \in Q$ ).
5.  $F$  is a set of final state/states of  $Q$  ( $F \subseteq Q$ ).

Difference between Deterministic finite automata (DFA) and Non deterministic finite automata (NFA):

S. No.	DFA	NFA
1.	DFA stands for deterministic finite automata.	NDFA stands for non-deterministic finite automata.
2.	When processing a string in DFA, there is always a unique state to go next when each character is read. It is because for each state in DFA, there is exactly one state that corresponds to each character being read.	In NDFA several choices may exist for the next state. Can move to more than one states.
3.	DFA cannot use empty string transition.	NDFA can use empty string transition.
4.	In DFA we cannot move from one state to another without consuming a symbol.	NDFA allows $\epsilon$ (null) as the second argument of the transition function. This means that the NDFA can make a transition without consuming an input symbol.
5	For every symbol of the alphabet, there is only one state transition in DFA.	We do not need to specify how the NDFA reacts according to some symbol.
6	DFA can understand as one machine.	NDFA can be understood as multiple title machines computing at the same time.
7	DFA will reject the string if it ends at other than accepting state	If all the branches of NDFA dies or rejects the string, we can say that NDFA reject the string.

<b>8</b>	It is more difficult to construct DFA.	NDFA is easier to construct.
<b>9</b>	DFA requires more space.	NDFA requires less space.
<b>10</b>	For every input and output we can construct DFA machine.	It is not possible to construct an NDFA machine for every input and output.

**Table 2.2: Difference between DFA & NDFA**

**Comparison between Deterministic Finite Automata (DFA) and the Nondeterministic Finite Automata (NFA):**

S. No.	Parameter	NFA	DFA
<b>1.</b>	Power	Same	Same
<b>2.</b>	Supremacy	Not all NFA are DFA.	All DFA are NFA
<b>3.</b>	Transition Function	Maps $Q \rightarrow (\Sigma \cup \{\lambda\} \rightarrow 2^Q)$ , the number of next states is zero or one or more.	$Q \times \Sigma \rightarrow Q$ , the number of next states is exactly one
<b>4.</b>	Time complexity	The time needed for executing an input string is more as compare to DFA.	The time needed for executing an input string is less as compare to NFA.
<b>5.</b>	Space	Less space requires.	More space requires.

**Table 2.3: Comparison between NFA and DFA**

**Equivalence of Deterministic finite automata and Non deterministic finite automata:**

Although the DFA and NFA have distinct definitions, an NFA can be translated to equivalent DFA using the subset construction algorithm. i.e., the constructed DFA and the NFA recognize the same formal language. Both types of automata recognize only regular languages.

Therefore, every language that can be described by some NDFA can also be described by some DFA.

### Theorem

Let  $M = (Q, \Sigma, \delta, q_0, F)$  be a non-deterministic finite automaton. There exists a deterministic finite automaton  $M'$ , such that  $L(M') = L(M)$ .

Therefore, for every NDFA there exists a DFA which simulates the behavior of NDFA. Hence if language  $L$  is accepted by NDFA, then there exist a DFA  $M'$  which also accept  $L$ . Where  $M' = (Q', \Sigma, \delta', q'_0, F')$

Conversion from Non deterministic finite automata (NFA) to Deterministic finite automata (DFA):

In NFA, when a specific input is given to the current state, the machine goes to multiple states. It can have zero, one or more than one move on a given input symbol. On the other hand, in DFA, when a specific input is given to the current state, the machine goes to only one state. DFA has only one move on a given input symbol.

Let,  $M = (Q, \Sigma, \delta, q_0, F)$  is an NFA which accepts the language  $L(M)$ . There should be equivalent DFA denoted by  $M' = (Q', \Sigma, q'_0, \delta', F')$  such that  $L(M) = L(M')$ .

Steps for converting NFA to DFA:

Step 1: Initially  $Q' = \emptyset$

Step 2: Add  $q_0$  of NFA to  $Q'$ . Then find the transitions from this start state.

Step 3: In  $Q'$ , find the possible set of states for each input symbol. If this set of states is not in  $Q'$ , then add it to  $Q'$ . Step 4: In DFA, the final state will be all the states which contain  $F$  (final states of NFA)

### Example 1:

Convert the given Non deterministic finite automata to Deterministic finite automata.

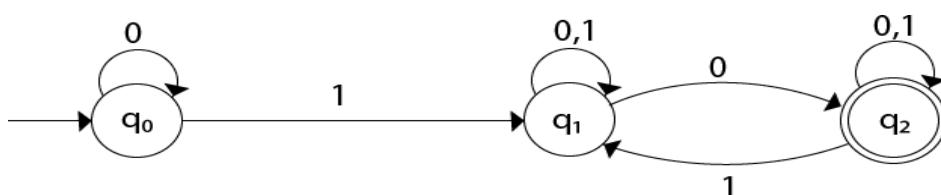


Figure 2.4: Transition Graph

### Solution:

For the given transition diagram we will first construct the transition table.

State	0	1
$\rightarrow q_0$	$q_0$	$q_1$
$q_1$	$\{q_1, q_2\}$	$q_1$
$*q_2$	$q_2$	$\{q_1, q_2\}$

Table 2.4: Transition Table

Now we will obtain  $\delta'$  transition for state  $q_0$ .

1.  $\delta'([q_0], 0) = [q_0]$
2.  $\delta'([q_0], 1) = [q_1]$

The  $\delta'$  transition for state  $q_1$  is obtained as:

1.  $\delta'([q_1], 0) = [q_1, q_2]$  (new state generated)
2.  $\delta'([q_1], 1) = [q_1]$

The  $\delta'$  transition for state  $q_2$  is obtained as: 1.  $\delta'([q_2], 0) = [q_2]$

2.  $\delta'([q_2], 1) = [q_1, q_2]$

Now we will obtain  $\delta'$  transition on  $[q_1, q_2]$ . 1.  $\delta'([q_1, q_2], 0) = \delta(q_1, 0) \cup \delta(q_2, 0)$

2.  $= \{q_1, q_2\} \cup \{q_2\}$
3.  $= [q_1, q_2]$
4.  $\delta'([q_1, q_2], 1) = \delta(q_1, 1) \cup \delta(q_2, 1)$
5.  $= \{q_1\} \cup \{q_1, q_2\}$
6.  $= \{q_1, q_2\}$
7.  $= [q_1, q_2]$

The state  $[q_1, q_2]$  is the final state as well because it contains a final state  $q_2$ .

The transition table for the constructed DFA will be:

State	0	1
$[q_1, q_2]$		

$\rightarrow[q_0]$	$[q_0]$	$[q_1]$
$[q_1]$	$[q_1, q_2]$	$[q_1]$
$*[q_2]$	$[q_2]$	$[q_1, q_2]$
$*[q_1, q_2]$	$[q_1, q_2]$	$[q_1, q_2]$

Table 2.5: Transition Table

The Transition diagram will be:

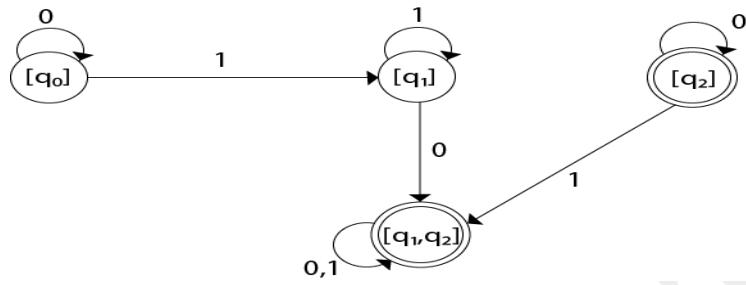


Figure 2.5: Transition Graph

The state  $q_2$  can be eliminated because  $q_2$  is an unreachable state.

#### Minimization of Deterministic Finite Automata (DFA)

Minimization of DFA means reducing the number of states from given FA. Thus, we get the FSM (finite state machine) with redundant states after minimizing the FSM.

We have to follow the various steps to minimize the DFA. These are as follows:

**Step 1:** Remove all the states that are unreachable from the initial state via any set of the transition of DFA.

**Step 2:** Draw the transition table for all pair of states.

**Step 3:** Now split the transition table into two tables T1 and T2. T1 contains all final states, and T2 contains non-final states.

**Step 4:** Find similar rows from T1 such that: 1.  $1 \cdot \delta(q, a) = p$

$$2. \quad 2. \delta(r, a) = p$$

That means, find the two states which have the same value of a and b and remove one of them.

**Step 5:** Repeat step 3 until we find no similar rows available in the transition table T1.

**Step 6:** Repeat step 3 and step 4 for table T2 also.

**Step 7:** Now combine the reduced T1 and T2 tables. The combined transition table is the transition table of minimized DFA.

Example:

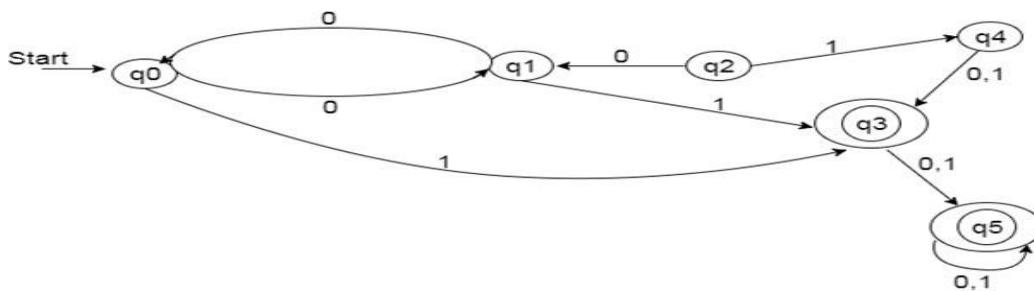


Figure 2.6: Transition Graph

**Solution:**

**Step 1:** In the given DFA, q2 and q4 are the unreachable states so remove them.

**Step 2:** Draw the transition table for the rest of the states.

State	0	1
$\rightarrow q_0$	$q_1$	$q_3$
$q_1$	$q_0$	$q_3$
$*q_3$	$q_5$	$q_5$
$*q_5$	$q_5$	$q_5$

Table 2.6: Transition Table/Matrix

**Step 3:** Now divide rows of transition table into two sets as:

1. One set contains those rows, which start from non-final states:

State	0	1
$q_0$	$q_1$	$q_3$

q1	q0	q3
----	----	----

Table 2.7: Transition Table/Matrix

2. Another set contains those rows, which starts from final states.

State	0	1
q3	q5	q5
q5	q5	q5

Table 2.8: Transition Table/Matrix

**Step 4:** Set 1 has no similar rows so set 1 will be the same.

**Step 5:** In set 2, row 1 and row 2 are similar since q3 and q5 transit to the same state on 0 and 1. So skip q5 and then replace q5 by q3 in the rest.

State	0	1
q3	q3	q3

Table 2.9: Transition Table/Matrix

**Step 6:** Now combine set 1 and set 2 as:

State	0	1
$\rightarrow q_0$	q1	q3
q1	q0	q3
*q3	q3	q3

Table 2.10: Transition Table/Matrix Now it is the transition table of minimized DFA.

## Regular Expression

- The language accepted by finite automata can be easily described by simple expressions called Regular Expressions. It is the most effective way to represent any language.
- The languages accepted by some regular expressions are referred to as Regular languages.
- A regular expression can also be described as a sequence of pattern that defines a string.
- Regular expressions are used to match character combinations in strings. String searching algorithm used this pattern to find the operations on a string.

For instance:

In a regular expression,  $x^*$  means zero or more occurrence of x. It can generate {e,x,xx, xxx,xxxx, ...}

In a regular expression,  $x^+$  means one or more occurrence of x. It can generate{x, xx,xxx,xxxx, ...}

Operations on Regular Language Properties of Regular Sets

**Property 1.** The union of two regular set is regular.

Proof –

Let us take two regular expressions  $RE1 = a(aa)^*$  and  $RE2 = (aa)^*$

So,  $L1 = \{a,aaa,aaaaa, ...\}$  (Strings of odd length excluding Null)

and  $L2 = \{\epsilon, aa,aaa,aaaaa, ...\}$  (Strings of even length including Null)

$L1 \cup L2 = \{\epsilon, a, aa, aaa, aaaa,aaaaa,aaaaaa, ...\}$

(Strings of all possible lengths including Null)

$RE(L1 \cup L2) = a^*$  (which is a regular expression itself)

**Property 2:**The intersection of two regular set is regular.

Proof –

Let us take two regular expressions  $RE1 = a(a^*)$  and  $RE2 = (aa)^*$

So,  $L1 = \{ a, aa, aaa, aaaa, \dots \}$  (Strings of all possible lengths excluding Null)

$L2 = \{ \epsilon, aa, aaaa, aaaaaa, \dots \}$  (Strings of even length including Null)

$L1 \cap L2 = \{ aa, aaaa, aaaaaa, \dots \}$  (Strings of even length excluding Null)

$RE(L1 \cap L2) = aa(aa)^*$  which is a regular expression itself.

**Property 3:** The complement of a regular set is regular.

**Proof –**

Let us take a regular expression –  $RE = (aa)^*$

So,  $L = \{ \epsilon, aa, aaaa, aaaaaa, \dots \}$  (Strings of even length including Null)

Complement of  $L$  is all the strings that is not in  $L$ .

So,  $L' = \{ a, aaa, aaaaa, \dots \}$  (Strings of odd length excluding Null)

$RE(L') = a(aa)^*$  which is a regular expression itself.

**Property 4:** The difference of two regular set is regular.

**Proof –**

Let us take two regular expressions –

$RE1 = a(a^*)$  and  $RE2 = (aa)^*$

So,  $L1 = \{ a, aa, aaa, aaaa, \dots \}$  (Strings of all possible lengths excluding Null)

$L2 = \{ \epsilon, aa, aaaa, aaaaaa, \dots \}$  (Strings of even length including Null)

$L1 - L2 = \{ a, aaa, aaaaa, aaaaaaa, \dots \}$

(Strings of all odd lengths excluding Null)

$RE(L1 - L2) = a(aa)^*$  which is a regular expression.

**Property 5:** The reversal of a regular set is regular.

**Proof –**

We have to prove  $L^R$  is also regular if  $L$  is a regular set. Let,  $L = \{01, 10, 11, 10\}$

$RE(L) = 01 + 10 + 11 + 10$

$L^R = \{10, 01, 11, 01\}$

$RE(L^R) = 01 + 10 + 11 + 10$  which is regular

**Property 6:** The closure of a regular set is regular.

**Proof –**

If  $L = \{a, aaa, aaaaa, \dots\}$  (Strings of odd length excluding Null)

i.e.,  $RE(L) = a(aa)^*$

$L^* = \{a, aa, aaa, aaaa, aaaaa, \dots\}$  (Strings of all lengths excluding Null)

$RE(L^*) = a(a)^*$

**Property 7:** The concatenation of two regular sets is regular.

**Proof –**

Let  $RE1 = (0+1)^*0$  and  $RE2 = 01(0+1)^*$

Here,  $L1 = \{0, 00, 10, 000, 010, \dots\}$  (Set of strings ending in 0)

and  $L2 = \{01, 010, 011, \dots\}$  (Set of strings beginning with 01)

Then,  $L1 \cdot L2 = \{001, 0010, 0011, 0001, 00010, 00011, 1001, 10010, \dots\}$

Set of strings containing 001 as a substring which can be represented by an RE –  $(0 + 1)^*001(0 + 1)^*$  Identities Related to Regular Expressions

Given R, P, L, Q as regular expressions, the following identities hold –

- $\emptyset^* = \epsilon$
- $\epsilon^* = \epsilon$
- $RR^* = R^*R$

- $R^*R^* = R^*$
- $(R^*)^* = R^*$
- $RR^* = R^*R$
- $(PQ)^*P = P(QP)^*$
- $(a+b)^* = (a^*b^*)^* = (a^*+b^*)^* = (a+b^*)^* = a^*(ba^*)^*$
- $R + \emptyset = \emptyset + R = R$  (The identity for union)
- $R \varepsilon = \varepsilon R = R$  (The identity for concatenation)
- $\emptyset L = L \emptyset = \emptyset$  (The annihilator for concatenation)
- $R + R = R$  (Idempotent law)
- $L(M + N) = LM + LN$  (Left distributive law)
- $(M + N)L = ML + NL$  (Right distributive law)
- $\varepsilon + RR^* = \varepsilon + R^*R = R^*$

**Note:** Two regular expressions are equivalent if languages generated by them are same. For example,  $(a+b^*)^*$  and  $(a+b)^*$  generate same language. Every string which is generated by  $(a+b^*)^*$  is also generated by  $(a+b)^*$  and vice versa.

Example 1:

Write the regular expression for the language accepting all combinations of a's, over the set  $\Sigma = \{a\}$

Solution:

All combinations of a's mean a may be zero, single, double and so on. If a is appearing zero times, that means a null string. That is, we expect the set of  $\{\epsilon, a, aa, aaa, \dots\}$ . So, we give a regular expression for this as:

1.  $R = a^*$

That is Kleene closure of a.

Example 2:

Write the regular expression for the language accepting all combinations of as except the null string, over the set  $\Sigma = \{a\}$

Solution:

The regular expression has to be built for the language 1.  $L = \{a, aa, aaa, \dots\}$

This set indicates that there is no null string. So, we can denote regular expression as:  $R = a^+$

Example 3:

Write the regular expression for the language accepting all the string containing any number of a's and b's.

Solution:

The regular expression will be: 1. r.e. =  $(a + b)^*$

This will give the set as  $L = \{\epsilon, a, aa, b, bb, ab, ba, aba, bab, \dots\}$ , any combination of a and b.

The  $(a + b)^*$  shows any combination with a and b even a null string.

Conversion of Regular Expression (RE) to Finite Automata (FA)

To convert the RE to FA, we are going to use a method called the subset method. This method is used to obtain FA from the given regular expression. This method is given below:

**Step 1:** Design a transition diagram for given regular expression, using NFA with  $\epsilon$  moves.

**Step 2:** Convert this NFA with  $\epsilon$  to NFA without  $\epsilon$ .

**Step 3:** Convert the obtained NFA to equivalent DFA.

**Example 1:**

Design a FA from given regular expression  $10 + (0 + 11)0^* 1$ .

**Solution:** First we will construct the transition diagram for a given regular expression.

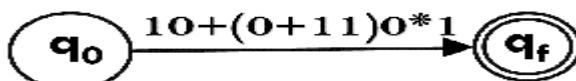


Figure 2.7: Transition Graph

**Step 2:**

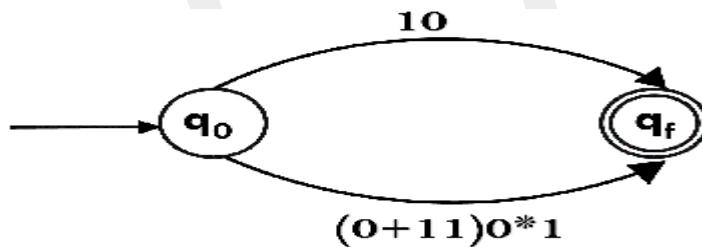


Figure 2.8: Transition Graph

**Step 3:**

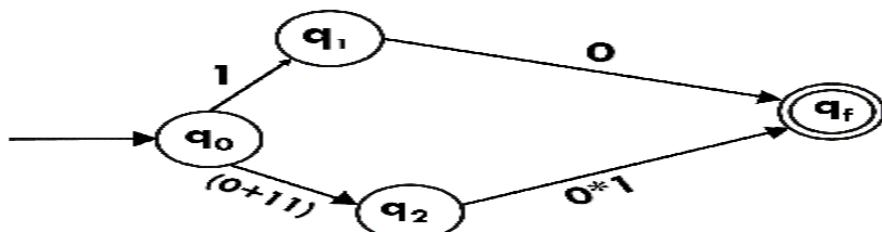


Figure 2.9: Transition Graph

Step 4:

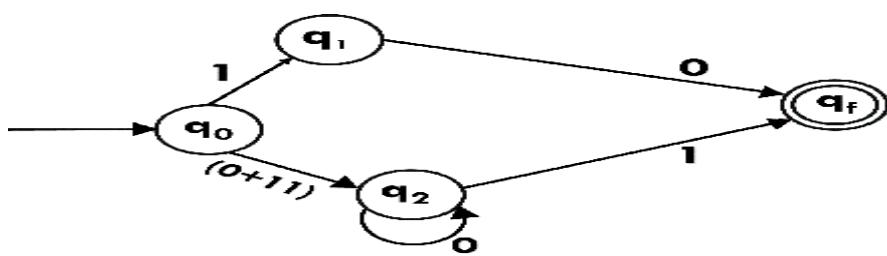


Figure 2.10: Transition Graph

Step 5:

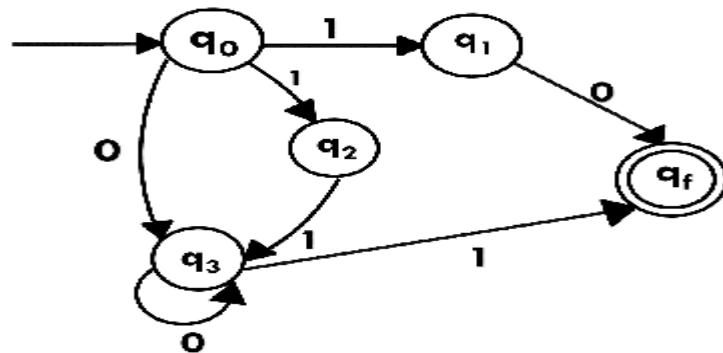


Figure 2.11: Transition Graph

Now we have got NFA without  $\epsilon$ . Now we will convert it into required DFA for that, we will first write a transition table for thisNFA.

State	0	1
$\rightarrow q_0$	$q_3$	$\{q_1, q_2\}$
$q_1$	$q_f$	$\emptyset$
$q_2$	$\emptyset$	$q_3$
$q_3$	$q_3$	$q_f$

<b>*qf</b>	$\emptyset$	$\emptyset$
------------	-------------	-------------

Table 2.11: Transition Table/Matrix

The equivalent DFA will be:

State	0	1
$\rightarrow[q_0]$	$[q_3]$	$[q_1, q_2]$
$[q_1]$	$[q_f]$	$\emptyset$
$[q_2]$	$\emptyset$	$[q_3]$
$[q_3]$	$[q_3]$	$[q_f]$
$[q_1, q_2]$	$[q_f]$	$[q_f]$
$*[q_f]$	$\emptyset$	$\emptyset$

Table 2.12: Transition Table/Matrix

## Construction of Finite Automata (FA) from a Regular Expressions (RE) Method

**Step 1** Construct an NFA with Null moves from the given regular expression.

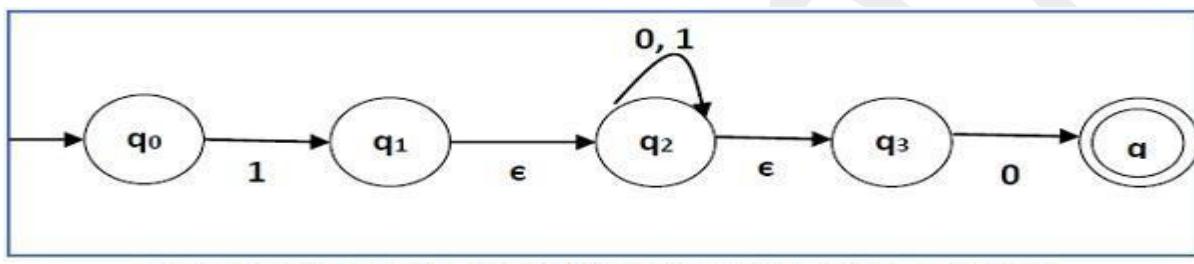
**Step 2** Remove Null transition from the NFA and convert it into its equivalent DFA.

Problem

Convert the following RE into its equivalent DFA –  $1(0 + 1)^* 0$

Solution

We will concatenate three expressions "1", "(0 + 1) $^*$ " and "0"



**NDFA with NULL transition for RA:  $1(0 + 1)^* 0$**

Figure 2.12: Transition Graph

Now we will remove the  $\epsilon$  transitions. After we remove the  $\epsilon$  transitions from the NDFA, we get the following –

**NDFA without NULL transition for RA:  $1(0 + 1)^* 0$**

Figure 2.13: Transition Graph

It is an NDFA corresponding to the RE – 1 (0 + 1) \* 0. If you want to convert it into a DFA, simply apply the method of converting NDFA to DFA

Applications:

- Regular expressions are useful in a wide variety of text processing tasks, and more generally string processing, where the data need not be textual. Common applications include data validation, data scraping (especially web scraping), data wrangling, simple parsing, the production of syntax highlighting systems, and many other tasks.
- While regular expressions would be useful on Internet search engines, processing them across the entire database could consume excessive computer resources depending on the complexity and design of the regular expression.

Arden's Theorem:

In order to find out a regular expression of a Finite Automaton, we use Arden's Theorem along with the properties of regular expressions.

Statement:

Let P and Q be two regular expressions.

If P does not contain null string, then  $R = Q + RP$  has a unique solution that is  $R = QP^*$

Proof:

$$R = Q + (Q + RP) P \quad [\text{After putting the value } R = Q + RP] \quad R = Q + QP + RPP$$

When we put the value of R recursively again and again, we get the following equation:  $R = Q + QP + QP^2 + QP^3 \dots$

$$R = Q (\epsilon + P + P^2 + P^3 + \dots)$$

$$R = QP^* \quad [\text{As } P^* \text{ represents } (\epsilon + P + P^2 + P^3 + \dots)] \quad \text{Hence, proved.}$$

Assumptions for Applying Arden's Theorem:

1. The transition diagram must not have NULL transitions
2. It must have only one initial state

Following algorithm is used to build the regular expression form given DFA.

1. Let  $q_1$  be the initial state.
2. There are  $q_2, q_3, q_4 \dots q_n$  number of states. The final state may be some  $q_j$  where  $j \leq n$ .
3. Let  $\alpha_{ji}$  represents the transition from  $q_j$  to  $q_i$ .
4. Calculate  $q_i$  such that  $q_i = \alpha_{j1}^* q_j$

If  $q_j$  is a start state then we have:

$$q_i = \alpha_{j1}^* q_j + \epsilon$$

5. Similarly, compute the final state which ultimately gives the regular expression ' $r$ '.

Example:

**Construct the regular expression for the given Deterministic finite automata**

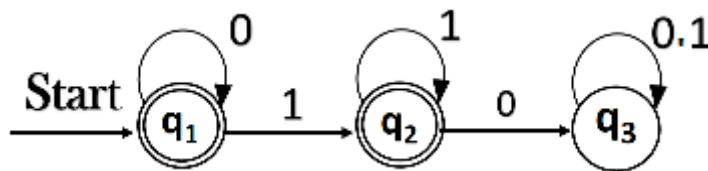


Figure 2.14: Transition Graph

**Solution:**

Let us write down the equations  $q_1 = q_1 0 + \epsilon$

Since  $q_1$  is the start state, so  $\epsilon$  will be added, and the input 0 is coming to  $q_1$  from  $q_1$  hence we write  
State = source state of input  $\times$  input coming to it

Similarly,

$$q_2 = q_1 1 + q_2 1$$

$$q_3 = q_2 0 + q_3 (0+1)$$

Since the final states are  $q_1$  and  $q_2$ , we are interested in solving  $q_1$  and  $q_2$  only. Let us see  $q_1$  first  $q_1 = q_1 0 + \epsilon$

We can re-write it as  $q_1 = \epsilon + q_1 0$

Which is similar to  $R = Q + RP$ , and gets reduced to  $R = OP^*$ . Assuming  $R = q_1$ ,  $Q = \epsilon$ ,  $P = 0$

We get

$$q_1 = \epsilon \cdot (0)^*$$

$$q_1 = 0^* (\epsilon \cdot R^* = R^*)$$

Substituting the value into  $q_2$ , we will get  $q_2 = 0^* 1 + q_2 1$

$$q_2 = 0^* 1 (1)^* (R = Q + RP \rightarrow Q P^*)$$

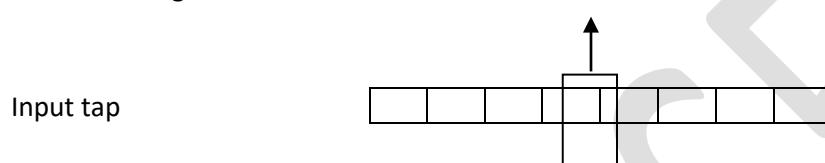
The regular expression is given by  $r = q_1 + q_2$

$$= 0^* + 0^* 1 \cdot 1^*$$

$$r = 0^* + 0^* 1^+ (1 \cdot 1^* = 1^+)$$

Two-way finite automata

The finite automata discussed so far has a  $\delta$  transition which indicate where to next from current state on receiving particular input. But 2-way FA is a model in which linear direction is mentioned on receiving particular and being in some current state. There are two direction that are allowed in 2-FA and these are left and right direction.



Tap head which can move L-R Finite Control

Figure 2.15: Two-way finite automata Formal Definition of Two-way finite automata

It is a collection of 5-tuple  $M = (Q, \Sigma, \delta, q_0, F)$ , where  $Q$ : a finite set of states

$\Sigma$ : a finite set called the input alphabet

$\delta$ : a transition function which maps  $Q \times \{L,R\}$

$q_0$ : a start state (also called initial state) which is an element of  $Q$  ( $q_0 \in Q$ )  $F$ : Set of final states.

Example:

**Consider the transition table. Check the string “101001” whether it is accepted by Two - Way Deterministic finite automata or not?**

Table 2.13:  
Table/Matrix  
string using two -  
finite automata is –

$q_0101001 \Rightarrow$

States	0	1
$\rightarrow q_0$	( $q_0, R$ )	( $q_1, R$ )
$q_1$ (Final State)	( $q_1, R$ )	( $q_2, L$ )
$q_2$	( $q_0, R$ )	( $q_2, L$ )

Transition  
Acceptability of the  
way Deterministic

$1q_101001$

$\Rightarrow 10 q_1 1001$

$\Rightarrow 1q_201001$

$\Rightarrow 10q_01001$

$\Rightarrow 101q_1001$

$\Rightarrow 1010q_101$

$\Rightarrow 10100q_11$

$\Rightarrow 1010q_201$

$\Rightarrow 10100q_01$

$\Rightarrow 101001q_1$

**Meaning of union, intersection, concatenation and closure**

**Properties of Context Free Languages**

**Union:** If  $L_1$  and If  $L_2$  are two context free languages, their union  $L_1 \cup L_2$  will also be context free.

For example:

$L_1 = \{ a^n b^n c^m \mid m \geq 0 \text{ and } n \geq 0 \}$  and  $L_2 = \{ a^n b^m c^m \mid n \geq 0 \text{ and } m \geq 0 \}$

$L_3 = L_1 \cup L_2 = \{ a^n b^n c^m \cup a^n b^m c^m \mid n \geq 0, m \geq 0 \}$  is also context free.

$L_1$  says number of  $a$ 's should be equal to number of  $b$ 's and  $L_2$  says number of  $b$ 's should be equal to number of  $c$ 's. Their union says either of two conditions to be true. So, it is also context free language.

Note: So CFL are closed under Union.

**Concatenation:** If  $L_1$  and  $L_2$  are two context free languages, their concatenation  $L_1.L_2$  will also be context free.

For example:

$L_1 = \{ a^n b^n \mid n \geq 0 \}$  and  $L_2 = \{ c^m d^m \mid m \geq 0 \}$

$L_3 = L_1.L_2 = \{ a^n b^n c^m d^m \mid m \geq 0 \text{ and } n \geq 0 \}$  is also context free.

$L_1$  says number of  $a$ 's should be equal to number of  $b$ 's and  $L_2$  says number of  $c$ 's should be equal to number of  $d$ 's. Their concatenation says first number of  $a$ 's should be equal to number of  $b$ 's, then number of  $c$ 's should be equal to number of  $d$ 's. So, we can create a PDA which will first push for  $a$ 's, pop for  $b$ 's, push for  $c$ 's then pop for  $d$ 's. So, it can be accepted by pushdown automata, hence context free.

Note: So CFL are closed under Concatenation.

**Kleene Closure:** If  $L_1$  is context free, its Kleene closure  $L_1^*$  will also be context free.

For example:

$L_1 = \{ a^n b^n \mid n \geq 0 \}$

$L_1^* = \{ a^n b^n \mid n \geq 0 \}^*$  is also context free.

Note: So, CFL are closed under Kleen Closure.

**Intersection and complementation:** If  $L_1$  and  $L_2$  are two context free languages, their intersection  $L_1 \cap L_2$  need not be context free.

For example:

$L_1 = \{ a^n b^n c^m \mid n \geq 0 \text{ and } m \geq 0 \}$  and  $L_2 = \{ a^m b^n c^n \mid n \geq 0 \text{ and } m \geq 0 \}$

$L_3 = L_1 \cap L_2 = \{ a^n b^n c^n \mid n \geq 0 \}$  need not be context free.

$L_1$  says number of  $a$ 's should be equal to number of  $b$ 's and  $L_2$  says number of  $b$ 's should be equal to number of  $c$ 's. Their intersection says both conditions need to be true, but push down automata can compare only two. So, it cannot be accepted by pushdown automata, hence not context free.

Similarly, complementation of context free language  $L_1$  which is  $\Sigma^* - L_1$ , need not be context free.

Note: So CFL are not closed under Intersection and Complementation.

## **Unit III**

**Syllabus: Introduction of Automata Theory:** Types of grammar, context sensitive grammar, and context free grammar, regular grammar. Derivation trees, ambiguity in grammar, simplification of context free grammar, conversion of grammar to automata machine and vice versa, Chomsky hierarchy of grammar, killing null and unit productions. Chomsky's normal form and Greibach's normal form.

**Unit Objective:** Analyze the grammar, its types, simplification and normal form

**Unit III: Grammars:**

**3.1 Grammars:**

Grammars denote syntactical rules for conversation in natural languages. Linguistics has attempted to define grammars since the inception of natural languages like English, Sanskrit, Mandarin, etc.

In formal language also grammar defines the rule or syntax or structure of the language.

**Example:** "Dog runs"

<Sentence> -><Noun><Verb>

< Noun > ->< Dog >

<Verb>-><Run>

### **Formal Definition of Grammar:**

A grammar G can be formally written as a 4-tuple  $(V, T, S, P)$  where –

- $VN$  is a set of variables or non-terminal symbols.
- $T$  or  $\Sigma$  is a set of Terminal symbols.
- $S$  is a special variable called the Start symbol,  $S \in V$
- $P$  is Production rules for Terminals and Non-terminals.

A production rule has the form  $\alpha \rightarrow \beta$ , where  $\alpha$  and  $\beta$  are strings on  $VN \cup \Sigma$  and least one symbol of  $\alpha$  belongs to  $VN$ .

Example:

Grammar G1 –

$\{\{S, A, B\}, \{a, b\}, S, \{S \rightarrow AB, A \rightarrow a, B \rightarrow b\}\}$

Here:

$S, A,$  and  $B$  are Non-terminal symbols;

$a$  and  $b$  are Terminal symbols

$S$  is the Start symbol,  $S \in N$

Productions,  $P : S \rightarrow AB, A \rightarrow a, B \rightarrow b$

### **3.2 Types of Grammar (Chomsky Classification):**

Type 0 Unrestricted/ Phase structured/ Recursively Enumerable

Type 1 Context Sensitive Grammar (CSG)

Type 2 Context Free Grammar (CFG)

Type 3 Regular Grammar

#### **Type 0: Unrestricted/ Phase structured/ Recursively Enumerable:**

- The productions have no restrictions.
- The productions can be in the form of  $\alpha \rightarrow \beta$  where  $\alpha$  is a string of terminals and non-terminals with at least one non-terminal and  $\alpha$  cannot be null.
- $\beta$  is a string of terminals and non-terminals.

#### **Type 1: Context Sensitive Grammar (CSG):**

- First of all, Type 1 grammar should be Type 0.
- The productions can be in the form of  $\alpha \rightarrow \beta$  & with restriction that  $|\alpha| \leq |\beta|$  that is count of symbol in Left side of the production:  $|\alpha|$  is less than or equal to Right side:  $|\beta|$

### Type 2: Context Free Grammar (CFG):

- First of all, it should be Type 1.
- The productions can be in the form of  $\alpha \rightarrow \beta$  & with restriction that Left hand side of production can have only one variable (Non-Terminal).  $|\alpha| = 1$ .
- There is no restriction on  $\beta$  it can be  $(V \cup T)^*$ .

### Type 3: Regular Grammar:

- First of all, it should be Type 2.
- It is most restricted form of grammar. It should be in the given form only:

$V \rightarrow VT^* / T^*$  or  $V \rightarrow T^*V / T^*$

- It can of two types either left linear or right linear

### 3.2.1 Chomsky hierarchy and classification:

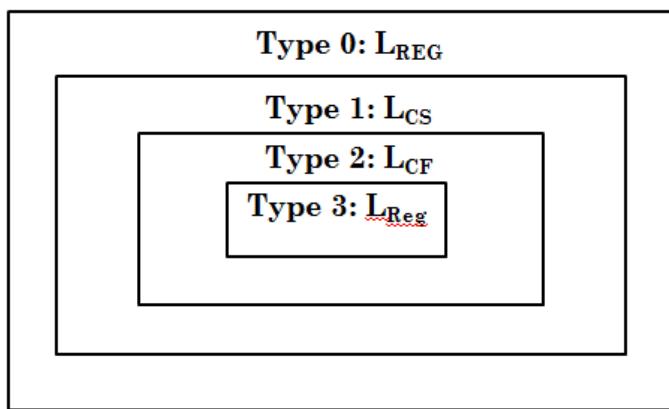
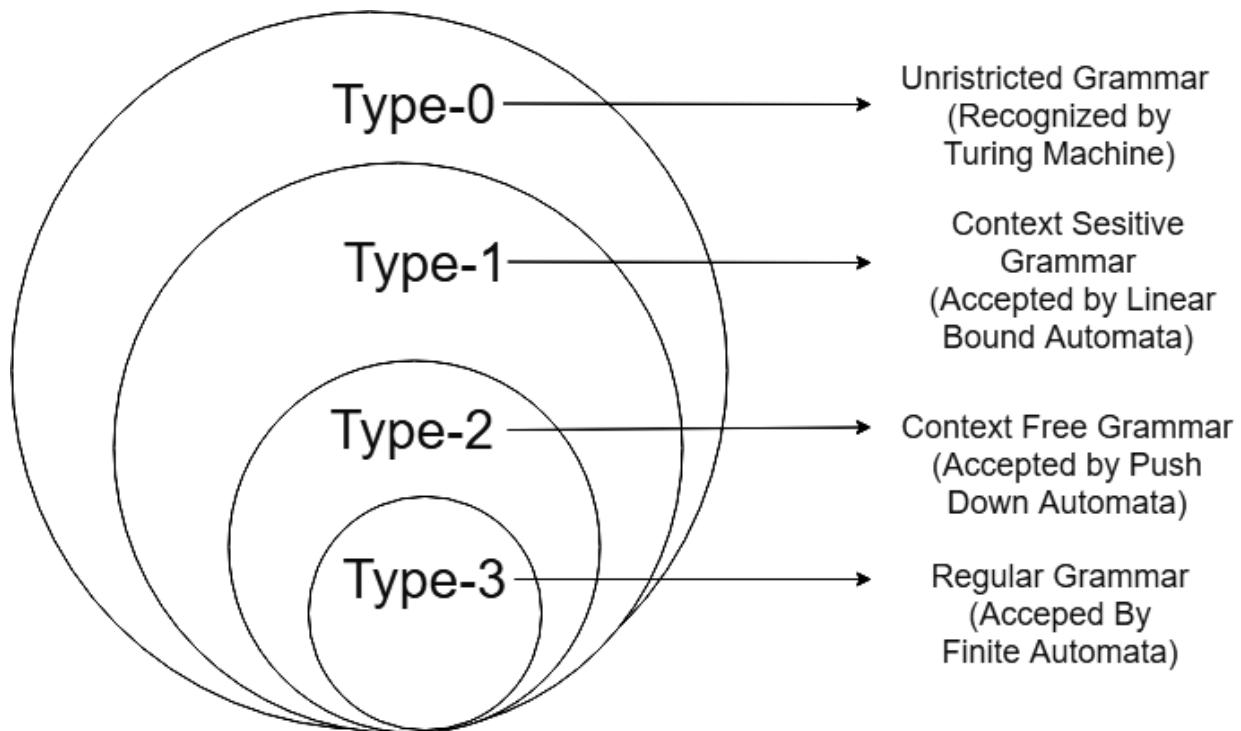


Figure 3.1: Chomsky's hierarchy



**Figure 3.2: Chomsky Classifications**

### 3.3 Derivation

To infer any string by replacing variable using productions of given CFG is known as derivation. Consider CFG  $G = (V, T, P, S)$  and  $\alpha A \beta$  be a string of terminals and variables with  $A$  is variable from  $V$  and  $\alpha, \beta$  are string of terminals and variables. Let's there is production  $A \rightarrow \gamma$  in grammar  $G$  then there will be derivation  $\alpha A \beta \Rightarrow \alpha \gamma \beta$ .

When derivation begins from start symbol  $S$  and end in string of terminals  $T$  then these inferred string of terminal is language of Grammars  $G$ .

#### Example: Grammar: Arithmetic Expression

```

E ----> E + E
E ----> E * E
E ----> (E)
E ----> a | b | c
  
```

Here is a sequence of replacements that leads to a sequence of terminal symbols.

#### LMD (Left Most Derivative):

$E \Rightarrow E * E \Rightarrow (E) * E \Rightarrow (E + E) * E \Rightarrow (a + E) * E \Rightarrow (a + b) * E \Rightarrow (a + b) * c$

In this case, we obtained the final sentence " $(a + b) * c$ " starting from  $E$ . One says that  $E$  derives " $(a + b) * c$ " or (to use passive voice) " $(a + b) * c$  is derived from  $E$ ". The sequence is called a derivation sequence. In this case it is a leftmost derivation sequence, because at each stage the leftmost non-terminal was replaced. Each non-terminal replaced is indicated by red color. If in any case there is only one non-terminal, then it will be leftmost on an honorary basis.

#### RMD (Right Most Derivatives):

$E \Rightarrow E * E \Rightarrow E * c \Rightarrow (E) * c \Rightarrow (E + E) * c \Rightarrow (E + b) * c \Rightarrow (a + b) * c$

This was a rightmost derivation sequence, because at each stage the rightmost non-terminal was replaced.

#### Parse tree:

The sentence  $(a + b) * c$  has a unique leftmost derivation, a unique (different) rightmost derivation. The parse trees are shown below:

Parse Tree:  $(a + b) * c$

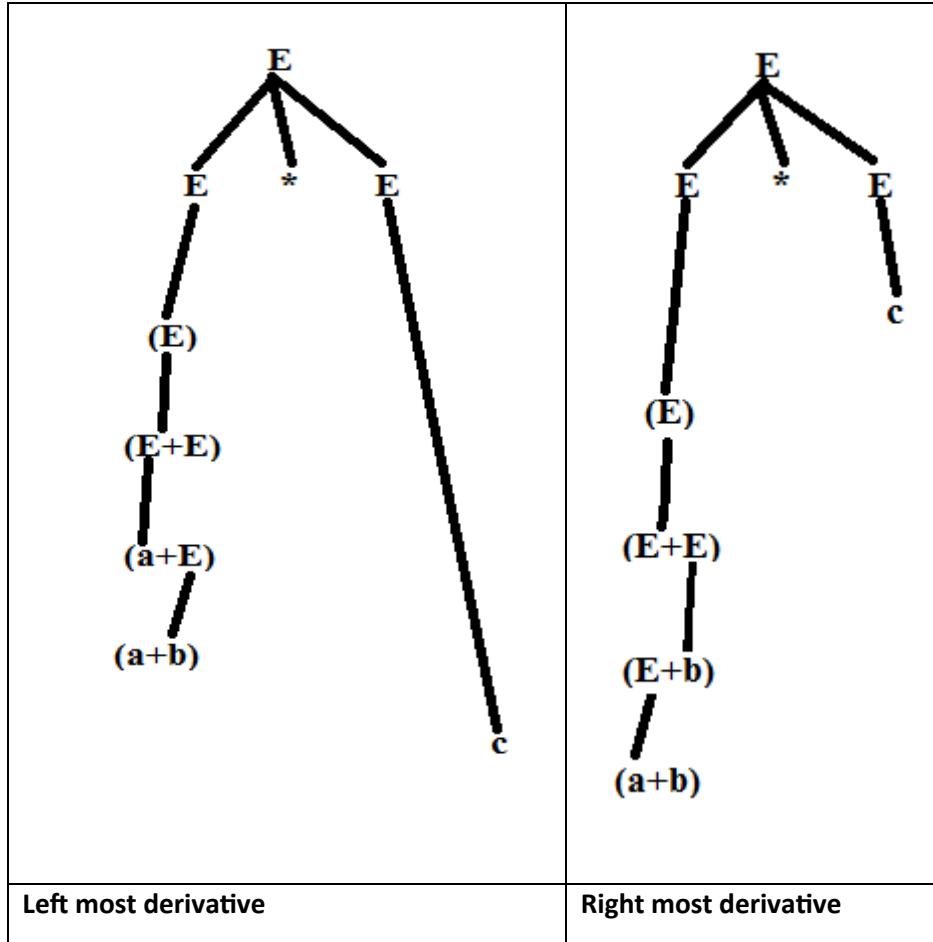


Figure 3.3: Parse Tree

### 3.4 Ambiguity:

Let us consider another sentence “ $a + b * c$ ” derived from above grammar. It can have more than one parse tree.

1st Leftmost Derivative	2nd Leftmost Derivative
$E \implies E+E$	$E \implies E * E$
$\implies a+E$	$\implies E+E * E$
$\implies a+E * E$	$\implies a+E * E$
$\implies a+b * E$	$\implies a+b * E$
$\implies a+b * c$	$\implies a+b * c$

Table 3.1: Two Leftmost derivation for same grammar

1st Parse Tree	2nd Parse Tree
<pre> graph TD     E1[E] --&gt; E1a[E]     E1 --&gt; plus[+]     E1a --&gt; a[a]     E1a --&gt; E1ab[E]     E1ab --&gt; b[b]     E1ab --&gt; E1abc[E]     E1abc --&gt; c[c]   </pre>	<pre> graph TD     E2[E] --&gt; E2a[E]     E2 --&gt; star[*]     E2 --&gt; E2b[E]     E2a --&gt; a[a]     E2b --&gt; b[b]     E2b --&gt; c[c]   </pre>

Figure 3.4: Parse Tree showing Ambiguity

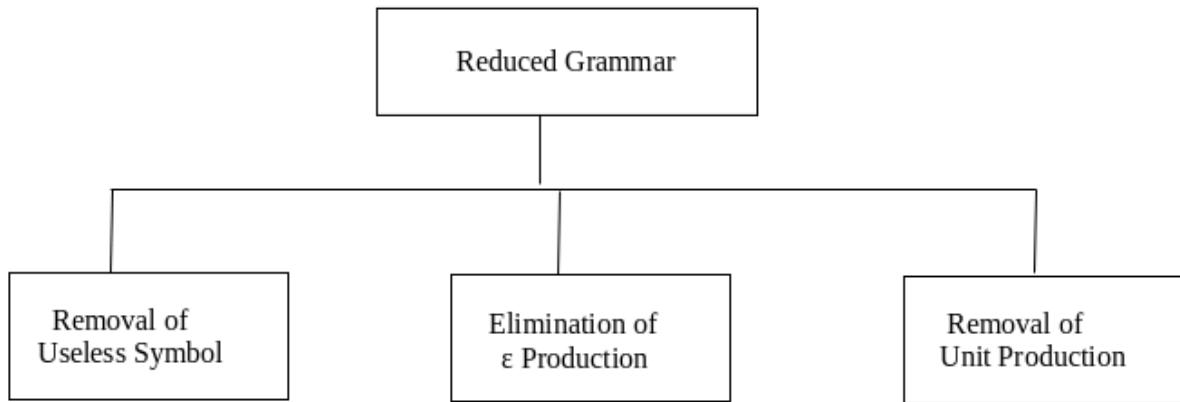
- Some parse trees are unique, but in case if there are multiple parse trees for any sentence, then the grammar is called ambiguous. Here multiple parse trees means if there are two LMDs or two RMDs can be produced from same grammar.
- In a programming language it is not acceptable to have more than one possible reading of a construct. We can't flip a coin to decide which parse tree to use. There are several ways around this problem:
  1. Rewrite the grammar so that it is no longer ambiguous yet still accepts exactly the same language. This is not always possible.
  2. Introduce extra rules that allow the program to decide which of multiple parse trees to use. These are called disambiguating rules.
- An ambiguous grammar may signal problems with language design, and the programming language itself might be changed.

### 3.5 Simplification of CFG:

- As we have seen, various languages can efficiently be represented by a context-free grammar. All the grammars are not always optimized that means the grammar may consist of some extra symbols (non-terminal).
- Grammars having extra symbols, unnecessary increase the length of grammar.
- Simplification of grammar means reduction of grammar by removing useless symbols.

The properties of reduced grammar are given below:

- Each variable (i.e., non-terminal) and each terminal of G appears in the derivation of some word in L.
- There should not be any production as  $X \rightarrow Y$  where X and Y are non-terminal.
- If  $\epsilon$  is not in the language L then there need not to be the production  $X \rightarrow \epsilon$ .



**Figure 3.5: Simplification (Reduction) of Grammar**

### 3.5.1 Elimination of Useless production/symbols from context free grammar:

#### Conditions of Useless Symbol:

- 1) We will entitle any variable useful only when it is deriving any terminal.
- 2) Also, if a symbol is deriving a terminal but not reachable from Start state.
- 3) All terminals will be useful symbols

#### Example: Remove Useless Productions/Symbols:

$$S \rightarrow AB/a$$

$$A \rightarrow BC/b$$

$$B \rightarrow aB/C$$

$$C \rightarrow aC/B$$

#### Solution:

- The symbols 'B' and 'C' having the productions which will never ends by any substitutions.
- So useful Symbols are: {a, b, S, A}

And any combination of useful symbols will also make LHS a useful symbol.

So, we could see that Symbol B and C are useless symbol, remove them (whole production in which it contains):

$S \rightarrow a$

$A \rightarrow b$

Since A is not reachable so we will remove  $A \rightarrow b$  as well:  $S \rightarrow a$

**One more example to remove useless:**

$S \rightarrow AB/AC$

$A \rightarrow aAb/bAa/a$

$B \rightarrow bbA/aaB/AB$

$C \rightarrow abCA/aDb$

$D \rightarrow bD/aC$

**Solution:**

First find out useful Symbols: {a, b, A, B, S}

And useless symbols are: {C, D}

So, remove them and write the whole grammar again:

$S \rightarrow AB$

$A \rightarrow aAb/bAa/a$

$B \rightarrow bbA/aaB/AB$

### 3.5.2 Elimination of null production from context free grammar

If  $\epsilon$  belongs to the language then we are supposed to generate it and thus we will not remove it. Using below example we will understand the whole concept.

**Example 1**

$S \rightarrow aSb/aAb/ab/a$

$A \rightarrow \epsilon$

- To know whether  $\epsilon$  is generated in the CFG or not, we find all variable which are generating  $\epsilon$ .
- So only A is generating  $\epsilon$ . Thus  $\epsilon$  does not belong to the language.

**Now we will proceed with elimination of NULL production:**

- Replace NULL producing symbol with and without in R.H.S. of remaining states and drop the productions which have  $\epsilon$  directly. eg.  $A \rightarrow \epsilon$

$S \rightarrow aSb/aAb/ab/ab/a$  but we no need to write "ab" twice

So,

$S \rightarrow aSb/aAb/ab/a$

### **Example 2 Remove Null productions**

$S \rightarrow AB$

$A \rightarrow aAA/\epsilon$

$B \rightarrow bBB/\epsilon$

Solution: Nullable Variables are {A, B, S}

Because start state also a Nullable symbol so  $\epsilon$  belongs to given CFG

We will proceed with the method:

$S \rightarrow AB/A/B/\epsilon$

$A \rightarrow aAA/aA/a$

$B \rightarrow bAA/bA/b$

### **3.5.3 Elimination of Unit production from context free grammar:**

A Unit production is like below:

$S \rightarrow B$  means  $V \rightarrow V$  that is a single Variable (non-terminal) producing another single Variable (non-terminal).

#### **Steps to remove Unit production:**

1. Write production without Unit production
2. Check what we are missing because of Step 1

#### **Example:**

Given grammar is:

$S \rightarrow Aa/B/c$

$B \rightarrow A/bb$

$A \rightarrow a/bc/B$

Remove unit Productions

#### **Solution:**

Now we will apply step 1:

$S \rightarrow Aa/c$

$B \rightarrow bb$

$A \rightarrow a/bc$

Now check what we are missing after applying Step 1:

First:  $S \rightarrow B \rightarrow bb$

And :  $S \rightarrow B \rightarrow A \rightarrow a$

And :  $S \rightarrow B \rightarrow A \rightarrow bc$

So add these in the production list of "S"

$S \rightarrow Aa/c/bb/a/bc$

$B \rightarrow bb$

$A \rightarrow a/bc$

Second:  $B \rightarrow A \rightarrow a$

And :  $B \rightarrow A \rightarrow bc$

So, add this in the production list of "B"

$S \rightarrow Aa/c/bb/a/bc$

$B \rightarrow bb/a/bc$

$A \rightarrow a/bc$

Third:  $A \rightarrow B \rightarrow bb$

So, add this in the production list of "A"

$S \rightarrow Aa/c/bb/a/bc$

$B \rightarrow bb/a/bc$

$A \rightarrow a/bc/bb$

**One more example with 1 variation:**

$S \rightarrow AC$

$A \rightarrow a$

$C \rightarrow B/d$

$B \rightarrow D$

$D \rightarrow E$

$E \rightarrow b$

**Solution:**

Now apply step 1:

$S \rightarrow AC$

$A \rightarrow a$

$C \rightarrow b$

Now check what we are missing after applying Step 1:

For C:  $C \rightarrow B \rightarrow D \rightarrow E \rightarrow b$

So add this in the production of "C"

$S \rightarrow AC$

$A \rightarrow a$

$C \rightarrow d/b$

For B:  $B \rightarrow D \rightarrow E \rightarrow b$

So add this in the production of "B"

$S \rightarrow AC$

$A \rightarrow a$

$C \rightarrow d/b$

$B \rightarrow b$

Similarly for D and E also add:

$S \rightarrow AC$

$A \rightarrow a$

$C \rightarrow d/b$

$B \rightarrow b$

$D \rightarrow b$

$E \rightarrow b$

But if we see that  $B \rightarrow b$ ,  $D \rightarrow b$  and  $E \rightarrow b$

Productions are useless as they can't be reached, Remove them:

$S \rightarrow AC$

$A \rightarrow a$

$C \rightarrow d/b$

### 3.6 Conversion of Grammar to Finite Automata:

Steps for converting grammar to Finite Automata are:

- Start from the first production
- And then for every left alphabet go to SYMBOL followed by it
- Start State: It will be the first production's state
- Final State: Take those states which end up with input alphabets. e.g., State A and C are below CFG

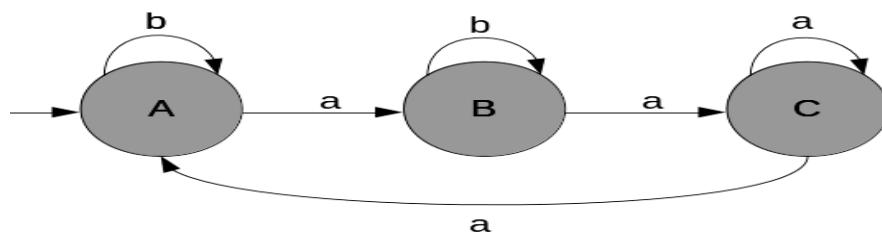
**Example:** Here we are giving one right linear grammar

$A \rightarrow aB/bA/b$

$B \rightarrow aC/bB$

$C \rightarrow aA/bC/a$

now let's see the output and you will understand what just happened.



**Figure 3.6: Grammar to automata**

### Explanation:

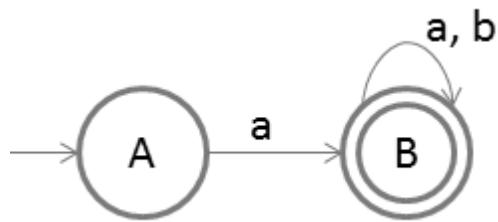
- As you can see for state A, transition on 'a' is going on state B and on 'b' it is going on state A itself. This method will apply to all the states.
- Note: Final states are A and C, because they are having terminal production

### 3.7 Conversion of Finite Automata to Grammar:

Steps for converting finite automata to grammar:

- Repeat the process for every state
- Begin the process from start state
- Write the production as the output followed by the state on which the transition is going
- And at the last add  $\epsilon$  because that's is required to end the derivation
- Note: We have added  $\epsilon$  because either you could continue the derivation or would like to stop it. So, to stop the derivation we have written  $\epsilon$

**Example:**



**Figure 3.7: Given automata**

**Solution:** Applying the above procedure as mentioned in section 3.7

Start with state 'A', we can write as:

$$A \rightarrow aB$$

And then we will pick state B and then we will go for each output.

So we will get the below production.

$$B \rightarrow aB/bB/\epsilon$$

So final we got right linear grammar as:

$$A \rightarrow aB$$

$$B \rightarrow aB/bB/\epsilon$$

### 3.8 Chomsky's Normal Form (CNF):

CNF stands for Chomsky normal form. A CFG (context free grammar) is in CNF (Chomsky normal form) if all production rules satisfy one of the following conditions:

- Start symbol generating  $\epsilon$ . For example,  $A \rightarrow \epsilon$ .
- A non-terminal generating two non-terminals. For example,  $S \rightarrow AB$ .
- A non-terminal generating a terminal. For example,  $S \rightarrow a$ .

### Steps for converting CFG into CNF:

**Step 1:** In the grammar, remove the null, unit and useless productions. (We can refer to the section 3.5 “Simplification of CFG”)

**Step 2:** Eliminate terminals from the RHS of the production if they exist with other non-terminals or terminals. For example, production  $S \rightarrow aA$  can be decomposed as:

$$S \rightarrow RA$$

$$R \rightarrow a$$

**Step 3:** Eliminate RHS with more than two non-terminals. For example,  $S \rightarrow ASB$  can be decomposed as:

$$S \rightarrow RS$$

$$R \rightarrow AS$$

#### Example: Convert given grammar into CNF

$$S \rightarrow bA/aB$$

$$A \rightarrow bAA/aS/a$$

**Solution:** Now we have to add new production:

$$S \rightarrow NA/MB$$

$$A \rightarrow NAA/MS/a$$

$$N \rightarrow a$$

$$M \rightarrow b$$

Now combine NAA with either NA or AA, So

$$S \rightarrow NA/MB$$

$$A \rightarrow NO/MS/a$$

$$N \rightarrow a$$

$$M \rightarrow b$$

$$O \rightarrow AA$$

#### 3.9 Greibach Normal Form (GNF):

GNF stands for Greibach normal form. A CFG (context free grammar) is in GNF (Greibach normal form) if all the production rules satisfy one of the following conditions:

- A start symbol generating  $\epsilon$ . For example,  $S \rightarrow \epsilon$ .
- A non-terminal generating a terminal. For example,  $A \rightarrow a$ .
- A non-terminal generating a terminal which is followed by any number of non-terminals. For example,  $S \rightarrow aASB$ .

**For example:**

$G1 = \{S \rightarrow aAB \mid aB, A \rightarrow aA \mid a, B \rightarrow bB \mid b\}$

$G2 = \{S \rightarrow aAB \mid aB, A \rightarrow aA \mid \epsilon, B \rightarrow bB \mid \epsilon\}$

- The production rules of Grammar G1 satisfy the rules specified for GNF, so the grammar G1 is in GNF.
- However, the production rule of Grammar G2 does not satisfy the rules specified for GNF as  $A \rightarrow \epsilon$  and  $B \rightarrow \epsilon$  contains  $\epsilon$  (only start symbol can generate  $\epsilon$ ). So, the grammar G2 is not in GNF.

**Steps for converting CFG into GNF:**

**Step 1:** Convert the grammar into CNF, if the given grammar is not in CNF. (We can refer to section 3.8 to convert the CFG into CNF: Chomsky normal form)

**Step 2:** If the grammar exists left recursion, eliminate it.

**Step 3:** In the grammar, convert the given production rule into GNF form.

**Example: Convert CFG to GNF**

$S \rightarrow XB \mid AA$

$A \rightarrow a \mid SA$

$B \rightarrow b$

$X \rightarrow a$

**Solution:**

As the given grammar G is already in CNF and there is no left recursion, so we can skip step 1 and step 2 and directly go to step 3.

The production rule  $A \rightarrow SA$  is not in GNF,

so we substitute  $S \rightarrow XB \mid AA$  in the production rule  $A \rightarrow SA$  as:

$S \rightarrow XB \mid AA$

$A \rightarrow a \mid XBA \mid AAA$

$B \rightarrow b$

$X \rightarrow a$

The production rule  $S \rightarrow XB$  and  $B \rightarrow XBA$  is not in GNF, so we substitute  $X \rightarrow a$  in the production rule  $S \rightarrow XB$  and  $B \rightarrow XBA$  as:

$S \rightarrow aB \mid AA$

$A \rightarrow a \mid aBA \mid AAA$

$B \rightarrow b$

$X \rightarrow a$

Now we will remove left recursion ( $A \rightarrow AAA$ ), we get:

$S \rightarrow aB \mid AA$

$A \rightarrow aC \mid aBAC$

$C \rightarrow AAC \mid \epsilon$

$B \rightarrow b$

$X \rightarrow a$

Now we will remove null production  $C \rightarrow \epsilon$ , we get:

$S \rightarrow aB \mid AA$

$A \rightarrow aC \mid aBAC \mid a \mid aBA$

$C \rightarrow AAC \mid AA$

$B \rightarrow b$

$X \rightarrow a$

The production rule  $S \rightarrow AA$  is not in GNF, so we substitute  $A \rightarrow aC \mid aBAC \mid a \mid aBA$  in production rule  $S \rightarrow AA$  as:

$S \rightarrow aB \mid aCA \mid aBACA \mid aA \mid aBAA$

$A \rightarrow aC \mid aBAC \mid a \mid aBA$

$C \rightarrow AAC$

$C \rightarrow aCA \mid aBACA \mid aA \mid aBAA$

$B \rightarrow b$

$X \rightarrow a$

The production rule  $C \rightarrow AAC$  is not in GNF, so we substitute  $A \rightarrow aC \mid aBAC \mid a \mid aBA$  in production rule  $C \rightarrow AAC$  as:

$S \rightarrow aB \mid aCA \mid aBACA \mid aA \mid aBAA$

$A \rightarrow aC \mid aBAC \mid a \mid aBA$

$C \rightarrow aCAC \mid aBACAC \mid aAC \mid aBAAC$

$C \rightarrow aCA \mid aBACA \mid aA \mid aBAA$

$B \rightarrow b$

$X \rightarrow a$

Hence, this is the GNF form for the grammar G.

#### Unit-IV

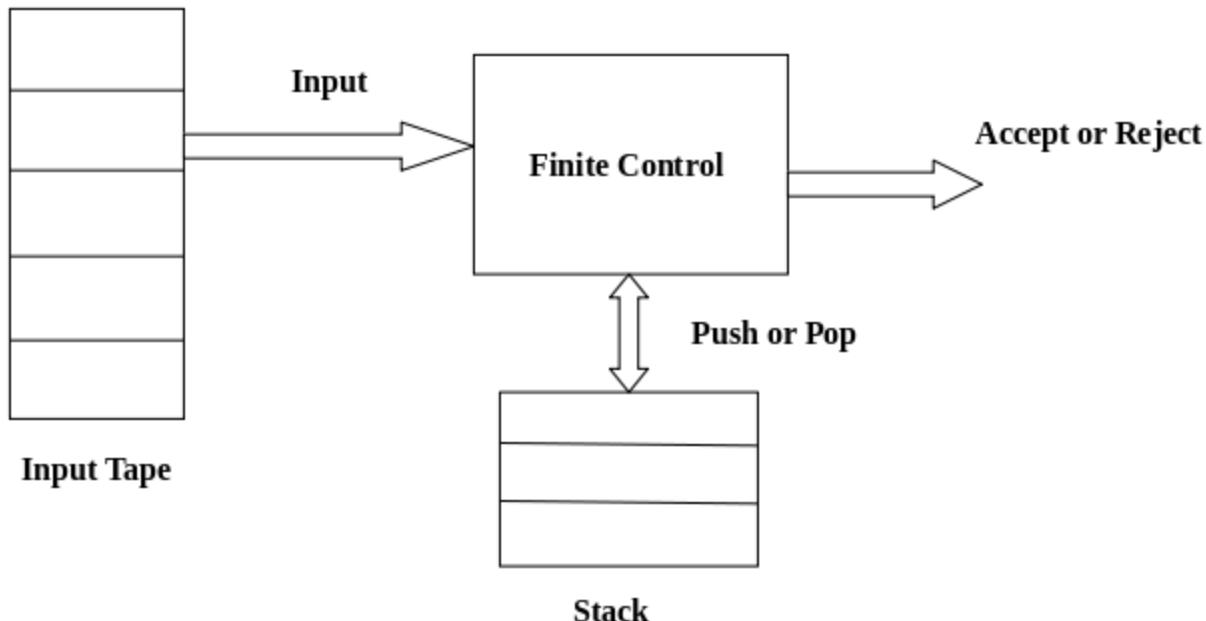
**Syllabus: Push down Automata:** example of PDA, deterministic and non-deterministic PDA, and conversion of PDA into context free grammar and vice versa, CFG equivalent to PDA, Petri net model.

**Objective:** The goal is to make a pushdown automaton that will accept all of the input strings that the context-free grammar accepts.

#### Unit-IV: Pushdown Automata:

A pushdown automaton is a way to implement a context-free grammar in a similar way we design DFA for a regular grammar. A DFA can remember a finite amount of information, but a PDA can remember an infinite amount of information.

Basically, a pushdown automaton is: "Finite state machine" + "a stack"



**Figure 4.1: Pushdown Automata**

A pushdown automaton has three components:

- An input tape,
- A control unit, and
- A stack with infinite size.

The stack head scans the top symbol of the stack.

A stack does two operations:

- Push: a new symbol is added at the top.
- Pop: the top symbol is read and removed.

A PDA may or may not read an input symbol, but it has to read the top of the stack in every transition.

#### **Applications of PDA:**

1. Online Transaction process system.
2. Used in compiler design (parser design for syntactic analysis)
3. Tower of Hanoi (Recursive Solution)

#### **Formal Definition of PDA:**

A deterministic pushdown automaton is a 7 -tuples  $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ , where

- $Q$  is a finite set of states,
- $\Sigma$  is a finite set of tape alphabet or input symbol
- $\Gamma$  is a finite set of stack alphabet
- $q_0$  is initial state,  $q_0$  is an element of  $Q$
- $Z_0$  is Initial symbol on top of stack
- $F$  set of final state which is sub set of  $Q$ .
- $\delta$  is transition function which maps  $(Q \times \Sigma \times \Gamma)$  into  $Q \times \Gamma^*$

#### **Transition of PDA:**

Transition function of PDA is denoted as  $\delta (q, a, X) = (p, Y)$  which specified transition in PDA is function of three components:

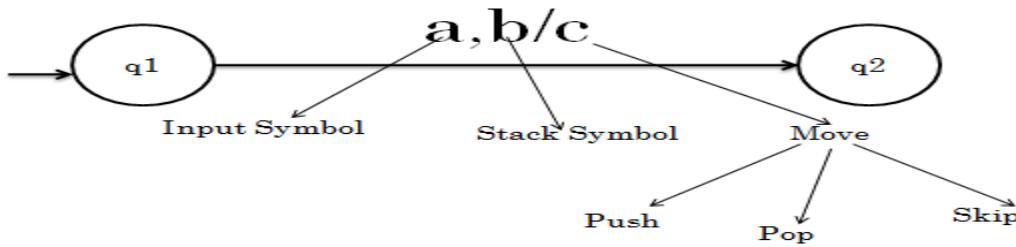
1. Present state of PDA,  $q$
2. Symbol of input alphabet,  $a$ , being read by PDA.
3. Symbol at top of stack,  $X$ .

In every transition

- PDA enters into new state  $p$  or remain into same state  $p = q$
- if  $a = \epsilon$  than no symbol is consumed

- If  $Y = zX$ , symbol  $z$  is pushed into the stack at the top.
- If  $Y = \epsilon$ , Symbol  $X$  at the top of stack is popped.
- If  $Y = X$ , No change of symbol at top of stack.

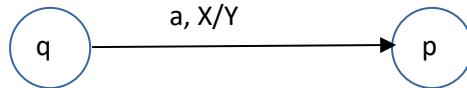
**Example:** The following diagram shows a transition in a PDA from a state  $q_1$  to state  $q_2$ , labeled as " $a,b/c$ "



**Figure 4.2: Example of PDA**

#### Representation of PDA:

Corresponding to transition function  $\delta(q, a, X) = (p, Y)$  transition diagram will have



**Figure 4.3: Representation of PDA**

#### Terminologies related to PDA:

Instantaneous Description:

The instantaneous description (ID) of a PDA is represented by a triplet  $(q, w, s)$  where

- $q$  is the state
- $w$  is unconsumed input
- $s$  is the stack contents

#### Turnstile Notation:

The "turnstile" notation is used for connecting pairs of ID's that represent one or many moves of a PDA. The process of transition is denoted by the turnstile symbol " $\vdash$ ".

Consider a PDA  $(Q, \Sigma, S, \delta, q_0, I, F)$ . A transition can be mathematically represented by the following turnstile notation:

$$(p, aw, T\beta) \vdash (q, w, \alpha b)$$

This implies that while taking a transition from state  $p$  to state  $q$ , the input symbol ‘ $a$ ’ is consumed, and the top of the stack ‘ $T$ ’ is replaced by a new string ‘ $\alpha'$ .

Note: If we want zero or more moves of a PDA, we have to use the symbol ( $\vdash^*$ ) for it.

**Example: Define the pushdown automata for language  $\{a^n b^n \mid n > 0\}$**

**Solution:**  $M = \text{where } Q = \{q_0, q_1\}$  and  $\Sigma = \{a, b\}$  and  $\Gamma = \{A, Z\}$  and  $\delta$  is given by

$$\delta(q_0, a, Z) = \{(q_0, AZ)\}$$

$$\delta(q_0, a, A) = \{(q_0, AA)\}$$

$$\delta(q_0, b, A) = \{(q_1, \epsilon)\}$$

$$\delta(q_1, b, A) = \{(q_1, \epsilon)\}$$

$$\delta(q_1, \epsilon, Z) = \{(q_1, \epsilon)\}$$

Let us see how this automaton works for **aabb**

Deterministic PDA:

A PDA is said to be deterministic if and only if following conditions are met

1.  $\delta(q, a, X)$  has at most one transition.
2.  $\delta(q, \epsilon, X) = \Phi$

**Acceptance of PDA:**

There are two different ways to define PDA acceptability.

**Final State Acceptability:**

In final state acceptability, a PDA accepts a string when, after reading the entire string, the PDA is in a final state. From the starting state, we can make moves that end up in a final state with any stack values. The stack values are irrelevant as long as we end up in a final state.

For a PDA  $(Q, \Sigma, \delta, q_0, I, F)$ , the language accepted by the set of final states  $F$  is:

$$L(PDA) = \{w \mid (q_0, w, I) \xrightarrow{*} (q, \epsilon, x), q \in F\} \text{ for any input stack string } x.$$

**Empty Stack Acceptability:**

Here a PDA accepts a string when, after reading the entire string, the PDA has emptied its stack.

For a PDA  $(Q, \Sigma, \delta, q_0, I, F)$ , the language accepted by the empty stack is:

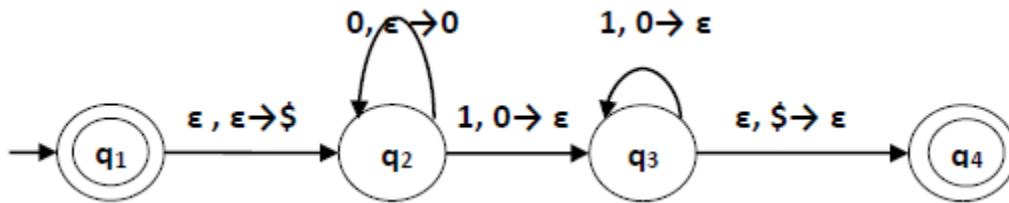
$$L(PDA) = \{w \mid (q_0, w, I) \xrightarrow{*} (q, \epsilon, \epsilon), q \in Q\}$$

**Example: Construct a PDA that accepts  $L = \{0^n 1^n \mid n \geq 0\}$**

**Solution:** This language accepts  $L = \{\epsilon, 01, 0011, 000111, \dots\}$

Here, in this example, the number of '0' and '1' has to be equal. So the logic is as follows:

- Initially we put a special symbol '\$' into the empty stack.
- Then at state  $q_2$ , if we encounter input 0 and top is Null, we push 0 into stack. This may iterate. And if we encounter input 1 and top is 0, we pop this 0.
- Then at state  $q_3$ , if we encounter input 1 and top is 0, we pop this 0. This may also iterate. And if we encounter input 1 and top is 0, we pop the top element.
- If the special symbol '\$' is encountered at top of the stack, it is popped out and it finally goes to the accepting state  $q_4$ .



**Figure 4.4: Example of PDA**

**Example: Construct a PDA that accepts the language L over {0, 1} by empty stack which accepts all the string of 0's and 1's in which a number of 0's are twice of number of 1's.**

Solution: There are two parts for designing this PDA:

- If 1 comes before any 0's
- If 0 comes before any 1's.

We are going to design the first part i.e., 1 comes before 0's. The logic is that read single 1 and push two 1's onto the stack. Thereafter on reading two 0's, POP two 1's from the stack. The  $\delta$  can be

1.  $\delta(q_0, 1, Z) = (q_0, 11, Z)$  Here Z represents that stack is empty
2.  $\delta(q_0, 0, 1) = (q_0, \epsilon)$

Now, consider the second part i.e., if 0 comes before 1's. The logic is that read first 0, push it onto the stack and change state from  $q_0$  to  $q_1$ . [Note that state  $q_1$  indicates that first 0 is read and still second 0 has yet to read].

Being with  $q_1$  state, if 1 is encountered then POP 0. Being in  $q_1$  state, if 0 is read then simply read that second 0 and move ahead. The  $\delta$  will be:

1.  $\delta(q_0, 0, Z) = (q_1, 0Z)$
2.  $\delta(q_1, 0, 0) = (q_1, 0)$
3.  $\delta(q_1, 0, Z) = (q_0, \epsilon)$  (indicate that one 0 and one 1 is already read, so simply read the second 0)
4.  $\delta(q_1, 1, 0) = (q_1, \epsilon)$

Now, summarize the complete PDA for given L is:

1.  $\delta(q_0, 1, Z) = (q_0, 1Z)$
2.  $\delta(q_0, 0, 1) = (q_1, \epsilon)$
3.  $\delta(q_0, 0, Z) = (q_1, 0Z)$
4.  $\delta(q_1, 0, 0) = (q_1, 0)$
5.  $\delta(q_1, 0, Z) = (q_0, \epsilon)$
6.  $\delta(q_0, \epsilon, Z) = (q_0, \epsilon)$  ACCEPT state

### **Non-deterministic Pushdown Automata**

The non-deterministic pushdown automata are very much similar to NFA (Non deterministic finite automata).

The CFG which accepts deterministic PDA accepts non-deterministic PDAs as well. Similarly, there are some CFGs which can be accepted only by NPDA and not by DPDA. Thus, NPDA is more powerful than DPDA.

Example: Design PDA for Palindrome strings of a's and b's.

Solution: Suppose the language consists of string  $L = \{aba, aa, bb, bab, bbabb, aabaa, \dots\}$ . The string can be odd palindrome or even palindrome.

Logic:

- 1) Push all the symbols either number of a's or number of b's into the stack until half of the string.
- 2) After half of the string pushed into the stack. Perform pop operation as mentioned in step 3).
- 3) Every single 'a' will pop every single 'a' means if 'a' is read out and top symbol of stack is also 'a'.
- 4) Similarly every single 'b' will pop every single 'b' means if 'b' is read out and top symbol of stack is also 'b'.
- 5) If stack becomes empty and no symbol remains for reading then this conditions prove that the given string is a palindrome.

This PDA is a non-deterministic PDA because finding the mid for the given string and reading the string from left and matching it with from right (reverse) direction leads to non-deterministic moves. Here is the ID.

1. $\delta(q_1, a, Z) = (q_1, aZ)$	
2. $\delta(q_0, b, Z) = (q_1, bZ)$	
3. $\delta(q_0, a, a) = (q_1, aa)$	
4. $\delta(q_1, a, b) = (q_1, ab)$	Pushing the symbols onto the stack
5. $\delta(q_1, a, b) = (q_1, ba)$	
6. $\delta(q_1, b, b) = (q_1, bb)$	
7. $\delta(q_1, a, a) = (q_2, \epsilon)$	
8. $\delta(q_1, b, b) = (q_2, \epsilon)$	
9. $\delta(q_2, a, a) = (q_2, \epsilon)$	
10. $\delta(q_2, b, b) = (q_2, \epsilon)$	Popping the symbols on reading the same kind of symbol
11. $\delta(q_2, \epsilon, Z) = (q_2, \epsilon)$	

Figure 4.5: Instantaneous description (IDs) of given PDA

Simulation of “abaaba”

1.  $\delta(q_1, abaaba, Z)$       Apply rule 1
2.  $\vdash \delta(q_1, baaba, aZ)$       Apply rule 5
3.  $\vdash \delta(q_1, aaba, baZ)$       Apply rule 4
4.  $\vdash \delta(q_1, aba, abaZ)$       Apply rule 7
5.  $\vdash \delta(q_2, ba, baZ)$       Apply rule 8
6.  $\vdash \delta(q_2, a, aZ)$       Apply rule 7
7.  $\vdash \delta(q_2, \epsilon, Z)$       Apply rule 11
8.  $\vdash \delta(q_2, \epsilon)$       Accept

#### PDA & Context Free Grammar:

If a grammar G is context-free, we can build an equivalent nondeterministic PDA which accepts the language that is produced by the context-free grammar G. A parser can be built for the grammar G.

Also, if P is a pushdown automaton, an equivalent context-free grammar G can be constructed where

$$L(G) = L(P)$$

#### Algorithm to find PDA corresponding to a given CFG:

Step 1: Convert the given productions of CFG into GNF.

Step 2: The PDA will only have one state {q}.

Step 3: The initial symbol of CFG will be the initial symbol in the PDA.

Step 4: For non-terminal symbol, add the following rule:

$$1. \delta(q, \epsilon, A) = (q, \alpha)$$

Where the production rule is  $A \rightarrow \alpha$

Step 5: For each terminal symbols, add the following rule:

$$1. \delta(q, a, a) = (q, \epsilon) \text{ for every terminal symbol}$$

**Example 1:**

Convert the following grammar to a PDA that accepts the same language.

$$1. S \rightarrow 0S1 \mid A$$

$$2. A \rightarrow 1A0 \mid S \mid \epsilon$$

Solution:

The CFG can be first simplified by eliminating unit productions:

$$1. S \rightarrow 0S1 \mid 1S0 \mid \epsilon$$

Now we will convert this CFG to GNF:

$$1. S \rightarrow OSX \mid 1SY \mid \epsilon$$

$$2. X \rightarrow 1$$

$$3. Y \rightarrow 0$$

The PDA can be:

$$R1: \delta(q, \epsilon, S) = \{(q, OSX) \mid (q, 1SY) \mid (q, \epsilon)\}$$

$$R2: \delta(q, \epsilon, X) = \{(q, 1)\}$$

$$R3: \delta(q, \epsilon, Y) = \{(q, 0)\}$$

$$R4: \delta(q, 0, 0) = \{(q, \epsilon)\}$$

$$R5: \delta(q, 1, 1) = \{(q, \epsilon)\}$$

**Example 2:**

Construct PDA for the given CFG, and test whether 0104 is acceptable by this PDA.

$$1. S \rightarrow 0BB$$

$$2. B \rightarrow 0S \mid 1S \mid 0$$

Solution:

The PDA can be given as:

$$1. A = \{(q), (0, 1), (S, B, 0, 1), \delta, q, S, ?\}$$

The production rule  $\delta$  can be:

$$R1: \delta(q, \epsilon, S) = \{(q, 0BB)\}$$

$$R2: \delta(q, \epsilon, B) = \{(q, 0S) \mid (q, 1S) \mid (q, 0)\}$$

$$R3: \delta(q, 0, 0) = \{(q, \epsilon)\}$$

$$R4: \delta(q, 1, 1) = \{(q, \epsilon)\}$$

Testing  $010^4$  i.e.  $010000$  against PDA:

1.  $\delta(q, 010000, S) \vdash \delta(q, 010000, 0BB)$
2.  $\vdash \delta(q, 10000, BB) \quad R1$
3.  $\vdash \delta(q, 10000, 1SB) \quad R3$
4.  $\vdash \delta(q, 0000, SB) \quad R2$
5.  $\vdash \delta(q, 0000, 0BBBB) \quad R1$
6.  $\vdash \delta(q, 000, BBB) \quad R3$
7.  $\vdash \delta(q, 000, 0BB) \quad R2$
8.  $\vdash \delta(q, 00, BB) \quad R3$
9.  $\vdash \delta(q, 00, 0B) \quad R2$
10.  $\vdash \delta(q, 0, B) \quad R3$
11.  $\vdash \delta(q, 0, 0) \quad R2$
12.  $\vdash \delta(q, \epsilon) \quad R3$
13. ACCEPT

Thus  $010^4$  is accepted by the PDA.

Example 3:

Draw a PDA for the CFG given below:

1.  $S \rightarrow aSb$
2.  $S \rightarrow a \mid b \mid \epsilon$

Solution: The PDA can be given as:

$$P = \{(q), (a, b), (S, a, b, z0), \delta, q, z0, q\}$$

The mapping function  $\delta$  will be:

$$R1: \delta(q, \epsilon, S) = \{(q, aSb)\}$$

R2:  $\delta(q, \epsilon, S) = \{(q, a) \mid (q, b) \mid (q, \epsilon)\}$

R3:  $\delta(q, a, a) = \{(q, \epsilon)\}$

R4:  $\delta(q, b, b) = \{(q, \epsilon)\}$

R5:  $\delta(q, \epsilon, z_0) = \{(q, \epsilon)\}$

Simulation: Consider the string aaabb

1.  $\delta(q, \epsilon a a a b b, S) \vdash \delta(q, a a a b b, a S b)$  R3
2.  $\vdash \delta(q, \epsilon a a b b, S b)$  R1
3.  $\vdash \delta(q, a a b b, a S b b)$  R3
4.  $\vdash \delta(q, \epsilon a b b, S b b)$  R2
5.  $\vdash \delta(q, a b b, a b b)$  R3
6.  $\vdash \delta(q, b b, b b)$  R4
7.  $\vdash \delta(q, b, b)$  R4
8.  $\vdash \delta(q, \epsilon, z_0)$  R5
9.  $\vdash \delta(q, \epsilon)$
10. ACCEPT

#### Algorithm to find CFG corresponding to a given PDA

Input – A CFG,  $G = (V, T, P, S)$

Output – Equivalent PDA,  $P = (Q, \Sigma, S, \delta, q_0, I, F)$  such that the non- terminals of the grammar G will be  $\{X_{wx} \mid w, x \in Q\}$  and the start state will be  $A_{q_0, F}$ .

Step 1 – for every  $w, x, y, z \in Q$ ,  $m \in S$  and  $a, b \in \Sigma$ , if  $\delta(w, a, \epsilon)$  contains  $(y, m)$  and  $(z, b, m)$  contains  $(x, \epsilon)$ , add the production rule  $X_{wx} \rightarrow a X_{yz} b$  in grammar G.

Step 2 – for every  $w, x, y, z \in Q$ , add the production rule  $X_{wx} \rightarrow X_{wy} X_{yx}$  in grammar G.

Step 3 – for  $w \in Q$ , add the production rule  $X_{ww} \rightarrow \epsilon$  in grammar G.

Example: Obtain CFG for the given PDA below:

$\delta(q_0, a, z_0) \rightarrow (q_1, az_0)$

$\delta(q_1, a, a) \rightarrow (q_1, aa)$

$\delta(q_1, b, a) \rightarrow (q_2, \lambda)$

$\delta(q_2, b, a) \rightarrow (q_2, \lambda)$

$\delta(q_2, \lambda, z_0) \rightarrow (q_f, \lambda)$

**Sol.** The productions of the grammar are as follows: -

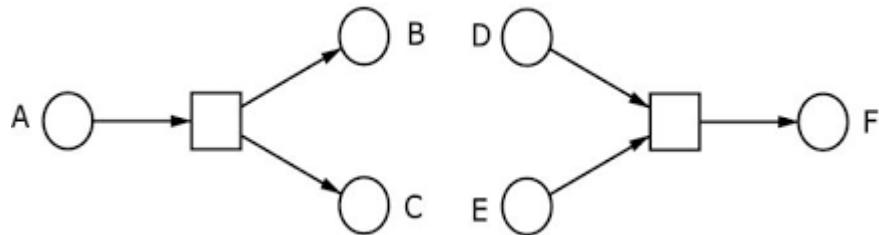
$$S \rightarrow [q_0, z_0, q]$$

- 1)  $[q_0, z_0, q] \rightarrow a [q_1, a, p] [p, z_0, q]$
- 2)  $[q_1, a, q] \rightarrow a [q_1, a, p] [p, z_0, q]$
- 3)  $[q_1, a, q] \rightarrow b$
- 4)  $[q_2, a, q] \rightarrow b$
- 5)  $[q_2, z_0, q] \rightarrow \lambda$

Where p & q are  $q_0$  to  $q_f$  all combinations.

### Petri nets Models

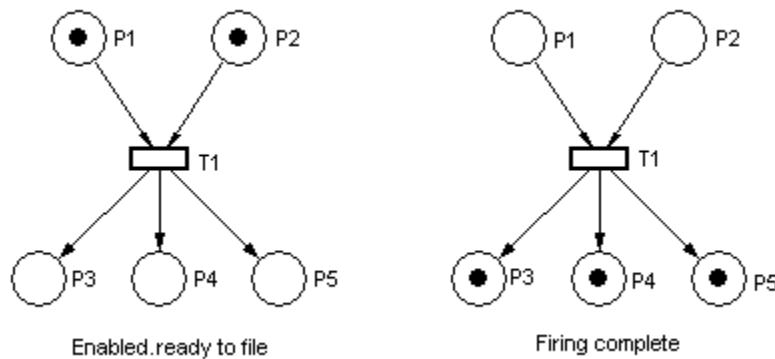
Petri nets are a basic model of parallel and distributed systems (named after Carl Adam Petri). The basic idea is to describe state changes in a system with transitions.



**Figure 4.6: Petri net transitions**

Petri nets contain places Circle and rectangle and transitions that may be connected by directed arcs. Places symbolize states, conditions, or resources that need to be met/be available before an action can be carried out. Transitions symbolize action.

Places may contain tokens that may move to other places by executing (“firing”) actions. A token on a place means that the corresponding condition is fulfilled or that a resource is available:



**Figure 4.7: Petri net execution**

In the example, transition t may “fire” if there are tokens on places p1 and p2. Firing t will remove those tokens and place new tokens on p3, p4 and p5

A Petri net is a tuple  $N = h P, T, F, W, m_0 i$ , where

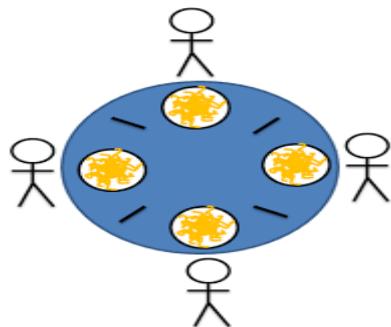
- $P$  is a finite set of places,
- $T$  is a finite set of transitions,
- The places  $P$  and transitions  $T$  are disjoint ( $P \cap T = \emptyset$ ),
- $F \subseteq (P \times T) \cup (T \times P)$  is the flow relation,
- $W: ((P \times T) \cup (T \times P)) \rightarrow \mathbb{N}$  is the arc weight mapping

(Where  $W(f) = 0$  for all  $f \in F$ , and  $W(f) > 0$  for all  $f \in F$ ), and

- $m_0: P \rightarrow \mathbb{N}$  is the initial marking representing the initial distribution of tokens.

### **Example: Dining philosophers**

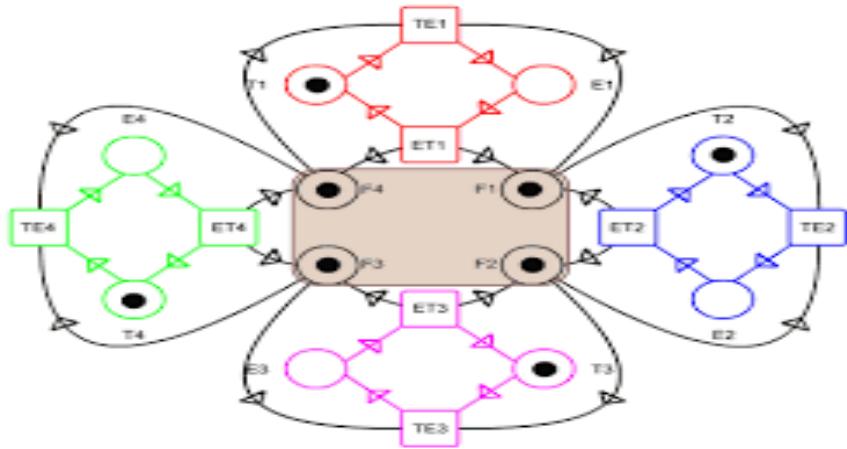
There are philosophers sitting around a round table. There are forks on the table, one between each pair of philosophers.



**Figure 4.8: Dining philosophers**

The philosophers want to eat rice from a large bowl in the center of the table.

### **Dining philosophers: Petri net**



**Figure 4.9: Dining philosophers-Petri net**

## Unit-V

**Turing Machine:** Techniques for construction. Universal Turing machine Multi-tape, multi-head and multidimensional Turing machine, N-P complete problems. Decidability and Recursively Enumerable Languages, decidability, decidable languages, undecidable languages, Halting problem of Turing machine & the post correspondence problem.

**Objective:** To develops an overview of how automata theory, languages and computation are applicable in engineering application.

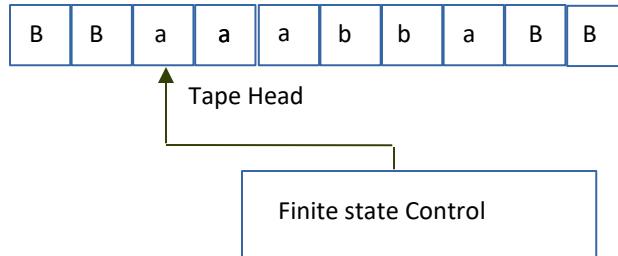
### Turing Machine:

#### Introduction:

Turing machine is considered as a simple model of a real computer. Turing machines can be used to accept all context-free languages, but also languages such as  $L = \{a^m b^n c^m : m \geq 0, n \geq 0\}$  which is not class

of language comes under regular and context free. Every problem that can be solved on a real computer can also be solved by a Turing machine.

### Description of Turing Machine:



**Figure 5.1: Turing Machine**

- There are k tapes, for some fixed  $k \geq 1$  of infinite length. Each tape is divided into cells, each cell stores a symbol belonging to a finite set of tape symbols/ alphabets  $\Gamma$ . B is called blank symbol which also belongs to  $\Gamma$ . If a cell contains B, it represents that the cell is actually empty. (The given diagram is TM of single tape).
- Each tape has a tape head which can move along the tape, one cell per move. It can also read the cell it currently scans and replace the symbol in this cell by another tape symbol.
- There is a finite state control, which can be in any one of a finite number of states. The finite set of states from Q. The set Q contains three special states: a start (initial) state, an accept state, and a reject state.

### Working of Turing Machine:

The Turing machine performs a sequence of computation steps. In one such step, it does the following:

- Immediately before the computation step, the Turing machine is in a state  $q$  of  $Q$ , and tape heads is on a certain cell.
- Depending on the current state  $q$  and the symbol that are read by the tape heads,
- ❖ The Turing machine switches to a state  $p$  of  $Q$  (which may be equal to  $p$ ),
- ❖ Each tape head writes a symbol of  $\Gamma$  in the cell it is currently scanning (this symbol may be equal to the symbol currently stored in the cell).

- ❖ Each tape head either moves one cell to the left, moves one cell to the right, or stays at the current cell.

### **Techniques of Construction:**

**Formal Definition of Turing Machine:** A Deterministic Turing machine is a 7-tuple

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  where

- $Q$  is a finite set of states,
- $\Sigma$  is a finite set of input alphabet; the blank symbol  $B$  is not contained in  $\Sigma$ ,
- $\Gamma$  is a finite set of tape alphabet; this alphabet also contains the blank symbol  $B$  and  $\Sigma \subseteq \Gamma$ ,
- $\delta$  is called the transition function, which maps:  $Q \times \Gamma$  into  $Q \times \Gamma \times D$ .
- $q_0$  is start state, element of  $Q$
- $B$  is Blank symbol  $\delta$  element of  $\Gamma$ ,
- $F$  is Set of final states which is subset of  $Q$ .

### **Transitions occur in Turing Machine:**

Transition function of TM is denoted as  $\delta(q, X) = (p, Y, D)$  which specified transition in TM is function of two components:

- Present state of TM,  $q$
- Tape Symbol  $X$ , being scanned by TM.

In every transition:-

- TM enters into new state  $p$  or remain into same state  $p = q$ .
- A new symbol  $Y$  is written on the scanning cell in place of symbol  $X$ .
- If  $Y = X$  then there is no change in symbol of scanning cell.
- If  $D = L$  or  $\leftarrow$ , tape head moves one cell left to cell being scanned.
- If  $D = R$  or  $\rightarrow$ , tape head moves one cell right to cell being scanned.

### **Computation by Turing Machine:**

Consider TM,  $T = (\{q_1 q_2 q_3 q_4 q_5\}, \{0, 1\}, \{0, 1, b\}, \delta, q_1, B, \{q_5\})$

Transition Function  $\delta$  is given by following transition table

Present State	Tape Symbol		
	B	0	1
$\rightarrow q_1$	$(q_2, 1, L)$	$(q_1, 0, R)$	-
$q_2$	$(q_3, b, R)$	$(q_2, 0, L)$	$(q_2, 1, L)$
$q_3$	-	$(q_0, b, R)$	$(q_5, b, R)$
$q_4$	$(q_5, 0, R)$	$(q_4, 0, R)$	$(q_4, 1, R)$
$q_5$	$(q_2, 0, L)$	-	-

Table 5.1: Computation by Turing Machine

Computation sequence for input string  $w = 00$

Initial ID:  $q_100$

$q_100 \mid - 0q_10 \mid - 00q_1 \mid - 0q_201 \mid - q_2001 \mid - q_2b001 \mid - q_3001 \mid - q_401 \mid - 0q_41 \mid - 01q_4 \mid - 010q_5 \mid - 01q_200 \mid - * - q_5000$

Transition Diagram of Turing Machine:

For transition functions  $\delta(q, \beta) = (p, Y, D)$  the transition diagram will have

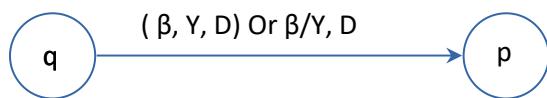


Figure 5.2: Transition Diagram

### **Acceptance of Language by Turing Machine:**

A TM accepts a language if it enters into a final state for any input string  $w$ . A language is recursively enumerable (generated by Type-0 grammar) if it is accepted by a Turing machine.

A TM decides a language if it accepts it and enters into a rejecting state for any input not in the language. A language is recursive if it is decided by a Turing machine. There may be some cases where a TM does not stop. Such TM accepts the language, but it does not decide it.

### **Example 1:**

Design a TM to recognize all strings consisting of an odd number of  $\alpha$ 's.

Solution:

- The Turing machine M can be constructed by the following moves:
- Let  $q_1$  be the initial state.
- If M is in  $q_1$ ; on scanning  $\alpha$ , it enters the state  $q_2$  and writes B (blank).
- If M is in  $q_2$ ; on scanning  $\alpha$ , it enters the state  $q_1$  and writes B (blank).

From the above moves, we can see that M enters the state  $q_1$  if it scans an even number of  $\alpha$ 's, and it enters the state  $q_2$  if it scans an odd number of  $\alpha$ 's. Hence  $q_2$  is the only accepting state.

Hence,

$M = \{q_1, q_2\}, \{1\}, \{1, B\}, \delta, q_1, B, \{q_2\}$  where  $\delta$  is given by:

Tape Alphabet Symbol	Present State "q1"	Present State "q2"
A	BR $q_1$	BR $q_2$

**Table 5.2: Example of Turing Machine**

### **Example 2:**

Design a Turing Machine that reads a string representing a binary number and erases all leading 0's in the string. However, if the string comprises of only 0's, it keeps one 0.

Solution:

Let us assume that the input string is terminated by a blank symbol, B, at each end of the string.

The Turing Machine, M, can be constructed by the following moves:

- Let  $q_0$  be the initial state.
- If M is in  $q_0$ , on reading 0, it moves right, enters the state  $q_1$  and erases 0. On reading 1, it enters the state  $q_2$  and moves right.

- If M is in  $q_1$ , on reading 0, it moves right and erases 0, i.e., it replaces 0's by B's. On reaching the leftmost 1, it enters  $q_2$  and moves right. If it reaches B, i.e., the string comprises of only 0's, it moves left and enters the state  $q_3$ .
- If M is in  $q_2$ , on reading either 0 or 1, it moves right. On reaching B, it moves left and enters the state  $q_4$ . This validates that the string comprises only of 0's and 1's.
- If M is in  $q_3$ , it replaces B by 0, moves left and reaches the final state  $q_f$ .
- If M is in  $q_4$ , on reading either 0 or 1, it moves left. On reaching the beginning of the string, i.e., when it reads B, it reaches the final state  $q_f$ .

Hence,  $M = \{q_0, q_1, q_2, q_3, q_4, q_f\}, \{0, 1, B\}, \{1, B\}, \delta, q_0, B, \{q_f\}$  where  $\delta$  is given by:

Tape Alphabet Symbol	Present State "q0"	Present State "q1"	Present State "q2"	Present State "q3"	Present State "q4"
0	BRq1	BRq1	0Rq2	-	0Lq4
1	1Rq2	1Rq2	1Rq2	-	1Lq4
B	BRq1	BLq3	BLq4	0Lqf	BRqf

**Table 5.3: Example of Turing Machine**

#### Universal Turing Machine:

A universal Turing machine (UTM) is a Turing machine that can simulate an arbitrary Turing machine on arbitrary input. The universal machine essentially achieves this by reading both the description of the machine to be simulated as well as the input thereof from its own tape.

A Turing machine is said to be universal Turing machine if it can accept:

- The input data
- An algorithm (description) for computing.
- This is precisely what a general purpose digital computer does. A digital computer accepts a program written in high level language. Thus, a general purpose Turing machine will be called a universal Turing machine if it is powerful enough to simulate the behavior of any digital computer, including any Turing machine itself.
- More precisely, a universal Turing machine can simulate the behavior of an arbitrary Turing machine over any set of input symbols. Thus, it is possible to create a single machine that can be used to compute any computable sequence.
- By modifying our basic model of a Turing machine we can design a universal Turing machine. The modified Turing machine must have a large number of states for simulating even a simple behavior. We can modify our basic model by following techniques :
  - Increase the number of read/write heads

- Increase the number of dimensions of input tape
- Adding a special purpose memory
- All the above modification in the basic model of a Turing machine will almost speed up the operations of the machine can do.

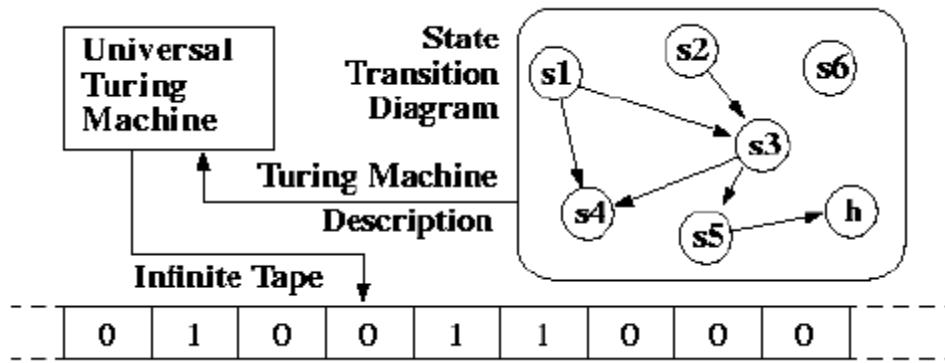
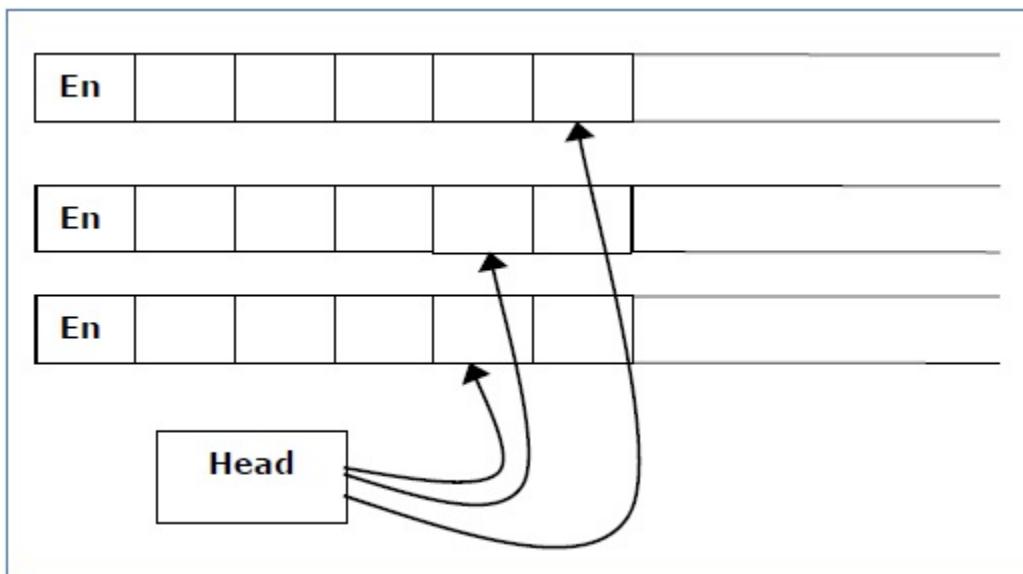


Figure 5.3: Universal Turing Machine

#### Multi-tape Turing Machine:

Multi-tape Turing Machines have multiple tapes where each tape is accessed with a separate head. Each head can move independently of the other heads. Initially the input is on tape 1 and others are blank. At first, the first tape is occupied by the input and the other tapes are kept blank. Next, the machine reads consecutive symbols under its heads and the TM prints a symbol on each tape and moves its heads.



#### **Figure 5.4: Multi-tape Turing Machine**

A Multi-tape Turing machine can be formally described as a 7-tuple  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  where –

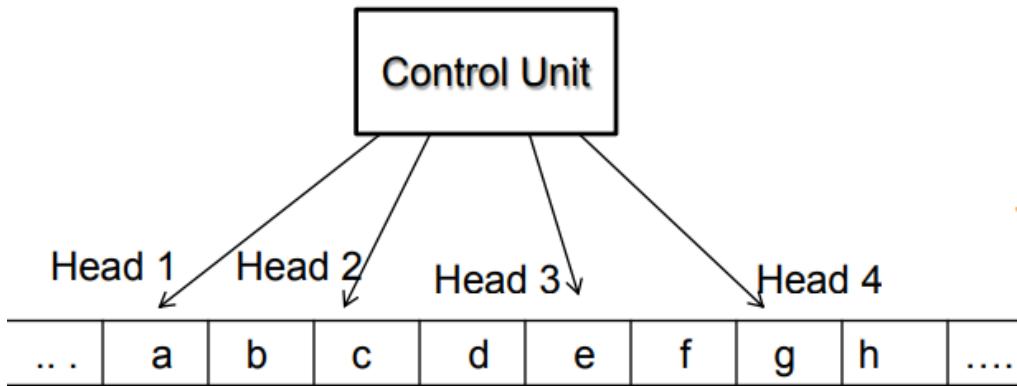
- $Q$  is a finite set of states
- $\Sigma$  is set of input alphabet
- $\Gamma$  is the tape alphabet
- $B$  is the blank symbol
- $\delta$  is a relation on states and symbols where  $\delta: Q \times X^k \rightarrow Q \times (X \times \{\text{Left shift, Right shift, OnShift}\})$   $k$  where there is  $k$  number of tapes
- $q_0$  is the initial state
- $F$  is the set of final states

**Note:** Every Multi-tape Turing machine has an equivalent single-tape Turing machine.

#### **Multi-head Turing Machine:**

- A multi-head Turing machine contains two or more heads to read the symbols on the same tape.
- In one step all the heads sense the scanned symbols and move or write independently.
- Multi-head Turing machine can be simulated by single head Turing machine.
- In this case, there is a single tape but  $k$  heads that can read/write at different places on the tape at the same time.
- This does not increase the computational power of TMs by showing that a multiple-head TM can be simulated by a standard single-head TM.
- The simulation details are similar to those for a multi-tape TM

5.5:  
head



**Machine**

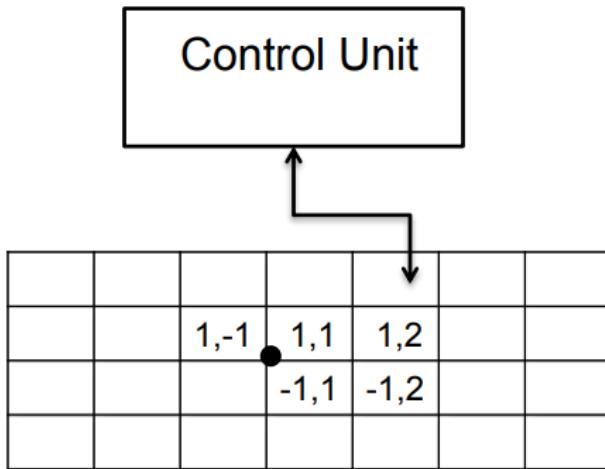
**Multidimensional Turing Machine:**

A multidimensional Turing machine has a multidimensional "tape;" for example, a two-dimensional Turing machine would read and write on an infinite plane divided into squares, like a checkerboard. Possible directions that the tape head could move might be labeled {N, E, S, W}. A three-dimensional Turing machine might have possible directions {N, E, S, W, U, D}.

**Simulation:** Standard TM simulated by Multidimensional TM and multidimensional TM simulated by Standard TM.

**Simulation:**

- The classes are equivalent because a standard Turing machine with two tracks can simulate the computation of a two-dimensional machine
- In simulation for two-dimensional machine, one track is used to store the cell contents and the other one to keep the associated address



**Figure 5.6: Multidimensional Turing Machine**

#### Non-Deterministic Turing Machine:

In a NDTM, for every state and symbol, there are a group of actions the TM can have. So, here the transitions are not deterministic.

An input is accepted if there is at least one node of the tree which is an accept configuration, otherwise it is not accepted. If all branches of the computational tree halt on all inputs, the non-deterministic Turing Machine is called a **Decider** and if for some input, all branches are rejected, the input is also rejected.

A non-deterministic Turing machine can be formally defined as a 7-tuple  $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$  where –

- $Q$  is a finite set of states
- $\Gamma$  is the tape alphabet
- $\Sigma$  is the input alphabet
- $\delta$  is a transition function;  $\delta: Q \times \Sigma \rightarrow P(Q \times \Sigma \times \{\text{Left shift, Right shift}\})$ .
- $q_0$  is the initial state
- $B$  is the blank symbol
- $F$  is the set of final states

#### Introduction of P, NP, NP Complete & NP Hard Problem:

### **P & NP Problems:**

If a problem can be solved in polynomial time, it is said to belong to the **P** class of problems. P-type problems are tractable.

For some intractable problems, you can verify that the solution is correct using a P-type algorithm. For example, you can verify that a given solution to the TSP visits every city. These problems are referred to as Non-deterministic Polynomial problems or NP-type problems. The challenge for programmers is to find a P-type solution to NP-type problems.

### **P Problems:**

As the name says these problems can be solved in polynomial time, i.e.;  $O(n)$ ,  $O(n^2)$  or  $O(nk)$  where k is a constant.

### **NP Problems:**

Some people think NP as Non-Polynomial. But actually, it is Non-deterministic Polynomial time. i.e.; “yes” instances of these problems can be solved in polynomial time by a non-deterministic Turing machine and hence can take up to exponential time (some problems can be solved in sub-exponential but super polynomial time) by a deterministic Turing machine. In other words, these problems can be verified (if a solution is given, say if it is correct or wrong) in polynomial time. So NP class includes all P problems but vice versa is not true for every problem. “Integer Factorization” is a problem which does not belongs to P but belongs to NP.

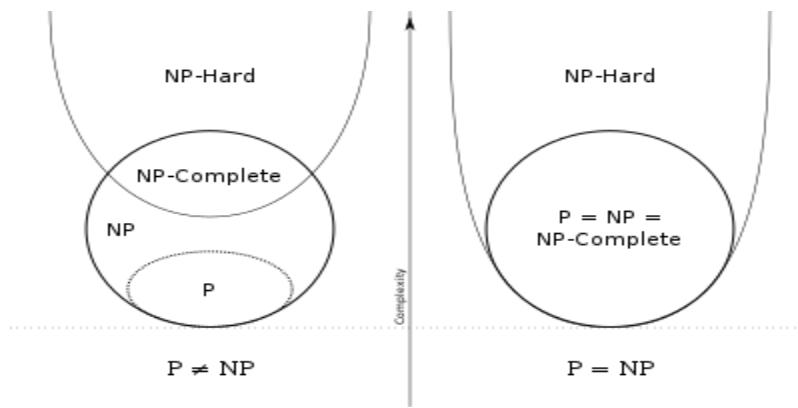
### **NP Complete Problems (NPC):**

Over the years many problems in NP have been proved to be in P (like Primarily Testing). Still, there are many problems in NP not proved to be in P. i.e.; the question still remains whether  $P=NP$  (i.e.; whether all NP problems are actually P problems).

NP Complete Problems helps in solving the above question. They are a subset of NP problems with the property that all other NP problems can be reduced to any of them in polynomial time. So, they are the hardest problems in NP, in terms of running time. If it can be showed that any NPC Problem is in P, then all problems in NP will be in P (because of NPC definition), and hence  $P=NP=NPC$ . So we can say that all NPC class problems belongs NP class.

### **NP Hard Problems (NPH):**

These problems need not have any bound on their running time. If any NPC Problem is polynomial time reducible to a problem X, that problem X belongs to NP Hard class. Hence, all NP Complete problems are also NPH. In other words, if a NPH problem is non-deterministic polynomial time solvable, it is a NPC problem. Example of a NP problem that is not NPC is Halting Problem.



**Figure 5.7: NP complete problem**

From the figure 5.7, it's clear that NPC problems are the hardest problems in NP while being the simplest ones in NPH. i.e.;  $NP \cap NPH = NPC$

Given a general problem, we can say it's in NPC, if and only if we can reduce it to some NP problem (which shows it is in NP) and also some NPC problem can be reduced to it (which shows all NP problems can be reduced to this problem).

Also, if a NPH problem is in NP, then it is NPC

#### Examples of NP Problems:

##### Boolean Satisfiability Problem:

Boolean Satisfiability or simply SAT is the problem of determining if a Boolean formula is satisfiable or unsatisfiable.

Satisfiable: If the Boolean variables can be assigned values such that the formula turns out to be TRUE, then we say that the formula is satisfiable.

Unsatisfiable: If it is not possible to assign such values, then we say that the formula is unsatisfiable.

Examples (as shown in table 5.4):

$F = A \wedge B'$ , is satisfiable, because  $A = \text{TRUE}$  and  $B = \text{FALSE}$  makes  $F = \text{TRUE}$ .

$G = A \wedge A'$ , is unsatisfiable, because:

A	$A'$	G
True	false	False
False	True	False

**Table 5.4: Example of SAT**

Boolean satisfiability problem is NP-complete (This was proved by Cook's Theorem).

## 2-SAT Problem:

- 2-SAT is a special case of Boolean Satisfiability Problem and can be solved in polynomial time.
- To understand this better, first let us see what is Conjunctive Normal Form (CNF) or also known as Product of Sums (POS).

**CNF:** CNF is a conjunction (AND) of clauses, where every clause is a disjunction (OR).

$$F = (A_1 \vee B_1) \wedge (A_2 \vee B_2) \wedge (A_3 \vee B_3) \dots \dots \dots (A_m \vee B_m)$$

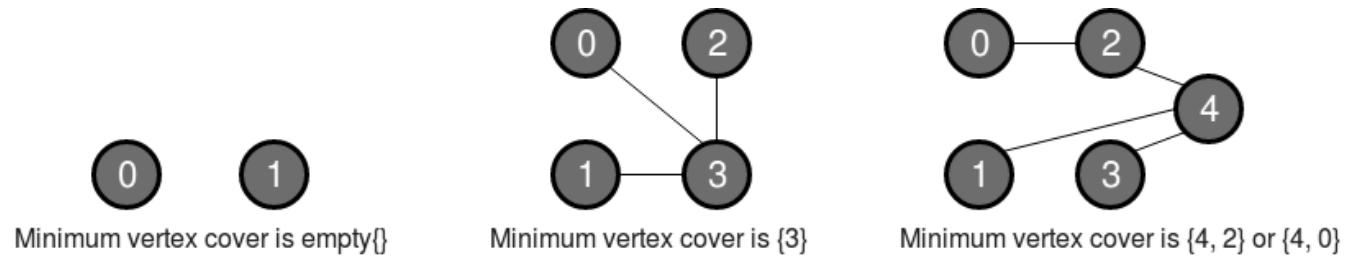
- Now, 2-SAT limits the problem of SAT to only those Boolean formula which are expressed as a CNF with every clause having only **2 terms** (also called **2-CNF**)
- For 2-SAT problem the CNF value is TRUE, if value of every clause is TRUE. Let one of the clauses be  $(A \vee B)$  so we can say  $(A \vee B) = \text{TRUE}$  in following two conditions
  - If  $A = 0$ ,  $B$  must be 1 i.e.  $(A' \Rightarrow B)$
  - If  $B = 0$ ,  $A$  must be 1 i.e.  $(B' \Rightarrow A)$

Thus  $(A \vee B)$  is true equivalent to  $(A' \Rightarrow B) \wedge (B' \Rightarrow A)$

## Vertex Cover Problem:

A vertex cover of an undirected graph is a subset of its vertices such that for every edge  $(u, v)$  of the graph, either ' $u$ ' or ' $v$ ' is in vertex cover. Although the name is Vertex Cover, the set covers all edges of the given graph. Given an undirected graph, the vertex cover problem is to find minimum size vertex cover.

Following are some examples: -



**Figure 5.8: Examples for vertex cover problem**

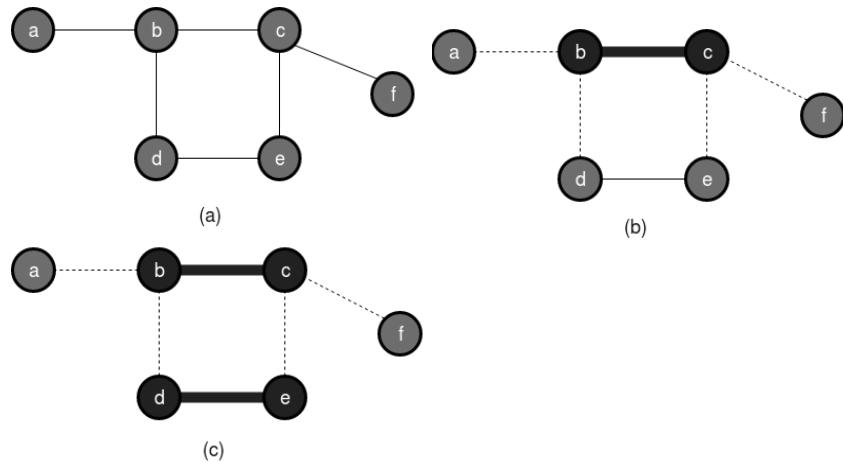
Vertex Cover Problem is a known NP Complete problem, i.e., there is no polynomial time solution for this unless  $P = NP$ . Although There can be an approximate polynomial time algorithm to solve the problem. Following is a simple approximate algorithm.

### Approximate Algorithm for Vertex Cover:

- 1) Initialize the result as {}
- 2) consider a set of all edges in given graph. Let the set be E.
- 3) Do following while E is not empty

- a) Pick an arbitrary edge  $(u, v)$  from set  $E$  and add ' $u$ ' and ' $v$ ' to result
  - b) Remove all edges from  $E$  which are either incident on  $u$  or  $v$ .
- 4) Return result

Below figure 5.9 show execution of above approximate algorithm:



Minimum vertex cover is  $\{b,c,d\}$  or  $\{b,c,e\}$

**Figure 5.9: Execution steps of vertex cover problem**

#### Hamiltonian Cycle Problem:

A Hamiltonian cycle is a cycle in a graph that visits each vertex exactly once. To show Hamiltonian Cycle Problem is NP-complete, we first need to show that it actually belongs to the class NP, and then use a known NP-complete problem to Hamiltonian Cycle.

So does Hamiltonian Cycle Problem  $\in$  NP?

Given:  $Graph G = (V, E)$

Certificate: List of vertices on Hamiltonian Cycle

To check if this list is actually a solution to the Hamiltonian cycle problem, one counts the vertices to make sure they are all there, and then checks that each is connected to the next by an edge, and that the last is connected to the first. It takes time proportional to  $n$ , because there are  $n$  vertices to count and  $n$  edges to check.  $n$  is a polynomial, so the check runs in polynomial time. Therefore Hamiltonian Cycle  $\in$  NP.

Prove Hamiltonian Cycle Problem  $\in$  NP-Complete

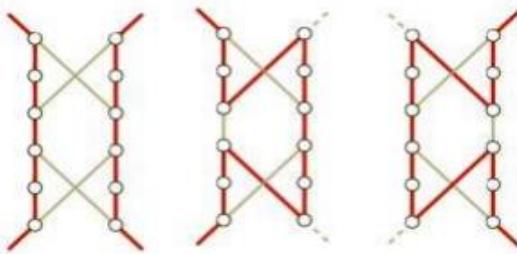
Reduction: Vertex Cover to Hamiltonian Cycle

Definition: Vertex cover is set of vertices that touch all edges in the graph.

Given a graph  $G$  and integer  $k$ , construct a graph  $G'$  such that  $G$  has a vertex cover of size  $k$  if  $G'$  has a Hamiltonian cycle.

Idea: To construct widget for each edge in the graph.

That is  $\forall uv$  in the graph  $G$ , create a widget shown below: -



**Figure 5.10: An Example for Hamiltonian Cycle Graph**

As shown in figure 5.10, there are three ways to traverse a widget; 1. Enter from  $u$ , go somewhere else in the graph, and then come back through the other side i.e.  $v$  2. Enter and Exit through  $u$  3. Enter and Exit through  $v$  Construct  $G'$  for  $G$  (Vertex cover) of size  $k = 2$  with the construction, any graph with a vertex cover, can be used to make a graph with a Hamiltonian Cycle graph. Since creating such a graph can be done under polynomial time, simply replace edges with widgets and make proper connections, we have a reduction from Vertex Cover to Hamiltonian Cycle. This means that finding whether a graph has a Hamiltonian Cycle or not is NP Hard. As we have seen earlier it's also in NP, therefore, Hamiltonian Cycle is an NP Complete Problem.

#### Traveling Salesman Problem:

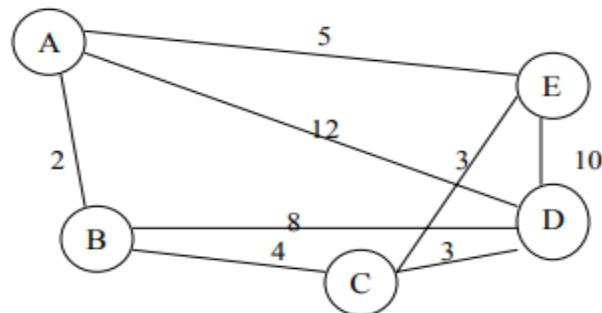
The traveling salesman problem consists of a salesman and a set of cities. The salesman has to visit each one of the cities starting from a certain one (e.g. the hometown) and returning to the same city. The challenge of the problem is that the traveling salesman wants to minimize the total length of the trip.

The traveling salesman problem can be described as follows:

$TSP = \{(G, f, t): G = (V, E)$  a complete graph that contains a traveling salesman tour with cost that does not exceed  $t.\}$

$f$  is a function  $V \times V \rightarrow \mathbb{Z}$ ,  $t \in \mathbb{Z}$ ,

Example: Consider the following set of cities as shown in figure 5.11:



### Figure 5.11: An example of TSP

The problem lies in finding a minimal path passing from all vertices once. For example, the path Path1 {A, B, C, D, E, A} and the path Path2 {A, B, C, E, D, A} pass all the vertices but Path1 has a total length of 24 and Path2 has a total length of 31.

**Theorem:** The traveling salesman problem is NP-complete.

#### Proof:

First, we have to prove that TSP belongs to NP. If we want to check a tour for credibility, we check that the tour contains each vertex once. Then we sum the total cost of the edges and finally we check if the cost is minimum. This can be completed in polynomial time thus TSP belongs to NP.

Secondly, we prove that TSP is NP-hard. One way to prove this is to show that Hamiltonian cycle  $\leq$  TSP (given that the Hamiltonian cycle problem is NP-complete). Assume  $G = (V, E)$  to be an instance of Hamiltonian cycle. An instance of TSP is then constructed. We create the complete graph  $= (V, \leq P G' E')$  where  $E' = \{(i, j) : i, j \in V \text{ and } i \neq j\}$ . Thus, the cost function is defined as:

$$t(i, j) = \begin{cases} 0 & \text{if } (i, j) \in E, \\ 1 & \text{if } (i, j) \notin E. \end{cases}$$

Now suppose that a Hamiltonian cycle  $h$  exists in  $G$ . It is clear that the cost of each edge in  $h$  is 0 in  $G'$  as each edge belongs to  $E$ . Therefore,  $h$  has a cost of 0 in  $G'$ . Thus, if graph  $G$  has a Hamiltonian cycle then graph  $G'$  has a tour of 0 costs.

Conversely, we assume that  $G'$  has a tour  $h'$  of cost at most 0. The cost of edges in  $E'$  are 0 and 1 by definition. So, each edge must have a cost of 0 as the cost of  $h'$  is 0. We conclude that  $h'$  contains only edges in  $E$ .

So, we have proven that  $G$  has a Hamiltonian cycle if and only if  $G'$  has a tour of cost at most 0. Thus, TSP is NP-complete.

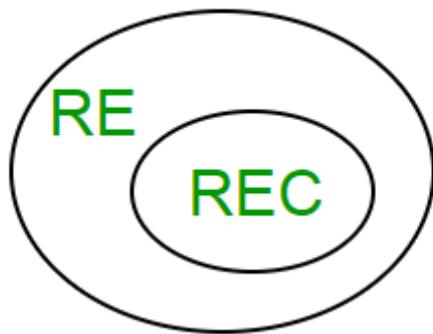
#### Recursive & Recursively Enumerable Language:

A Turing Machine may

- Halt and accept the input
- Halt and reject the input, or
- Never halt/loop

**Recursive Enumerable or Type-0 Language:** RE languages or type-0 languages are generated by type-0 grammars. A RE language can be accepted or recognized by Turing machine which means it will enter into final state for the strings of language and may or may not enter into rejecting state for the strings which are not part of the language. It means TM can loop forever for the strings which are not a part of the language. RE languages are also called as Turing recognizable languages.

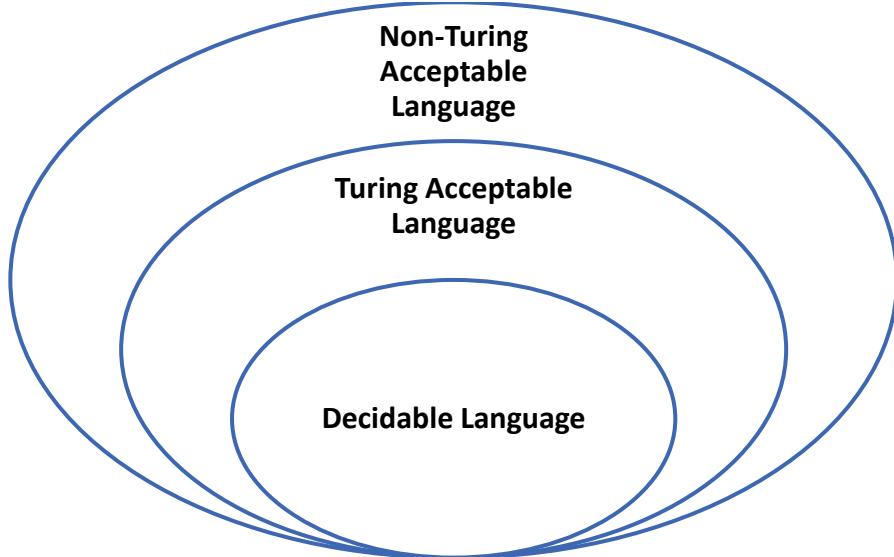
**Recursive Language:** A recursive language (subset of RE) can be decided by Turing machine which means it will enter into final state for the strings of language and rejecting state for the strings which are not part of the language. e.g.;  $L = \{a^n b^n c^n \mid n \geq 1\}$  is recursive because we can construct a Turing machine which will move to final state if the string is of the form  $a^n b^n c^n$  else move to non-final state. So, the TM will always halt in this case. REC languages are also called as Turing decidable languages.



**Figure 5.12: Relationship between RE & REC**

#### **Language Decidability:**

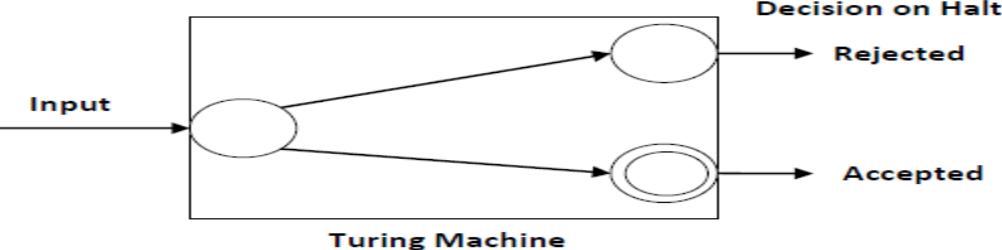
A language is called Decidable or Recursive if there is a Turing machine which accepts and halts on every input string  $w$ . Every decidable language is Turing-Acceptable.



**Figure 5.13: Language Decidability**

A decision problem  $P$  is decidable if the language  $L$  of all yes instances to  $P$  is decidable.

For a decidable language, for each input string, the TM halts either at the “accept” or the “reject” state as depicted in the following diagram:



**Figure 5.14: Decidable Language**

**Example 1:** Find out whether the following problem is decidable or not: Is a number 'm' prime?

Solution:

Prime numbers = {2, 3, 5, 7, 11, 13, .....}

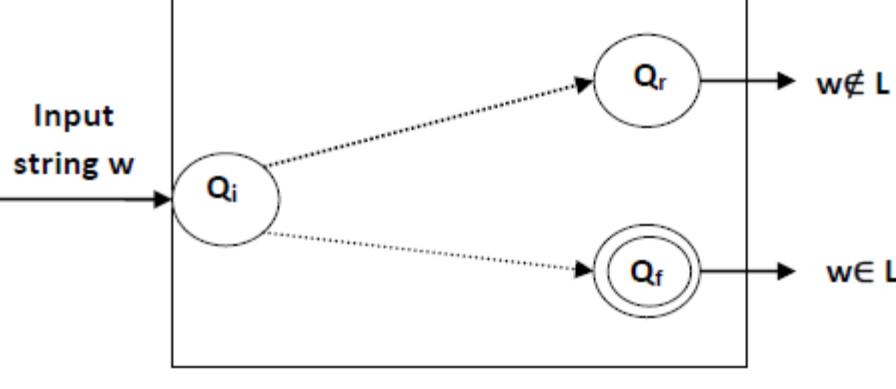
Divide the number 'm' by all the numbers between '2' and ' $\sqrt{m}$ ' starting from '2'. If any of these numbers produce a remainder zero, then it goes to the "Rejected state", otherwise it goes to the "Accepted state". So, here the answer could be made by 'Yes' or 'No'.

Hence, it is a decidable problem.

**Example 2:** Given a regular language L and string w, how can we check if  $w \in L$ ?

Solution:

Take the DFA that accepts L and check if w is accepted

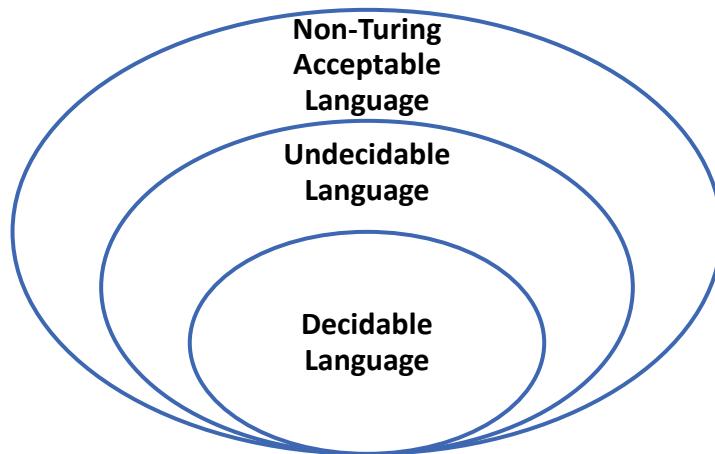


**Figure 5.15: Example of Language Decidable**

**Note:**

1. If a language  $L$  is decidable, then its complement  $L'$  is also decidable.
2. If a language is decidable, then there is an enumerator for it.

### **Undecidable Language:**



**Figure 5.16: Undecidable Language**

For an undecidable language, there is no Turing Machine which accepts the language and makes a decision for every input string  $w$  (TM can make decision for some input string though). A decision problem  $P$  is called “undecidable” if the language  $L$  of all yes instances to  $P$  is not decidable. Undecidable languages are not recursive languages, but sometimes, they may be recursively enumerable languages.

### **Example:**

- The halting problem of Turing machine
- The mortality problem
- The mortal matrix problem
- The Post correspondence problem, etc.

### **Turing Machine Halting Problem:**

**Input:** A Turing machine and an input string  $w$ .

**Problem:** Does the Turing machine finish computing of the string  $w$  in a finite number of steps? The answer must be either yes or no.

**Proof:** At first, we will assume that such a Turing machine exists to solve this problem and then we will show it is contradicting itself. We will call this Turing machine as a Halting machine that produces a ‘yes’ or ‘no’ in a finite amount of time. If the halting machine finishes in a finite amount of time, the output comes as ‘yes’, otherwise as ‘no’. The following is the block diagram of a Halting machine:

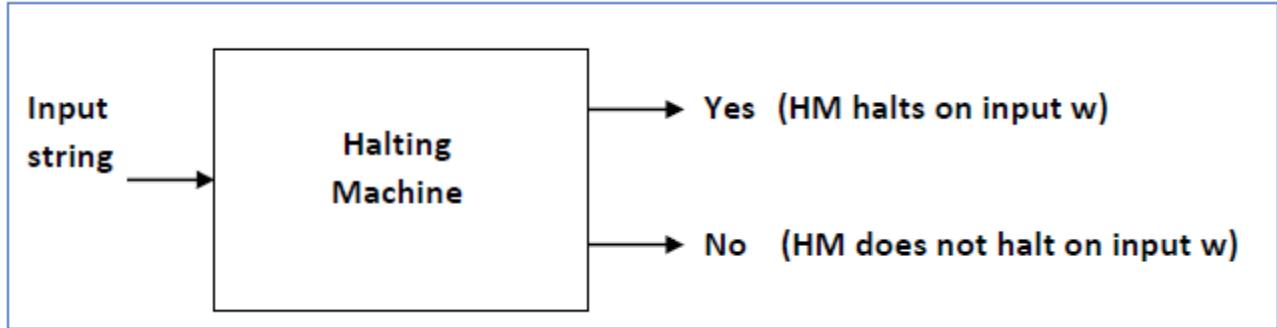


Figure 5.17: Turing Machine Halting Problem

Now we will design an inverted halting machine (HM)' as:

- If H returns YES, then loop forever.
- If H returns NO, then halt.

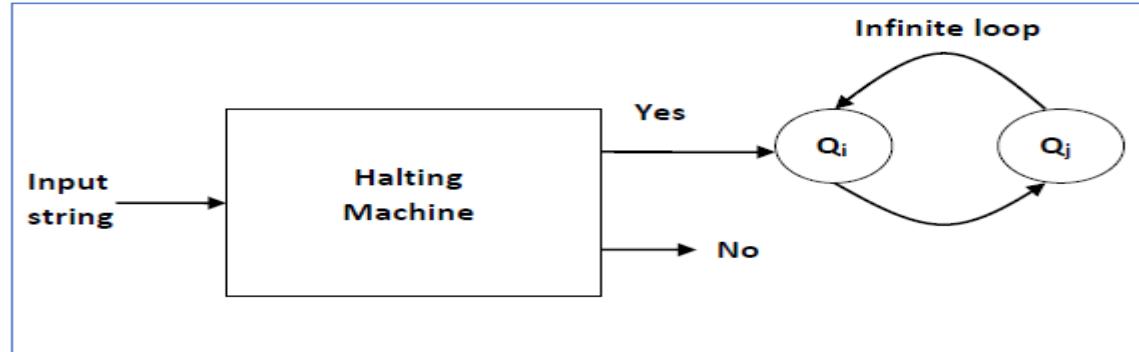


Figure 5.18: Inverted Halting Machine

Further, a machine (HM)2 which input itself is constructed as follows:

- If (HM)2 halts on input, loop forever.
- Else, halt.

Here, we have got a contradiction. Hence, the halting problem is undecidable.

Chameli Devi Group of Institutions, Indore

Department of Artificial intelligence and Data Science

Subject Notes

AD 505- Internet and Web Technology

UNIT-I

Syllabus: Introduction: Concept of WWW, Internet and WWW, HTTP Protocol: Request and Response, Web browser and Web servers, Features of Web 2.0, Web Design: Concepts of effective web design, Web design issues including Browser, Bandwidth and Cache, Display resolution, Look and Feel of the Website, Page Layout and linking, User centric design, Sitemap, Planning and publishing website, Designing effective navigation.

Internet overview

The Internet is a giant network of networks.

- A network may include PCs, and other devices like servers or printers.
- A network is connected through a communication channel.
- Early research was performed by the US Department of Defense in 1962. This research group established ARPAnet (Advanced Research Project Agency) in order to connect the US Defense Department network.
- Original aim was to create a network that would allow users of a research computer at one university to be able to ‘talk to’ research computers at other universities.
- A side benefit of ARPAnet’s design was that, because messages could be routed or re-routed in more than one direction, the network could continue to function even if parts of it were destroyed in the event of a military attack or other disaster.
- The users of the Internet took a direction of their own.

Figure 1.1: Internet and World Wide Web

### 1.1 World Wide Web (WWW)

In the course of life, when people say “Internet”, most of the time they actually refer to the World Wide Web or the WWW. The www is the collection of all the information that is available in the Internet. So, all the text, images, audio, videos online all this forms the www. Most of this information is accessed

through websites and we identify websites by their domain names. There is huge amount of information available in the www. Only a tiny part of this information is searchable through popular search engines like Google. However, most of the information lies in the Deep Web and Dark Web. WWW uses http protocol to access the information from various servers. Information is sent as web pages which are organized in the form of websites. Various web pages are interlinked with each other through hyperlinks. Web pages and other pieces of information in WWW are identified by their Uniform Resource Locator (URL) address.

Table 1.1: Difference between Internet and WWW

INTERNET	WWW
1. Internet originated sometimes in late 1960s.	English scientist Tim Berners-Lee invented the World Wide Web in 1989.
2. Nature of Internet is hardware.	Nature of www is software.
3. Internet consists of computers, routers, cables, bridges, servers, cellular towers, satellites etc.	WWW consists of information like text, images, audio, video.
4. The first version of the Internet was known as ARPANET.	In the beginning WWW was known as NSFNET.
5. Internet works based on Internet Protocol (IP)	WWW works based on Hyper Text Transfer Protocol (HTTP).
6. Internet is independent of WWW.	WWW requires the Internet to exist.
7. Internet is superset of WWW.	WWW is a subset of the Internet. Apart from supporting www, the Internet's hardware infrastructure is used for other things as well (e.g. FTP, SMTP).
8. Computing devices are identified by IP Addresses.	Information pieces are identified by Uniform Resource Locator (URL).

## 1.2 HTTP Protocol

What is http: HTTP full form Hyper Text Transfer Protocol used mainly to access data on the World Wide Web. HTTP is a Server and Client communication Protocol, which is primarily set of rules for formatting and transferring webpage data (text, images, video, and Multimedia files) over the World Wide Web. This is the Protocol used to create communication between Web Servers and Web Users. HTTP is an application layer Protocol that works on the top of the TCP/IP suite of Protocols.

The features of the http protocol are as follows:

- Http is connectionless: The http client, i.e., the web browser makes an http request and waits for the server to respond. Now, it is the task of the server to process the request made by the client. So, after

processing, the server gives the response to the client. After receiving the response, the client disconnects the connection.

- Http is media independent: Here, media independent means that any data can send. Also, we must mention the content type as per the requirement of the client and the server.
- Http is stateless: Http is a stateless protocol. Only during the current request, the client and the server know about each other and when the connection disconnects, both client and the server forgets about each other. Due to this nature, both the client and the server do not retain the information between the different requests processed.

#### Basic architecture client/server

As we know, it is a client/server-based architecture. So, it makes use of a request/response protocol. In this, web browsers, search engines, act as the client of the system and the web server acts as the server of the system.

This system is completely explained through the diagram also.

Figure 1.2: Basic architecture of client/server model

**Http Client:** The http client makes the request in the form of a request method followed by the message body over a TCP/IP connection.

**Http Server:** The request sent by the client, is responded by the server in the form of a status line followed by the other necessary information with the message body.

#### 1.3 Request and Response

The operation of Hypertext Transfer Protocol (HTTP) involves the communication between a Hypertext Transfer Protocol (HTTP) client application (Usually web browser) and a Hypertext Transfer Protocol (HTTP) server application (Web servers like IIS). Hypertext Transfer Protocol (HTTP) uses Transmission Control Protocol (TCP) as the Transport Layer Protocol at Well Known port number 80. Once the TCP connection is established, the two steps in Hypertext Transfer Protocol (HTTP) communication are:

- 1) **HTTP Client Request:** Hypertext Transfer Protocol (HTTP) client sends a Hypertext Transfer Protocol (HTTP) Request to the Hypertext Transfer Protocol (HTTP) Server according to the HTTP standard, specifying the information the client like to retrieve from the Hypertext Transfer Protocol (HTTP) Server.
- 2) **HTTP Server Response:** Once the Hypertext Transfer Protocol (HTTP) Request arrived at the Hypertext Transfer Protocol (HTTP) server, it will process the request and creates a Hypertext Transfer Protocol (HTTP) Response message. The Hypertext Transfer Protocol (HTTP) response message may contain the resource the Hypertext Transfer Protocol (HTTP) Client requested or information why the Hypertext Transfer Protocol (HTTP) request failed.

**Multiple Host Name Support:** HTTP/1.1 supports specifying a Hostname in header. This allows running multiple websites using a single IP address helps preventing the depletion of IPv4 addresses. Any combination of IP address, Port number and Hostname can be used to identify a website.

Figure 1.3: Hypertext Transfer Protocol (HTTP) Request and Response

#### 1.4 Web browser and Web servers

##### What is Web Server?

Web server is a computer system, which provides the web pages via HTTP (Hypertext Transfer Protocol). IP address and a domain name are essential for every web server. Whenever, you insert a URL or web address into your web browser, this sends request to the web address where domain name of your URL is already saved. Then this server collects all information of your web page and sends to browser, which you see in form of web page on your browser.

All the communication between client (web browser) and server takes place via HTTP. The main aim of a web server is to work as a medium for storing, processing, and granting access to web pages to the users to clients of the internet. As it is already mentioned, HTTP that is a system for collaboration, distribution and hypermedia information system is used for the channel of communication between server and client. HTTP then delivers the pages in the format of HTML documents that comprises the information in the shape of scripts, style sheets, videos and images in addition to simple text-based content. The process of getting the desired information starts by the web browser that sends the request for the specific page using the HTTP and then the web server responds to that request by either providing the requested content or giving the error message.

Web server is designed for the purpose of providing information; however, it also receives the data from the clients as well. Uploading of files or data, submitting web forms, etc. are the common examples of giving content to the web server. The four highlighted features of web server are virtual hosting, large file support even greater than 2GB, bandwidth throttling, and server-side scripting to generate the dynamic web pages. The web server can handle the limited load normally from two to 80,000 connections at once. This is not common or everyone as by default most of the web server supports the load from 500 to 1000 connections for a single IP address.

##### List of popular web servers

- Apache HTTP server
- Internet Information Server (IIS)
- NGINX (EN-jin-EKS)
- Oracle Iplanet web server
- Caddy

- Tomado
- Lighttpd

### What is Web Browser?

Web browser is a client, program, software, or tool through which we sent HTTP request to web server. The main purpose of web browser is to locate the content on the World Wide Web and display in the shape of web page, image, audio, or video form. You can call it a client server because it contacts the web server for desired information. If the requested data is available in the web server data, then it will send back the requested information again via web browser. Microsoft Internet Explorer, Mozilla Firefox, Safari, Opera and Google Chrome are examples of web browser, and they are more advanced than earlier web browser because they are capable to understand the HTML, JavaScript, AJAX, etc.

Now days, web browser for mobiles are also available, which are called micro browser. The web browser is used as a medium for accessing information from any web server in files in a file system or in private networks. The process of getting the information from the web server starts by the web browser. The modern-day web browser supports the display in almost all digital media formats like images, scripts, videos, images, and text. The main aim of the web browser is to send the request to the web server for any specific data and present them in any digital media format. However, the features of the web browser can vary from one to another. Most of the latest web browser supports the various other functions in the shape of email, internet relay chat, Usenet news, etc.

List of popular web browsers are as follows:

- Firefox (Mozilla)
- Internet Explorer (Microsoft)
- Edge (Microsoft)
- Opera
- Safari
- Netscape
- Mosaic

### 1.5 What is a Web 2.0 technology?

- When it comes to define web 2.0. The term means such internet applications which allow sharing and collaboration opportunities to people and help them to express themselves online.
- It's a simply improved version of the first world wide web, characterized specifically by the change from static to dynamic or user-generated content and the growth of social media. The concept behind Web 2.0 refers to rich web applications, web- oriented architecture and social web. It refers to changes in the way's web pages are designed and used by the users, without any change in any technical specifications.

Table 1.2: Comparison between Web 1.0 and Web 2.0

Web 1.0	Web 2.0
Mostly Read-Only	Widely Read-Write
Company Focus	Community Focus
Home Pages	Blogs/Wikis
Owning Content	Sharing Content
Web Forms	Web Applications
Directories	Tagging
Page Views	Cost Per Click
Banner Advertising	Interactive Advertising
Britannica Online	Wikipedia

- Major features of Web 2.0 allow users to collectively classify and find dynamic information that flows two ways between site owner and site user by means of evaluation, comments and reviews. Site users can add content for others to see. Web 2.0 sites provide APIs to allow automated usage by an app or mash up like it provides location metadata that can be processed by a simple browser tool.

Figure 1.4: Web 2.0 Components and Groups

### 1.6 Concepts of effective web design

Web design is a concept of planning, creating, and maintaining websites. The very process of using creativity to design and construct a website and updating it regularly to incorporate changes is also referred to as web designing. Besides the creation and updating, this concept also involves taking care of the user interface, the architecture of information present, the layout, the colors, content, navigation ergonomics, as well as the designs of the various icons. Some other areas in web design include search engine optimization, user experience designs, standardized codes, graphic design, as well as interface design.

### Basic elements of a good web design

In order to come up with a good web design and an effective visual and technical appeal of a website, there are some elements that must be incorporated. To know more about these elements, you can go through the following given points:

**Shape** – On most websites and web pages, the shapes used are squares or rectangular, but they don't necessarily have to be. Shapes are responsible for the creation of an enclosed boundary in the overall design, and you can experiment with any shape you seem suitable. It can either be a geometric shape or any other abstract shape that fits in the design.

**Texture** – Texture is one element that can help provide your website with a feeling of a surface beneath. There are several types of textures that you can incorporate, and some of them include natural textures and artificial textures. This element must be used in such a way that it brings out the content given on the website and makes it look more appealing.

**Direction** – Direction is the element of a web design which is responsible for lending it movement or motion. A good web design is one which automatically makes our eyes move from one corner or content of the website to another, according to relevance or hierarchy.

**Color** – Another basic element of a good website is the use of color. A black and white website may work for certain niches like photography websites, but it is always better to raise the appeal of a website using colors in a creative way. The colors are added in the later stage and not during the designing.

## Web design Principles

Web design is not only about how the website looks and feels but is also a lot about how it works and responds. When web designers work on a website, they incorporate not just those elements that add a visual appeal to it but also try to make it highly responsive, functional, quick and useful. In order to create a highly usable and effective website, designers follow certain principles that act as thumb rules or standard points to keep in mind. The following are the various principles of an effective web design:

**Size hierarchy** – As the name suggests, size hierarchy is the kind of hierarchy in which the most important content or image is of the largest size on a webpage, followed by the second most important content or image in the second largest size and so on. The distinction in sizes should be such that a visitor would view the items in the order of importance, and the pecking order of things is obvious.

**Content hierarchy** – Besides the hierarchy of size, which is one of the best ways to create the order of importance, another way you can incorporate this principle is by creating a hierarchy of content. You can place content in such a way that the human eye first travels to the content that is most important, for example, the business's objective or purpose and then moves to the less important content blocks in a hierarchical order.

## Accessibility

Another highly important principle that must not be ignored when designing a web page or website is the accessibility of it. When a visitor enters the website, he/she must be able to access each bit of information in the easiest manner. This means that the text must be legible, the colors must not be

harsh on the eyes and the background must not overpower the content, etc. To make the website accessible to everyone, you can follow some of the following points:

Typefaces – Make sure you select a font type and font size which is readable to all and is not too fancy for some to access or understand. For example, Fonts like Verdana, Times New Roman, Arial, etc. are simple fonts that almost everyone can easily read online. Similarly, the font size that works the best is 16 px but you can be a little flexible with it.

Colors – As far as the user experience is concerned, your color scheme and contrast must be well thought of and should be able to create visual harmony and balance. It is always better to choose contrasting colors for the background and written content so that it can be easily read. Choose a darker text color and a lighter background shade so that the result is easy to the eyes. Extra bright colors must be used sparingly.

Images – Do you know that the human mind perceives and processes images a lot faster than text? Well, it is thus a good idea to choose and place the right images on your web pages to communicate with the audiences in a better way. Make sure they are high-quality images and are suitable for your purpose.

#### Communication and content

Everyone who visits your website is looking for some or the other kind of information or content, and thus it is very important for you to communicate with them clearly and in an engaging manner. Your information must be compelling, easy to read and easy to process. Communication is not just about providing written information but also about offering images, infographics, and another form of media such as videos and audio pieces. Web design takes into consideration a concept called ‘Visible language’.

The following are some of the other things that can be considered as a part of a simple design:

Grid-based layout – To avoid a messy structure or appearance of the website, you must opt for a grid-based layout in which content is divided into columns, boxes, and different sections.

F-pattern design – It is a fact that the human eye scans screens in an ‘F’ pattern. Thus, it is a good idea to design a web page or website in a way that complements the natural reading behavior of the visitors.

Conventional designs – Conventional designs are not always boring and rather work well as far as visitor response or likeability is concerned. They add a hint of trust, reliability as well as brand credibility.

#### Regular testing

Test Early and Test Often or ‘TETO’ is another web design principle that all designers and website owners must consider. Conducting usability tests every now and then provide important results and insights into many kinds of problems and complications related to a website layout or aspects of design. What happens is that websites often tend to get into certain issues and by not testing them often, they may create issues that could be driving visitors away. Websites constantly need upgrades and updates to maintain the visitor footsteps and customer interest.

The following are some points you must keep in mind to test your webpage or website:

- Testing one user at the beginning of a project is better than testing 50 users towards the end as it makes way for improvements that could prove important for driving traffic and increasing sales. Also solving errors during process is least expensive rather than later.
- Another point to be kept in mind is that testing is an iterative process which means that designers must first design, then test, then fix and then test again. There is a strong chance that there may be some problems that were not solved the first time because of diversion of attention to other major issues.

## 1.7 Web design issues including Browser

1. There's no clear path. You want to extend a warm welcome to your visitors. Give them an easy way in and through. Too much competition for attention is a turn off. When you provide too many options, the functional result is no options.
2. Outdated design. Your site was state of the art in 2009. It's got a header, a couple of sidebars, and a big chunk of information running down the center. It looks like it's nearly a decade old. Time to refresh with current design thinking. The layout of a page has evolved over the past decade. These days the best sites break up content into smaller, digestible bits.
3. Overused stock images and icons. If visitors see the same image on multiple sites, it erodes trust.
4. Too many textures and colors. You are trying to add interest, but you just add clutter. Limit colors and fonts. Maintain a thematic color scheme. For professional sites, try to limit the variety of fonts to three or fewer.
5. Design for the wrong reasons. Always begin by identifying your target audience and customizing design and content. You may want your site to look "modern" or like another site you've seen, but if you haven't checked in with what your audience needs and wants, you can fail miserably.
6. Cute that doesn't cut it. When your links have adorable, witty names, the experience gets tired fast. Links that don't make much sense are not user friendly and won't ingratiate you with your visitors. Be practical and basic when naming links. Make it easy for people. Design for multiple visits. A rotating banner is cool the first time, and maybe the second, but at some point, it's just a stale eyesore.  
Monotonous calls to action.
7. Your site isn't optimized for mobile. You shouldn't need to be reminded of this, but numbers don't lie. Mobile is overtaking desktop. It's increasingly likely that your visitors see your site on a tiny screen. If they must pinch and stretch to read, they'll find a better source of information. Be sure to test your site on smart phone and tablet.
8. You play hard to get. If you want customers to find you, make sure your address, phone number and hours of operation are easily accessible on your site. Too often, that information is hidden or completely absent.

Main technical issues in web design are as follows:

Before you create a website, you should consider the technical issues relating to web design, specifically:

- Browser compatibility
- Screen resolutions
- Web technologies
- Internet speed

## 1.8 Bandwidth and Cache

### Bandwidth

The short answer is that caching saves money. It saves time as well, which is sometimes the same thing if you believe that “time is money.” It provides a more efficient mechanism for distributing information on the Web. Consider an example from our physical world: the distribution of books. Specifically, think about how a book gets from publisher to consumer. Publishers print the books and sell them, in large quantities, to wholesale distributors. The distributors, in turn, sell the books in smaller quantities to bookstores. Consumers visit the stores and purchase individual books. On the Internet, web caches are analogous to the bookstores and wholesale distributors.

The analogy is not perfect, of course. Books cost money; web pages (usually) don’t. Books are physical objects, whereas web pages are just electronic and magnetic signals. It’s difficult to copy a book, but trivial to copy electronic data.

The point is that both caches and bookstores enable efficient distribution of their respective contents. An Internet without caches is like a world without bookstores.

- Client requests must exhibit locality of reference.
- The cost of caching must be less than the cost of direct retrieval.

We can intuitively conclude that the first requirement is true. Certain web sites are very popular. Classic examples are the starting pages for Netscape and Microsoft browsers. Others include searching and indexing sites such as Yahoo! and Altavista. Event-based sites, such as those for the Olympics, NASA’s Mars Pathfinder mission, and World Cup Soccer, become extremely popular for days or weeks at a time. Finally, every individual has a few favorite pages that he or she visits on a regular basis.

Let’s take a closer look at the three primary benefits of caching web content:

- To make web pages load faster (reduce latency)
- To reduce wide area bandwidth usage
- To reduce the load placed on origin servers

### Cache

Another reason to utilize web caches is bandwidth reduction. Every request that results in a cache hit saves bandwidth. If your Internet connection is congested, installing a cache is likely to improve performance for other applications (e.g., email, interactive games, streaming audio), because all your network applications compete for bandwidth. A web cache reduces the amount of bandwidth consumed by HTTP traffic, leaving a larger share for the others. It is also correct to say that a web cache increases your effective bandwidth. If your network supports 100 users without a cache, you can probably support 150 users with a cache.

Even if your Internet connection has plenty of spare bandwidth, you may still want to use a cache. In the United States, Internet service is typically billed at flat rates. Many other countries, however, have usage-based billing. That is, you pay only for the bandwidth you use. Australia and New Zealand were among the first countries to meter bandwidth usage.

### 1.9 Display resolution

Web page resolution is a big deal. Many sites that teach web design have written about it and depending upon who you believe, you should design pages for the lowest common denominator (640x480), the most common resolution (800x600), or the most cutting edge (1280x1024 or 1024x768).

#### Facts about Screen Resolutions

- While 640x480 is not as common as it used to be, this resolution is still around. Older computers, laptops with smaller screens, netbooks, and people who need larger fonts all use this resolution. Even if you choose not to design your page to this resolution you should test your site at this resolution.
- Many website design guides recommend designing websites for 800x600 resolutions. While this resolution is more common on the web at large, this may not be the case for your customers. If you're planning on redesigning your website, take a few weeks to analyze your browser statistics to determine the most common resolutions used by your customers.
- Screens are getting larger and 1024x768 is the other popular size to design for because many designers have monitors that support this natively. But this resolution can be hard to read for many people. A 14-inch laptop monitor might support 1024x768, but the text is virtually unreadable. And laptops are very popular.
- Mostly you'll see these larger resolutions on desktop computers, or high-end laptops. But often at these larger sizes, the customers aren't browsing full screen. So, designing a site that's wider than 1000 pixels is going to cause horizontal scrollbars on most screens.

### 1.10 Look and Feel of the Web

The success of your website layout lies in easy navigation, user friendliness and high functionality. Here we discuss 10 Tips to improve your website and its look and feel:

1. Your website should be fast: One of the main reasons for the downfall of a website is when it loads too slowly. Even if your site takes 20 seconds to load- that means a lifetime on the internet- and most people don't have the time and patience to wait for that long. Understand that wild graphics and animations are traffic killers and only use them if you really have to. Make sure you design your website so that everyone can view- not only the ones with high-speed internet connection. If your website takes more than 10 seconds to load, chances are that you may be losing more than half of your site's traffic.
2. Easy navigation: This is one of the most important and simple tips to improve a website. Statistics show that websites lose more than 50% of site visitor's every time they must click on a link to find a page. Make sure that all your web pages are not more than five clicks from the home page.
3. Make your website mobile-friendly: More than half the population owns a mobile now and half of that percentage uses Smartphone's, so there are chances that your potential clients will land on your site from mobile browsing. But it's very tough to browse a website that is not optimized for a mobile screen. This is another important factor for improving a website and there are plenty of ways to create a mobile version for your current website.
4. Your website should match your company branding: If you update your company logo or colors, make sure it matches your online presence to create brand awareness and make it easy for customers to recognize your company online.
5. Never overdo Pop-ups: A lot of webmaster overuse pop-ups thinking that it will help customers make more purchases but that's not the case. You'll probably lose a potential client if you have pop-ups every time they open or close a window or click on a link. Use pop-ups only to offer customers and site visitor's something of high value and something that they will be really interested in.
6. Make sure to use appropriate colors and fonts: A simple and clean looking website is creative enough to improve the look and feel of your site. Throwing all kinds of colors in your site is overwhelming and can put site visitors off. Remember, nothing is easier on the eye than black text on a white background. Use small amounts of colors and have your website looking elegant which is more appealing to the eye. Also make sure to have fonts that can be easily read on any device or browser.
7. Incorporate Social Media widgets: Including social media widgets will help your business and website become available to customers whenever they need to check any updates about your site. The widgets shouldn't be the most prominent feature in your site, but they shouldn't be hard to spot either.
8. Use images to your advantage: Images play an important role when people search for something on the internet. A lot of web users are simply attracted to websites that have optimized images and clean, well photographed images can enhance your business or service much more than words can.
9. Interact with customers: Interacting with customers, both present and potential is very important in any business. Consider supporting interaction among the web readers and putting a comment box where they can easily exchange suggestions.
10. Never forget your purpose: The most important factor to improve your website remember your purpose of having a website. Each page of your site should relate to the purpose of having a website. All your images, graphics, colors, and links should help you achieve what you wanted from having a website

in the first place. If you want more sales and to promote your business or service, you must go for a sales-oriented website instead of one that is just nice-looking.

## 1.11 Sitemap

A sitemap is a resource that the client and the web design team can refer to throughout the project. It's a handy tool that displays the relationships between your site's pages and its content elements. Ultimately, building a website without a sitemap is like building a house without a blueprint.

Why a site map is built before web design starts?

These are four reasons why it is important to know what is sitemap and what does it accomplish?

- Puts everyone on the same page:

A web-designing project needs participation of many people. It includes business owners, web designers and account managers etc. By outlining a solid, agreed-upon plan at the beginning of a project, the entire process tends to flow better.

- Establishes the site's goals and purpose:

On the internet today, there are far too many websites that provide poor user experience. These difficult-to-navigate sites typically lack a sitemap that outlines its goals and purposes. Once a sitemap is created, the pieces of a website are more easily plugged into place.

- Prevents creation of duplicate content:

In this duplicate content is another preventable web design mistake that can hurt your search ranking. Map out a sitemap beforehand, you will be able to avoid redundancy by determining early on where all your content should be in same.

- Sets up a clear conversion funnel path:

If you want to generate leads with digital marketing, you must have a clear conversion path. To achieve the best results, this process should start in the sitemap phase. Then, you can make sure your visitors are properly directed from your calls-to-action on each web page.

## 1.12 Page Layout and linking

Page layout refers to the arrangement of text, images, and other objects on a page. The term was initially used in desktop publishing (DTP), but is now commonly used to describe the layout of web-pages as well. Page layout techniques are used to customize the appearance of magazines, newspapers, books, websites, and other types of publications.

The page layout of a printed or electronic document encompasses all elements of the page. This includes the page margins, text blocks, images, object padding, and any grids or templates used to define positions of objects on the page. Page layout applications, such as Adobe InDesign and QuarkXpress, allow page designers to modify all of these elements for a printed publication. Web

development programs, such as Adobe Dreamweaver and Since there are many applications that create customized page layouts, there is also a specific file format category for page layout file types. These files are similar to word processing documents but may contain additional page formatting information and other types of visual content.

### Linking within a Web Page

Clicking a link usually loads a web page into the browser window. However, what if you want to link not only to a web page but to a specific spot on that page? You'll see this frequently on long web pages, where links at the top of the page let visitors jump down to specific content lower on the page. You can create in-page links two ways: by using a named-anchor or by adding an ID to the target section. The named-anchor method has been around since the earliest days of the Web; it uses a special type of link. With the ID technique, you give the destination spot a unique ID and then link to that ID. Although this method is newer, it works with all current web browsers. You'll learn about both methods below.

### 1.13 User-Centered Design Basics

The User-centered design (UCD) process outlines the phases throughout a design and development life cycle all while focusing on gaining a deep understanding of who will be using the product. The international standard 13407 is the basis for many UCD methodologies. It's important to note that the UCD process does not specify exact methods for each phase.

User-centric design process goes through six phases:

- Specify the use context and users' needs;
- Specify business requirements;
- Build design solutions from rough concept to finished design;
- Evaluate designs with usability testing;
- Implementation — develops and delivers the product;
- Deployment — the final product is evaluated, as consumer needs change.

### 1.14 Planning and publishing website

Interesting, sharable content to your website on a regular basis is a key market strategy that successful companies have been using for years. Content planning and publishing creates an upward spiral of online reputation building, industry credibility, and strong Google ranking, which all lead to business growth. Following process describes the website planning and publishing process:

#### 1. Set your purpose and goals

What is the purpose of your website? Is it to gain publicity for your business?

It's important to identify your website's purpose, as well as your target audience. You should also define your goals. How many visitors do you expect per month?

## 2. Create a budget

Whether you're an established, mid-sized organization or a fledgling start-up, you should always set a budget for your website expenses. This will probably include funds for web design, programming, and web hosting.

## 3. Assign roles

- Company stakeholders
- Web developer
- Content writer and/or editor
- HTML/CSS professional

## 4. Create a content strategy

What kind of content will you be displaying on your website? Content is basically anything that gives your visitors information.

## 5. Structure your website

Decide what pages you'll be using and what features will be on each one. Most websites have an About and Contact page, but the pages you use should meet your business' needs.

## 6. Create a mock-up

A page mock-up, also known as a wireframe, is essentially the outline of your website (with the initial design being the first draft). Usually created in Photoshop or Fireworks, you don't have to put too much detail into your mock-

## 7. Start designing

The importance of good web design can't be stressed enough. Good website design includes both usability and aesthetics. An ugly website will drive away visitors, as will a website that's difficult to navigate.

## 8. Test it out

Testing is important for getting out bugs out and catching details that you might have missed initially. Make sure your website shows up the way you want it to in all browsers, including Chrome, Firefox, Internet Explorer, and mobile web browsers like Safari and Opera Mini. Test it on your cell phone.

## 9. Maintain your site

Once your site is launched, the work isn't over. A website is an ongoing entity that continuously represents your company, so maintenance is very important. Monitor your analytics software to see how your website is performing with the public. Keep an eye on metrics like your number of unique visitors, bounce rate, and which pages are most popular on your website.

## 1.15 Designing effective navigation

Navigation is all about the links you provide to enable people to move between the different parts and pages of your website. Your navigation needs to address four key challenges:

Visitors won't know what's on your website unless you tell them. Entire sections of your site could be overlooked if you tell people don't know they're there.

- People don't have the patience or commitment to try to figure out how to use your website. They just want to be able to understand it immediately and want to intuitively know how to get around it. If they can't work out where to find something they want, they'll probably leave your website altogether.
- People don't know how your website is structured unless you tell them. They want to feel that they understand where they are in the whole site, not to feel like they're bouncing between pages aimlessly.
- Visitors will want to take a different path through your website, depending on their interests. You can't assume or expect that everyone will want to view your web pages in the same order.

It is possible, and desirable, to put links anywhere in your content. If a blog post mentions your latest product, then you should link to it so that people can find out more information easily. Providing links in context at the time people might want them makes it much easier for people to explore your website.

When people think of navigation, though, they usually think of those links that are separated from the content that provide access to everything the site offers. A group of links at the top or the side of the page is often called a "navigation bar", or a "navbar" for short. Because it's separated from the page content, it's easy to spot, and it often uses design elements that draw attention to it.

Chameli Devi Group of Institutions, Indore

Department of Artificial Intelligence And Data Science

Subject Notes

AD 505- Internet and Web Technology

UNIT-II

Syllabus: HTML: Basics of HTML, formatting and fonts, commenting code, color, hyperlink, lists, tables, images, forms, XHTML, Meta tags, Character entities, frames, and frame sets, Browser architecture, and Web site structure. Overview and features of HTML5.

## 2.1 BASICS OF HTML

HTML (Hypertext Mark-up Language) is the most basic building block of the Web. It defines the meaning and structure of web content. Other technologies besides HTML are generally used to describe a web page's appearance/presentation.

HTML uses "mark-up" to annotate text, images, and other content for display in a Web browser. HTML mark-up includes special "elements" such as `<head>`, `<title>`, `<body>`, `<header>`, `<footer>`, `<article>`, `<section>`, `<p>`, `<div>`, `<span>`, `<img>`, `<aside>`, `<audio>`, `<canvas>`, `<datalist>`, `<details>`, `<embed>`, `<nav>`, `<output>`, `<progress>`, `<video>` and many others.

### 2.1 FORMATTING AND FONTS IN HTML

#### HTML Formatting Elements

HTML defines special elements for defining text with a special meaning. HTML uses elements like `<b>` and `<i>` for formatting output, like bold or italic text. Formatting elements were designed to display special types of text:

- `<b>` - Bold text
- `<strong>` - Important text
- `<i>` - Italic text
- `<em>` - Emphasized text
- `<mark>` - Marked text
- `<small>` - Small text
- `<del>` - Deleted text
- `<ins>` - Inserted text
- `<sub>` - Subscript text
- `<sup>` - Superscript text

HTML <sup> Element: The HTML <sup> element defines superscripted text.

Example: <p>This is <sup>superscripted</sup> text.</p>

HTML Font Tag:

It specifies the font size, font face and color of text:

Example:

```
<font size="3" color="red">This is some text!</font>
<font size="2" color="blue">This is some text!</font>
<font face="verdana" color="green">This is some text!</font>
```

## 2.2 COMMENT CODE

This element is used to add a comment to an HTML document. An HTML comment begins with <!-- and the comment closes with -->. HTML comments are visible to anyone that views the page source code but are not rendered when the HTML document is rendered by a browser.

HTML comment:

- HTML comments are not displayed on the website. They are used to help you, or other developers understand your code.
- Commenting serves various purposes, such as explaining a string of code or debugging a program.
- Multiline HTML comments usually explain large sections of code.
- Conditional HTML comments are interpreted only by the IE browser.
- Some browsers allow using the <comment> tag.
- A smart HTML comment is the one that adds value and meaning to the source code.
- <!-- comment text -->

### HTML Comment Tags

This type of HTML comment is a regular single-line comment. It is quite similar to the <q> element used for quoting. You can use a single-line comment to deactivate a line of code while debugging. Just insert HTML comment opening and closing tags around the part of the code you want to deactivate.

### Multiline HTML Comments

HTML block comments or HTML multiline comments allow you to comment on a complex or a long piece of code. It works as a regular HTML comment tag, but as it can take several lines to explain a bigger part of the code, a single-line comment won't be enough.

```
<!--  
comment line1  
comment line2  
comment line3  
-->
```

This comment tag works similarly to the `<blockquote>` element, which is used for quoting a bigger piece of text.

Multiline comments in HTML can also disable a block of code. All you have to do is include opening and closing tags around the code you want to deactivate.

#### Example

```
<div>  
  <p>Block comment example</p>  
  <!--  
  A block comment helps when you need to comment out large sections of code.  
  -->  
  </div>
```

### 2.3 COLOR

HTML colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

#### Color Values

In HTML, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values.

#### Example

```
<h1 style="background-color:rgb(255, 99, 71);">...</h1>  
<h1 style="background-color:#ff6347;">...</h1>  
<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>
```

```
<h1 style="background-color:rgba(255, 99, 71, 0.5);">...</h1>  
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">...</h1>
```

### RGB Value

In HTML, a color can be specified as an RGB value, using this formula: `rgb(red, green, blue)`. Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.

For example: `rgb(255, 0, 0)` is displayed as red, because red is set to its highest value (255) and the others are set to 0.

To display black, set all color parameters to 0, like this: `rgb(0, 0, 0)`.

To display white, set all color parameters to 255, like this: `rgb(255, 255, 255)`.

Experiment by mixing the RGB values below: `rgb (255, 99, and 71)`.

### HEX Value

In HTML, a color can be specified using a hexadecimal value in the form: `#rrggbb`, where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).

For example, `#ff0000` is displayed as red, because red is set to its highest value (ff) and the others are set to the lowest value (00).

### HSL Value

In HTML, a color can be specified using hue, saturation, and lightness (HSL) in the form:

`HSL (hue, saturation, lightness)`

The degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue. Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color. Lightness is also a percentage, 0% is black, 50% is neither light nor dark, and 100% is white.

### RGBA Value

RGBA color values are an extension of RGB color values with an alpha channel – which specifies the opacity for a color. An RGBA color value is specified with: `rgba(red, green, blue, alpha)`.

### HSLA Value

HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color. An HSLA color value is specified with: `hsla(hue, saturation, lightness, alpha)`.

With HTML, easily add hyperlinks to any HTML page. Link team page, about page, or even a test by creating a hyperlink. You can also create a hyperlink for an external website. To make a hyperlink in an HTML page, use the `<a>` and `</a>` tags, which are the tags used to define the links.

The `<a>` tag indicates where the hyperlink starts and the `</a>` tag indicates where it ends. Whatever text gets added inside these tags, will work as a hyperlink. Add the URL for the link in the `<a href="">`. Just keep in mind that you should use the `<a>...</a>` tags inside `<body>...</body>` tags. Following are the examples of HTML links:

## Links

Hyperlinks are defined with the HTML `<a>` tag:

```
<a href="url">link text</a>
```

## Local Links

The example above used an absolute URL (a full web address).

A local link (link to the same web site) is specified with a relative URL (without https://www....).`<a href="html_images.asp">HTML Images</a>`

Example:

```
<style>
a:link {
    color: green;
    background-color: transparent;
    text-decoration: none;
}
```

```
a:visited {
    color: pink;
    background-color: transparent;
    text-decoration: none;
}
```

```
a:hover {
```

```
color: red;  
background-color: transparent;  
text-decoration: underline;  
}  
  
a:active {
```

```
color: yellow;  
background-color: transparent;  
text-decoration: underline;  
}  
  
</style>
```

## 2.5 LISTS

### Unordered Lists

An unordered list is a list in which the order of the list items does not matter. Unordered lists should be used when rearranging the order of the list items would not create confusion or change the meaning of the information on the list.

The `<ul>` element opens and closes an unordered list. The items on the list are contained between list item, `<li>` tags. A simple unordered list containing three items could be created with the following HTML.

### Ordered Lists

Ordered lists are used for lists of items for which the order of the items does matter. The syntax for an ordered list is the same as for an unordered list. However, to create an ordered list, the `<ol>` tag is used rather than the `<ul>` tag.

### Changing Numbering

There are times when you want to control the numbering of ordered lists. For example, your list may be broken up by a paragraph that appears mid-list to expand on a certain concept, or you may create a countdown list that begins at a high number and counts down. Lastly, maybe you'd rather use roman numerals. HTML and CSS make it easy to control the numbering of ordered lists.

### Creating a Countdown List

To reverse the number of a list, simply add the `reversed` attribute to the opening `<ol>` tag.

```
<ol reversed>
```

```
  <li>Item 3</li>
```

```
<li>Item 2</li>  
<li>Item 1</li>  
</ol>
```

### Starting a List on a Specific Number

The start attribute is used to specify the number on which an ordered list starts. For example, imagine you have a list of 5 items, and after the second and fourth items, you want to add a sentence or two with additional details. You could use the following HTML to do this without restarting the list numbering after each paragraph.

### Changing the Numbering Style

You can use CSS to change the marker style of an ordered list. In addition to standard numbering (referred to as decimal in CSS), you can also use:

- upper-roman for uppercase roman numerals
- lower-roman for lowercase roman numerals
- decimal-leading-zero to add a “0” placeholder before single-digit list items

### Description Lists

Description lists are created with the dl tag. Used far less frequently than their ordered and unordered peers, description lists are used to contain name-value groups. Each name-value group consists of one name, or term, placed between dt tags, followed by one or more values with each value, or description, placed between dd tags.

### Using Lists for Menus

One of the most common uses of lists is to create website navigation menus. Unordered lists are usually the list-of-choice for this purpose. With just a few lines of CSS, we can convert an unordered list into an attractive horizontal navigation menu.

### Styling Lists

List typography is usually best styled to match the typography of the website's paragraph text. List-specific styling can be accomplished with CSS.

There are three list properties that can be styled with CSS:

- list-style-type: Defines the marker type that precedes each list item. Common values include disc (the default unordered list style type), decimal (the default ordered list style type), circle, square, lower- or upper-roman, and lower- or upper-latin, although several additional styles may also be used.

- **list-style-position:** Determines whether the list item marker should be placed inside the content box, or outside of the content box in the item's left-hand padding area.
- **list-style-image:** An image can also be used as the item marker. This property is used to specify the image file to be used.

## 2.7 TABLE

An HTML table is defined with the `<table>` tag.

Each table row is defined with the `<tr>` tag. A table header is defined with the `<th>` tag. By default, table headings are bold and centered. A table data/cell is defined with the `<td>` tag.

Example:

```
<table style="width:100%">

<tr>
  <th>Firstname</th>
  <th>Lastname</th>
  <th>Age</th>
</tr>

<tr>
  <td>Jill</td>
  <td>Smith</td>
  <td>50</td>
</tr>

<tr>
  <td>Eve</td>
  <td>Jackson</td>
  <td>94</td>
</tr>

</table>
```

## 2.8 IMAGES

In HTML, images are defined with the `<img>` tag.

The `<img>` tag is empty; it contains attributes only and does not have a closing tag.

The src attribute specifies the URL (web address) of the image:

```

```

Images in another Folder

If not specified, the browser expects to find the image in the same folder as the web page.

However, it is common to store images in a sub-folder. You must then include the folder name in the src attribute:

Example: 

## 2.9 FORMS

HTML Form Example

First name:

Last name:

The <form> Element

The HTML <form> element defines a form that is used to collect user input:

```
<form>
```

  form elements

```
  </form>
```

An HTML form contains form elements.

Form elements are different types of input elements, like text fields, checkboxes, radio buttons, submit buttons, and more.

The <input> Element

The <input> element is the most important form element.

The <input> element can be displayed in several ways, depending on the type attribute.

Examples:

Table: 2.1 HTML Form Elements

Type	Description
<input type="text">	Defines a one-line text input field
<input type="radio">	Defines a radio button (for selecting one of many choices)
<input type="submit">	Defines a submit button (for submitting the form)

## 2.10 XHTML

Many pages on the internet contain "bad" HTML.

This HTML code works fine in most browsers (even if it does not follow the HTML rules):

```
<html>
<head>
<title>This is bad HTML</title>
```

```
<body>
```

```
  <h1>Bad HTML
```

```
  <p>This is a paragraph
```

```
</body>
```

The important differences from HTML are as follows:

### Document Structure

- XHTML DOCTYPE is mandatory.
- The xmlns attribute in <html> is mandatory.
- <html>, <head>, <title>, and <body> are mandatory.

### XHTML Elements

- XHTML elements must be properly nested.
- XHTML elements must always be closed.

- XHTML elements must be in lowercase.
- XHTML documents must have one root element.

### XHTML Attributes

- Attribute names must be in lower case.
- Attribute values must be quoted.
- Attribute minimization is forbidden.

## 2.11 META TAGS

Meta tags are snippets of text that describe a page's content; the meta tags don't appear on the page itself, but only in the page's code. We all know tags from blog culture, and meta tags are the same thing, little content descriptors that help tell search engines what a web page is about.

### HTML <meta> Tag

Example:

Describe metadata within an HTML document:

```
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Free Web tutorials">
  <meta name="keywords" content="HTML,CSS,XML,JavaScript">
  <meta name="author" content="CDGI">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
```

## 2.12 CHARACTER ENTITIES

The following table lists the essential entities in HTML:

Table: 2.2 HTML Entities

Character	Entity Name	Entity Number	Description
&	&amp; &#38;	Ampersand	
"	&quot; &#34;	Double quote mark	
<	&lt; &#60;	Less than symbol	
>	&gt; &#62;	Greater than symbol	
'	&apos; &#39;	Apostrophe (XHTML only)	

## 2.13 FRAMES AND FRAME SET

<Frame> tags are used to divide the web browser window into multiple sections where each section can be loaded separately. A frameset tag is the collection of frames in the browser window:

Example:

A simple three-framed page:

```
<frameset cols="25%,50%,25%">  
  <frame src="frame_a.htm">  
  <frame src="frame_b.htm">  
  <frame src="frame_c.htm">  
</frameset>
```

### 2.14.1 WEB BROWSER ARCHITECTURE

A browser is a software application used to locate, retrieve and display content on the World Wide Web, including Web pages, images, videos, and other files. As a client/server model, the browser is the client run on a computer that contacts the Web server and requests information. The Web server sends the information back to the Web browser which displays the results on the computer or other Internet-enabled device that supports a browser.

High-level architecture of browser

The below image shows the main components of a web browser:

Figure: 2.1 Main components of the browser

**The User Interface:** The user interface is the space where the user interacts with the browser. It includes the address bar, back, and next buttons, home button, refresh and stop, bookmark option, etc. Every other part, except the window where the requested web page is displayed, comes under it.

**The Browser Engine:** The browser engine works as a bridge between the User interface and the rendering engine. According to the inputs from various user interfaces, it queries and manipulates the rendering engine.

**The Rendering Engine:** The rendering engine, as the name suggests is responsible for rendering the requested web page on the browser screen. The rendering engine interprets the HTML, XML documents, and images that are formatted using CSS and generates the layout that is displayed in the User Interface. However, using plugins or extensions, it can display other types of data also. Different browsers use different rendering engines.

**Networking:** Component of the browser which retrieves the URLs using the common internet protocols of HTTP or FTP. The networking component handles all aspects of Internet communication and security. The network component may implement a cache of retrieved documents in order to reduce network traffic.

**JavaScript Interpreter:** It is the component of the browser which interprets and executes the JavaScript code embedded in a website. The interpreted results are sent to the rendering engine for display. If the script is external, then first the resource is fetched from the network. Parser keeps on hold until the script is executed.

**UI Backend:** UI backend is used for drawing basic widgets like combo boxes and windows. This backend exposes a generic interface that is not platform-specific. It underneath uses operating system user interface methods.

**Data Persistence/Storage:** This is a persistence layer. Browsers support storage mechanisms such as localStorage, IndexedDB, WebSQL, and FileSystem. It is a small database created on the local drive of the computer where the browser is installed. It manages user data such as cache, cookies, bookmarks, and preferences

## 2.14.2 WEB SITE STRUCTURE

Structuring your website is crucial for both its usability and findability. Many sites lack a sound structure to guide visitors to the information they're looking for. Having a clear site structure also leads to a better understanding of your site by Google, so it's incredibly important for your SEO. Let's take a closer look at how this works.

### Ideal website structure

Let's start by looking at an ideal situation: if you're starting from scratch, how should you organize your site? We think a well-organized website looks like a pyramid with several levels:

1. Homepage
2. Categories (or sections)
3. Subcategories (only for larger sites)

#### 4. Individual pages and posts

The homepage should be all the way to the top. Then, you have some sections or category pages beneath it. You should be able to file all your content under one of these categories. If your site is larger, you can divide these sections or categories into subcategories as well. Beneath your categories or subcategories are your individual pages and posts.

Figure: 2.2 Website Structure

#### 2.15 OVERVIEW AND FEATURES OF HTML5

HTML5 introduces several new elements and attributes that can help you in building modern websites.

Following describes the basic structure of HTML-5 document:

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

#### New Semantic/Structural Elements

HTML5 offers new elements for better document structure:

Tag	Description
-----	-------------

<article>
-----------

Defines an article in a document
----------------------------------

<aside>

Defines content aside from the page content

<bdi>

Isolates a part of the text that might be formatted in a different direction from other text outside it

<details>

Defines additional details that the user can view or hide

<dialog>

Defines a dialog box or window

<figcaption>

Defines a caption for a <figure> element

<figure>

Defines self-contained content

<footer>

Defines a footer for a document or section

<header>

Defines a header for a document or section

<main>

Defines the main content of a document

<mark>

Defines marked/highlighted text

<meter>

Defines a scalar measurement within a known range (a gauge)

<nav>

Defines navigation links

<progress>

Represents the progress of a task

<rp>

Defines what to show in browsers that do not support ruby annotations

<rt>

Defines an explanation/pronunciation of characters (for East Asian typography)

<ruby>

Defines a ruby annotation (for East Asian typography)

<section>

Defines a section in a document

Table: 2.3 HTML5 Semantic/Structure Elements

New Input Types

- color
- date
- datetime
- datetime-local
- email
- month
- number
- range
- search
- tel
- time

Chameli Devi Group of Institutions, Indore

Department of Artificial Intelligence And Data Science

Subject Notes

AD 505- Internet and Web Technology

UNIT-III

Syllabus: Style sheets: Need for CSS, introduction to CSS, basic syntax and structure, using CSS, background images, colors and properties, manipulating texts, using fonts, borders and boxes, margins, padding lists, positioning using CSS, CSS2, Overview and features of CSS3, JavaScript: Client-side scripting with JavaScript, variables, functions, conditions, loops and repetition, Pop-up boxes, Advance JavaScript: JavaScript and objects, JavaScript own objects, DOM and web browser environments, Manipulation using DOM, forms and validations, DHTML : Combining HTML, CSS, and JavaScript, Events and buttons.

Style sheet:

A Style Sheet is a collection of style rules that tells a browser how the various styles are to be applied to the HTML tags to present the document. Rules can be applied to all the basic HTML elements, for example, the `<p>` tag, or we can define our own variations and apply them as per the requirement.

There are three types of Style Sheets:

- Embedded: The style rules are included within the HTML at the top of the web page.
- Inline: The style rules appear throughout the HTML of the web page.
- Linked: The style rules are stored in a separate file external to all the web pages.

### 3.1 NEED FOR CSS

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on the screen, paper, or in other media.
- CSS saves a lot of work. It can control the layout of multiple web pages all at once.
- External style sheets are stored in CSS files.

### INTRODUCTION TO CSS

- HTML was NEVER intended to contain tags for formatting a web page.

- HTML was created to describe the content of a web page, like:
  - <h1>This is a heading</h1>
  - <p>This is a paragraph.</p>
- When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

### 3.2 BASIC SYNTAX AND STRUCTURE

A CSS ruleset consists of a selector and a declaration block:

Figure 3.1: Syntax and Structure of CSS

The selector points to the HTML element we want to style.

The declaration block contains one or more declarations separated by semicolons. Each declaration includes a CSS property name and a value, separated by a colon. A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

#### Example

In this example all <p> elements will be center aligned, with a red text color:

```
p {  
    color: red;  
    text-align: center;  
}
```

### 3.3 USING CSS BACKGROUND IMAGES

We can set background images in CSS using the background-image and several other properties to control the behavior of the image. Background properties include background-repeat, background-attachment, etc.

Possible values are:

- Top.
- Right.
- Bottom.
- Left.
- Enter any combination of the above

## CSS background-image Property

Example: Set a background-image for the <body> element:

```
body {  
background-image: url("paper.gif");  
background-color: #cccccc;  
}
```

## 3.4 COLORS AND PROPERTIES

Adding color is done using the "color" property for the foreground color and the "background-color" property for the background color. The "color" property specifies the color of the text for the HTML element. There are two ways to specify color in CSS:

Example: Set the text-color for different elements:

```
body {  
color: red;  
}  
  
h1 {  
color: #00ff00;  
}  
  
p.ex {  
color: rgb(0,0,255);  
}
```

## 3.5 MANIPULATING TEXTS

The text can be styled with some of the text formatting properties. The heading uses the text-align, text-transform, and color properties. The paragraph is indented, aligned, and the space between characters is specified.

### Text Color

The color property is used to set the color of the text. The color is specified by:

- Color name - like "red"
- HEX value - like "#ff0000"

- RGB value - like "rgb(255,0,0)"

Look at CSS color values for a complete list of possible color values.

Example

```
body {
  color: blue;
}
```

```
h1 {
  color: green;
}
```

### 3.6 USING FONTS

CSS Fonts is a module of CSS that defines font-related properties and how font resources are loaded. It lets us define the style of a font, such as its family, size and weight, line height, and the glyph variants to use when multiple are available for a single character.

Font Family

- The font family of a text is set with the font-family property.
- The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.
- Start with the font we want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

Note: If the name of a font family is more than one word, it must be in quotation marks, like: "Times New Roman".

Example

```
p {
  font-family: "Times New Roman", Times, serif;
}
```

### 3.7 BORDERS AND BOXES

By default, in the CSS box model, the width and height we assign to an element are applied only to the element's content box. Border-box tells the browser to account for any border and padding in the values we specify for an element's width and height.

Example

Include padding and border in the element's total width and height:

```
#example1 {  
    box-sizing: border-box;  
}
```

### 3.8 MARGINS

The margin property sets the margins for an element, and is a shorthand property for the following properties:

- margin-top
- margin-right
- margin-bottom
- margin-left

In case the margin property has four values:

- margin: 10px 5px 15px 20px;
- o Top margin is 10px
- o Right margin is 5px
- o Bottom margin is 15px
- o Left margin is 20px

In case the margin property has three values:

- margin: 10px 5px 15px;
- o Top margin is 10px
- o Right and left margins are 5px
- o Bottom margin is 15px

In case the margin property has two values:

- margin: 10px 5px;
- o Top and bottom margins are 10px
- o Right and left margins are 5px

In case the margin property has one value:

- margin: 10px;
- o All four margins are 10px

Example: Set the margin for all four sides of a <p> element to 35 pixels:

```
p {  
    margin: 35px;  
}
```

### 3.9 PADDING LIST

The CSS padding properties are used to generate space around an element's content, inside of any defined borders. With CSS, we have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

CSS has properties for specifying the padding for each side of an element:

- padding-top
- padding-right
- padding-bottom
- padding-left

All the padding properties can have the following values:

- Length - specifies padding in px, pt, cm, etc.
- % - specifies padding in % of the width of the containing element
- inherit - specifies that the padding should be inherited from the parent element

Example: Set different padding for all four sides of a <div> element:

```
div {  
    padding-top: 50px;  
    padding-right: 30px;  
    padding-bottom: 50px;  
    padding-left: 80px;  
}
```

### 3.10 POSITIONING USING CSS

The position property specifies the type of positioning method used for an element.

There are five different position values:

- static

- relative
- fixed
- absolute
- sticky

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

#### Example

```
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}
```

```
div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
```

#### 3.11 CSS2

Cascading Style Sheets Level 2 (CSS2) is the second version of cascading style sheets developed by W3C. It's a declarative language used to enhance the hyper extensive text mark-up language. CSS2 is a subset of Cascading Style Sheets Level 1 and has enhanced capabilities. CSS2 has backward compatibility, so all valid CSS1 is also valid CSS2.

- Media types concept
- Aural style sheets

- Features for internationalization
- Extended font selection
- Automatic numbering and generated content
- Cursors
- Dynamic outlines
- Capability to control content overflow, clipping
- Absolute, fixed and relative positioning
- Extended selector mechanism

### 3.12 OVERVIEW AND FEATURES CSS3

CSS3 version introduced new features that allowed developers to style HTML elements with less CSS code. CSS3 is most notorious for modules that speed up the specification process. CSS3 is backwards-compatible with former CSS versions. It means that CSS3 features work on web pages using older CSS. Browsers that support CSS2 also represent the modifications with CSS3. CSS3 allows developers to style HTML elements easier. They are less dependent on image files and can complete CSS styling with fewer lines of code.

Features of CSS3 are as follows:

- Box Shadow: One of the CSS3 new features is the box-shadow property that adds a shadow to an element. Instead of using multiple images around an item, this property lets us add shadow with a short line of code.
- Opacity: One of the CSS3 properties called opacity makes elements see-through or completely transparent. For instance, we can apply opacity to images or other HTML elements. The transparency level depends on the indicated values.
- Rounded Corners: Before the release of CSS3, developers had to write long code to produce rounded corners. Now, it is enough to apply the border-radius CSS3 property to HTML elements.
- Attribute Selectors: CSS3 also introduced new selectors in addition to the ones in CSS2. Instead of applying IDs or classes for styling, developers can select HTML elements according to their attributes. As a result, we do not have to create unique IDs only to apply CSS rules.
- New Colors: One of the CSS3 features is the addition of new colors: RGBA, HSL, HSLA, Gradient Colors

### 3.13 CLIENT-SIDE SCRIPTING WITH JAVASCRIPT

Server-side scripting is executed by a web server; client-side scripting is executed by a browser. Client-end scripts are embedded in a website's HTML mark-up code, which is housed on the server in a language that's compatible with or compiled to communicate with the browser.

JavaScript is a scripting language most often used for client-side web development. Client-side refers to operations that are performed by the client (in our case the client is the browser) in a client-server relationship. Despite the name, JavaScript is essentially unrelated to the Java programming language.

Figure 3.2: Scripting language workflow

Example

```
<script>  
document.getElementById("demo").innerHTML = "Hello JavaScript!";  
</script>
```

### 3.14 VARIABLE, FUNCTION, CONDITION, LOOP, AND REPETITIONS

#### 3.14.1 VARIABLE

JavaScript variables are containers for storing data values.

In this example, x, y, and z are variables:

Example

```
var x = 5;  
var y = 6;  
var z = x + y;
```

#### 3.14.2 FUNCTION

Functions in Java Script are declared with the following syntax:

```
function functionName(parameters) {  
    // code to be executed  
}
```

Example

```
function myFunction(a, b) {  
    return a * b;  
}
```

#### 3.14.3 CONDITION

Very often when we write code, we want to perform different actions for different decisions.

In JavaScript we have the following conditional statements:

- Use if to specify a block of code to be executed, if a specified condition is true
- Use else to specify a block of code to be executed, if the same condition is false
- Use else if to specify a new condition to test, if the first condition is false
- Use switch to specify many alternative blocks of code to be executed

### If Statement

Use the if statement to specify a block of JavaScript code to be executed if a condition is true.

#### Syntax

```
if (condition) {  
    // Block of code to be executed if the condition is true  
}
```

#### Example

```
if (hour < 18) {  
    greeting = "Good day";  
}
```

### 3.14.4 LOOP AND REPETITIONS

What we need is a generic solution for repeating code with control over how many times the code repeats. In JavaScript, this solution is provided in the form of something known as a loop. There are three kinds of loops we can use to repeat some code:

- for loops
- while loops
- do...while loops

Each of these three loop variations allows us to specify the code we want to repeat (aka loop) and a way to stop the repetition when a condition is met.

#### Example

```
<!DOCTYPE html>  
  
<html>  
  
<head>  
  
<meta charset="utf-8">
```

```
<title>Loops!</title>
<style>
</style>
</head>
<body>
<script>
    for (var i = 0; i < count; i++) {
        saySomething();
    }
    function saySomething() {
        document.writeln("hello!");
    }
</script>
</body>
</html>
```

### 3.15 POP UP BOXES

#### Alert Box

An alert box is often used if we want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

#### Syntax

```
window.alert("alert text");
```

The `window.alert()` method can be written without the `window` prefix.

#### Example

```
alert ("I am an alert box!");
```

#### Confirm Box

A confirm box is often used if we want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

#### Syntax

```
window.confirm("confirmation text");
```

The `window.confirm()` method can be written without the `window` prefix.

Example

```
if (confirm("Press a button!")) {  
    txt = "You pressed OK!";  
}  
else {  
    txt = "You pressed Cancel!";  
}
```

### 3.16 ADVANCE JAVASCRIPT: JAVASCRIPT AND OBJECTS

Objects are Variables

JavaScript variables can contain single as well as multiple values.

Example

```
var person = "John Doe";
```

The values are written as name: value pairs (name and value separated by a colon).

Example

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

The following can be represented as objects in JavaScript:

- Booleans can be objects (if defined with the `new` keyword)
- Numbers can be objects (if defined with the `new` keyword)
- Strings can be objects (if defined with the `new` keyword)
- Dates are always objects
- Maths are always objects
- Regular expressions are always objects
- Arrays are always objects
- Functions are always objects
- Objects are always objects

### 3.17 JAVA SCRIPT OWN OBJECTS

In real life, a car is an object.

A car has properties like weight and color, and methods like start and stop:

Table 3.1: Object, Properties and Methods Example

Object	Properties	Methods
--------	------------	---------

car.name = Fiat

car.model = 500

car.weight = 850kg

car.color = white

car.start()

car.drive()

car.brake()

car.stop()

### 3.18 DOM AND WEB BROWSER ENVIRONMENTS

A host environment provides platform-specific objects and functions to the language core. Web browsers give a means to control web pages. Below figure shows the DOM with browser environment:

Figure 3.3: DOM and web browser environments

What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard.

- "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The W3C DOM standard is separated into 3 different parts:

- Core DOM - standard model for all document types

- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

## HTML DOM

The HTML DOM is a standard object model and programming interface for HTML. It defines:

- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements

## 3.19 MANIPULATION USING DOM

DOM manipulation methods allow to add, edit or delete DOM element(s) in the web page. Use the selector to get the reference of an element(s) and then call manipulate methods to edit it.

Important DOM manipulation methods are: append(), propend(), before(), after(), remove(), replaceAll(), wrap().

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements on the page
- JavaScript can change all the HTML attributes on the page
- JavaScript can change all the CSS styles on the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events on the page
- JavaScript can create new HTML events on the page

### Example

//This example changes the content of a <p> element:

```
<html>
```

```
<body>
```

```
<p id="p1">Hello World!</p>
```

```
<script>
```

```
document.getElementById("p1").innerHTML = "New text!";
</script>
```

```
</body>
```

```
</html>
```

### 3.20 FORM AND VALIDATIONS

The data entered in the form needs to be in the right format and certain fields need to be filled in order to effectively use the submitted form. Username, password, contact information is some details that are mandatory in forms and thus need to be provided by the user.

The below example validates the form:

Form validation:

```
<script>
function validation()
{
    var name = document.forms["RegForm"]["Name"];
    var email = document.forms["RegForm"]["EMail"];
    var phone = document.forms["RegForm"]["Telephone"];
    var what = document.forms["RegForm"]["Subject"];
    var password = document.forms["RegForm"]["Password"];
    var address = document.forms["RegForm"]["Address"];

    if (name.value == "")
    {
        window.alert("Please enter your name.");
        name.focus();
        return false;
    }
}
```

```
if (address.value == "")  
{  
    window.alert("Please enter your address.");  
    name.focus();  
    return false;  
}  
  
if (email.value == "")  
{  
    window.alert("Please enter a valid e-mail address.");  
    email.focus();  
    return false;  
}  
  
if (email.value.indexOf("@", 0) < 0)  
{  
    window.alert("Please enter a valid e-mail address.");  
    email.focus();  
    return false;  
}  
  
if (email.value.indexOf(".", 0) < 0)  
{  
    window.alert("Please enter a valid e-mail address.");  
    email.focus();  
    return false;  
}  
  
if (phone.value == "")
```

```

{
    window.alert("Please enter your telephone number.");
    phone.focus();
    return false;
}

if (password.value == "")
{
    window.alert("Please enter your password");
    password.focus();
    return false;
}

if (what.selectedIndex< 1)
{
    alert("Please enter your course.");
    what.focus();
    return false;
}

return true;
}
</script>

```

### 3.21DHTML

Dynamic HTML, or DHTML, is an umbrella term for a collection of technologies used together to create interactive and animated websites by using a combination of a static markup language (such as HTML), a client-side scripting language (such as JavaScript), a presentation definition language (such as CSS).

Figure 3.4: Working of DHTML

Following are the various characteristics or features of DHTML (Dynamic HTML):

- Its simplest and the main feature is that we can create the web page dynamically.
- Dynamic Style is a feature that allows the users to alter the font, size, color, and content of a web page.
- It provides the facility for using the events, methods, and properties. And, also provides the feature of code reusability.
- It also provides the feature in browsers for data binding.
- Using DHTML, users can easily create dynamic fonts for their websites or web pages.
- With the help of DHTML, users can easily change the tags and their properties.
- The web page functionality is enhanced because the DHTML uses a low-bandwidth effect.

### 3.22 Combining HTML

User can merge two or more table cells together by using the colspan attribute in a <td> HTML tag (table data). For example, in the below code is a table with three rows and three columns. If we wanted to combine the first two cells into one cell, we could use the colspan="2" attribute in the first <td> tag.

```
<table>
<tr>
<td colspan="2">&nbsp;</td>
<td>&nbsp;</td>
</tr>
<tr>
<td>&nbsp;</td>
<td>&nbsp;</td>
<td>&nbsp;</td>
</tr>
<tr>
<td>&nbsp;</td>
<td>&nbsp;</td>
<td>&nbsp;</td>
</tr>
</table>
```

### 3.23 CSS AND JAVASCRIPT

- The <script> tag is used to define a client-side script (JavaScript).
- The <script> element either contains script statements, or it points to an external script file through the src attribute.
- Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content.
- To select an HTML element, JavaScript most often uses the document, getElementById () method.

Example: This JavaScript example writes "Hello JavaScript!" into an HTML element with id="demo"

```
<Script>

document.getElementById("demo").inner HTML = "Hello JavaScript!";

</script>
```

### 3.24 EVENTS AND BUTTONS

An event handler allows us to execute code when an event occurs.

```
<h1 onclick="this.innerHTML='Ooops!'>Click on this text</h1>
```

We can also add a script in the head section of the page and then call the function from the event handler as shown below:

```
<html>

<head>

<script type="text/javascript">

function changetext(id)

{

id.innerHTML="Ooops!";

}

</script>

</head>

<body>

<h1 onclick="changetext(this)">Click on this text</h1>
```

```
</body>
```

```
</html>
```

Chameli Devi Group of Institutions, Indore

Department of Artificial Intelligence And Data Science

Subject Notes

AD 505- Internet and Web Technology

UNIT-IV

Syllabus: XML: Introduction to XML, uses of XML, simple XML, XML key components, DTD and Schemas, Using XML with the application, Transforming XML using XSL and XSLT, PHP: Introduction and basic syntax of PHP, decision and looping with examples, PHP and HTML, Arrays, Functions, Browser control, and detection, String, Form processing, Files, Advance features: Cookies and Sessions, Object-Oriented Programming with PHP.

4.1      Introduction to XML:

XML (Extensible Markup Language) is a markup language similar to HTML, but without predefined tags to use. Instead, we define our own tags designed specifically for our needs. This is a powerful way to store data in a format that can be stored, searched, and shared. Most importantly, since the fundamental format of XML is standardized, if we share or transmit XML across systems or platforms, either locally or over the internet, the recipient can still parse the data due to the standardized XML syntax. Following are the properties of XML:

- XML stands for extensible Markup Language
- XML is a markup language like HTML
- XML is designed to store and transport data
- XML is designed to be self-descriptive

#### Example

```
<?xml version="1.0" encoding="UTF-8"?>  
<message>  
<warning>  
    Hello World  
</warning>  
</message>
```

The top line `<?xml version="1.0" encoding="UTF-8"?>` is called XML prolog.

## 4.2 Uses of XML

XML has a variety of uses for Web, e-business, and portable applications. The following are some of the many applications for which XML is useful:

- Web publishing: XML allows us to create interactive pages, allows the customer to customize those pages, and makes creating e-commerce applications more intuitive. With XML, we store the data once and then render that content for different viewers or devices based on style sheet processing using an Extensible Style Language (XSL)/XSL Transformation (XSLT) processor.
- Web searching and automating Web tasks: XML defines the type of information contained in a document, making it easier to return useful results when searching the Web:

For example, using HTML to search for books authored by Tom Brown is likely to return instances of the term 'brown' outside of the context of the author. Using XML restricts the search to the correct context (for example, the information contained in the `<author>` tag) and returns only the information that we want. By using XML, Web agents and robots (programs that automate Web searches or other tasks) are more efficient and produce more useful results.

- General applications: XML provides a standard method to access information, making it easier for applications and devices of all kinds to use, store, transmit, and display data.
- e-business applications: XML implementations make electronic data interchange (EDI) more accessible for information interchange, business-to-business transactions, and business-to-consumer transactions.
- Metadata applications: XML makes it easier to express metadata in a portable, reusable format.
- Pervasive computing: XML provides portable and structured information types for display on pervasive (wireless) computing devices such as personal digital assistants (PDAs), cellular phones, and others. For example, WML (Wireless Markup Language) and VoiceXML are currently evolving standards for describing visual and speech-driven wireless device interfaces.

#### 4.3 Simple XML

This simplified version of XML is a variant of the proposed XML language. It removes some features and adds a few others. It mainly differs from XML in these aspects:

- Separation between lexical part and grammar part means that is much easier to parse.
- There is no distinction between `<foo></foo>` and `<foo/>`, they can be used interchangeably.
- Documents can be nested: there may be `<!doctype>` declarations in the middle of a document.
- Default attribute declarations are scoped: they are valid until the end of the current element or doctype.
- Whitespace handling is simplified: a newline immediately before a '`<`' and a newline immediately after a '`>`' are ignored; no other whitespace is ignored by the parser (i.e., all other whitespace is passed on to the application).
- Only character entities are allowed, no other types of entities exist.
- There is no internal document type subset. The allowed structure of the document can only be specified in a separate document.

#### 4.4 XML Key Components

Following are the key components of XML:

1. XML Base: It overrides the default URI of a document or any part of a document starting at a given element;
2. Stylesheets in XML: It associates an XSLT transformation with an XML document so that a Web browser will format it.
3. XLink: A vocabulary for hypertext in XML.
4. `xml:id`: It identifies an XML attribute or element as containing a name that can be used as a unique identifier within a document.

5. XInclude: It includes all or part of other text or XML documents or duplicate part of the current XML document.
6. XPointer: This is a framework for different ways to point into XML documents and is used by XLink.
7. XForms: A more powerful version of HTML forms.
8. XML Events, XHTML Modularization: These specifications are primarily related to the use of XML in Web browsers or other DOM-based systems.

#### 4.5.1 DTD

The purpose of a DTD (Document Type Definition) is to define the structure of an XML document. It defines the structure with a list of legal elements.

Example

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

The DTD above is interpreted like this:

- !DOCTYPE note: It defines that the root element of the document is note.
- !ELEMENT note :It defines that the note element must contain the elements: "to, from, heading, body".
- !ELEMENT to: It defines the element to be of type "#PCDATA".
- !ELEMENT from: It defines the from element to be of type "#PCDATA".
- !ELEMENT heading: It defines the heading element to be of type "#PCDATA".
- !ELEMENT body: It defines the body element to be of type "#PCDATA".

#### 4.5.2 Schema

An XML Schema describes the structure of an XML document. The XML Schema language is also referred to as XML Schema Definition (XSD).

Example

```

<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>

```

The purpose of an XML Schema is to define the legal building blocks of an XML document:

- The elements and attributes that can appear in a document
- The number of (and order of) child elements
- Data types for elements and attributes
- Default and fixed values for elements and attributes

#### 4.6 Using XML with Application

The Extensible Markup Language (XML) is a subset of SGML designed to enable generic SGML to be served, received, and processed on the web easily and efficiently. Because its format is not fixed, XML enables us to design our own customized markup languages with which to create various types of documents. In the IDE, XML documents are represented by XML nodes. The IDE provides tools to assist us in creating, editing, checking, and validating the various XML document types it supports.

The IDE supports several types of XML documents, including:

- Cascading Style Sheet. A CSS provides a simple mechanism for formatting and adding style to HTML and XML documents.

- Document Type Definition. A DTD describes the grammar that can be used in an XML file and indicates the valid arrangement of the file's tags. It can exist as a prologue to an XML file (internal DTD) or as a separate file (external DTD).
- Parsed Entity Objects. Parsed entity nodes represent parsed entity (.ent) files. If several of our XML documents refer to a single piece of information, we can store that information in a parsed entity. Any changes we make to the parsed entity are then automatically reflected in all the documents that reference it.
- XML Catalogs. XML catalogs provide mapping information that maps an external entity in an XML document to the actual location of the document being referenced. We can use a catalog to redirect mappings for an external entity, such as a DTD or XML file, to a different location.
- XSL Style sheet. XSL style sheets define transformation rules for our XML documents. We can perform an XML transformation to transform the data in an XML document into the output of our choice, such as formatted HTML or a text file.

#### 4.7 Transforming XML using XSL and XSLT

XSL (Extensible Style Sheet Language) or XSLT (Extensible Style Sheet Language Transformations) is a language for transforming XML documents into other XML documents, or other formats such as HTML for web pages, plain text, or into XSL Formatting Objects, which may subsequently be converted to other formats, such as PDF, PostScript, and PNG. With XSLT we can transform an XML document into HTML. XSLT is the recommended style sheet language for XML. XSLT is far more sophisticated than CSS. With XSLT we can add/remove elements and attributes to or from the output file. We can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more. XSLT uses XPath to find information in an XML document.

##### Example

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "student.xslt"?>
<class>
<student rollno = "393">
<firstname>Dinkar</firstname>
<lastname>Kad</lastname>
<nickname>Dinkar</nickname>
<marks>85</marks>
</student>
</class>
```

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsl:stylesheet version = "1.0"
xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
<xsl:template match = "/">
<html>
<body>
<h2>Students</h2>
<table border = "1">
<tr bgcolor = "#9acd32">
<th>Roll No</th>
<th>First Name</th>
<th>Last Name</th>
<th>Nick Name</th>
<th>Marks</th>
</tr>
<xsl:for-each select="class/student">
<tr>
<td>
<xsl:value-of select = "@rollno"/>
</td>
<td><xsl:value-of select = "firstname"/></td>
<td><xsl:value-of select = "lastname"/></td>
<td><xsl:value-of select = "nickname"/></td>
<td><xsl:value-of select = "marks"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
```

```
</xsl:template>  
</xsl:stylesheet>
```

#### 4.8.1 Introduction of PHP

PHP or Hypertext Pre-processor is a widely used open-source general-purpose scripting language and can be embedded with HTML. PHP files are saved with the “.php” extension.

Following are the features of PHP:

- PHP is a widely used, open-source scripting language.
- PHP scripts are executed on the server.
- PHP is free to download and use.
- PHP can generate the dynamic page content.
- PHP can create, open, read, write, delete, and close files on the server.
- PHP can add, delete, modify data in our database.
- PHP can be used to control user access.
- PHP can encrypt data.
- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.).
- PHP is compatible with almost all servers used today (Apache, IIS, etc.).
- PHP supports a wide range of databases.

#### 4.8.2 Basic Syntax of PHP

```
<?php  
// PHP code goes here  
?>
```

Example

```
<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My first PHP page</h1>
```

```
<?php  
echo "Hello World!";  
?>
```

```
</body>  
</html>
```

#### 4.9.1 Decision in PHP with Examples

Very often when we write code, we want to perform different actions for different conditions. We can use conditional statements in our code to do this.

In PHP we have the following conditional statements:

- if statement - executes some code if one condition is true.
- if else statement - executes some code if a condition is true and another code if that condition is false.
- if elseif else statement - executes different codes for more than two conditions.
- switch statement - selects one of many blocks of code to be executed.

Example

```
<?php  
$t = date("H");  
if ($t < "20") {  
    echo "Have a good day!";  
} else {  
    echo "Have a good night!";  
}  
?>
```

```
<?php  
$favcolor = "red";  
switch ($favcolor) {
```

```

case "red":
    echo "Your favorite color is red!";
    break;
case "blue":
    echo "Your favorite color is blue!";
    break;
case "green":
    echo "Your favorite color is green!";
    break;
default:
    echo "Your favorite color is neither red, blue, nor green!";
}

?>

```

#### 4.9.2 Looping in PHP with Examples

Often when we write code, we want the same block of code to run repeatedly a certain number of times. So, instead of adding several almost equal code lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, if a certain condition is true.

In PHP, we have the following loop types:

- while - Loops through a block of code if the specified condition is true
- do while - Loops through a block of code once, and then repeats the loop if the specified condition is true
- for - Loops through a block of code a specified number of times
- for each - Loops through a block of code for each element in an array

Example

```
<?php
$x = 1;
while($x <= 5) {
    echo "The number is: $x <br>";
}
```

```

$x++;
}

?>

<?php

$x = 1;

do {

echo "The number is: $x <br>";

$x++;

} while ($x <= 5);

?>

<?php

for ($x = 0; $x <= 10; $x++)

{

echo "The number is: $x <br>";

}

?>

<?php

$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {

echo "$value <br>";

}

?>

```

#### 4.10 PHP and HTML

On the HTML page, PHP code is enclosed within special PHP tags. When a visitor opens the page, the server processes the PHP code and then sends the output (not the PHP code itself) to the visitor's browser. It is quite simple to integrate HTML and PHP. A PHP script can be treated as an HTML page, with bits of PHP inserted here and there. Anything in a PHP script that is not contained within <?php ?> tags is ignored by the PHP compiler and passed directly to the web browser.

#### Example

```
<html>
<head></head>
<body>
<ul>
<?php for($i=1;$i<=5;$i++){ ?>
<li>Menu Item <?php echo $i; ?></li>
<?php } ?>
</ul>
</body>
</html>
```

#### 4.11.1 Arrays

An array is a special variable, which can hold more than one value at a time. If we have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars [0]. " , " . $cars[1] . "and " . $cars[2] . ".";
?>
```

#### 4.11.2 Functions

PHP functions are like other programming languages. A function is a piece of code that takes one or more input in the form of a parameter and does some processing and returns a value.

Example

```
<?php  
function writeMsg() {  
    echo "Hello world!";  
}  
writeMsg(); // call the function  
?>
```

#### 4.11.3 Browser Control and Detection

**PHP get\_browser() Function:** The `get_browser()` function in PHP is an inbuilt function which is used to tell the user about the browser's capabilities. The `get_browser()` function looks up the user's `browscap.ini` file and returns the capabilities of the user's browser.

Syntax: `get_browser(user_agent, return_array)`

Parameters Used:

The `get_browser()` function in PHP accepts two parameters:

1. `user_agent` : It is an optional parameter that specifies the name of an HTTP user agent. Default is the value of `$HTTP_USER_AGENT`.
2. `return_array`: It is an optional parameter that returns an array instead of an object if it is set to True.

Exceptions:

1. The `user_agent` parameter can be bypassed with a NULL value.
2. The `cookies` value simply means that the browser itself can accept cookies and does not mean the user has enabled the browser to accept cookies or not.
3. In order for this function to work the `browscap` configuration setting in `php.ini` must point to the correct location of the `browscap.ini` file on our system.

Example

```
<?php  
echo$_SERVER['HTTP_USER_AGENT'];  
//using get_browser() to display capabilities of the user browser  
$mybrowser= get_browser();  
print_r($mybrowser);  
?>
```

## 4.12 PHP String

A string is a sequence of characters, like "Hello world!". Following are some commonly used functions to manipulate strings:

- `strlen()` - Return the length of a string

```
<?php  
echo strlen("Hello world!"); // outputs 12  
?>
```

- `str_word_count()` - Count words in a string

```
<?php  
echo str_word_count("Hello world!"); // outputs 2  
?>
```

- `strrev()` - Reverse a string

```
<?php  
echo strrev("Hello world!"); // outputs !dlrowolleH  
?>
```

- `strpos()` - Search for a text within a string

```
<?php  
echo strpos("Hello world!", "world"); // outputs 6  
?>
```

- `str_replace()` - Replace text within a string

```
<?php  
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!  
?>
```

## 4.13 Form Processing

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.

Example

```
<html>

<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

```
<html>
<body>

Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>

</body>
</html>
```

#### 4.14 Files

File handling is an important part of any web application. We often need to open and process a file for different tasks. PHP has several functions for creating, reading, uploading, and editing files as follows:

```
<?php
echo readfile("webdictionary.txt");
?>
```

```
<?php  
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");  
echo fread($myfile,filesize("webdictionary.txt"));  
fclose($myfile);  
?>
```

```
<?php  
$myfile = fopen("webdictionary.txt", "r");  
// some code to be executed....  
fclose($myfile);  
?>
```

```
<?php  
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");  
echo fgets($myfile);  
fclose($myfile);  
?>
```

```
<?php  
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");  
// Output one character until end-of-file  
while(!feof($myfile)) {  
    echo fgetc($myfile);  
}  
fclose($myfile);  
?>
```

#### 4.15 Advance Features

Following are some of the advanced features of PHP:

## 1. Uploading an Image file:

- \$target\_dir = "uploads/" - specifies the directory where the file is going to be placed.
- \$target\_file specifies the path of the file to be uploaded.
- \$uploadOk=1 is not used yet (will be used later).
- \$imageFileType holds the file extension of the file (in lower case).

Example

```
<?php  
  
$target_dir = "uploads/";  
  
$target_file = $target_dir .basename($_FILES["fileToUpload"]["name"]);  
  
$uploadOk = 1;  
  
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));  
  
if(isset($_POST["submit"])) {  
  
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);  
  
    if($check !== false) {  
  
        echo "File is an image - " . $check["mime"] . ".";  
  
        $uploadOk = 1;  
  
    } else {  
  
        echo "File is not an image.";  
  
        $uploadOk = 0;  
  
    }  
  
}  
  
?>
```

## 2. Validate IPv6 Address

The following example uses the filter\_var() function to check if the variable \$ip is a valid IPv6 address:

```
<?php  
  
$ip = "2001:0db8:85a3:08d3:1319:8a2e:0370:7334";  
  
if (!filter_var($ip, FILTER_VALIDATE_IP, FILTER_FLAG_IPV6) === false) {
```

```

echo("$ip is a valid IPv6 address");

} else {

echo("$ip is not a valid IPv6 address");

}

?>

```

#### 4.16.1 Cookies

- HTTP is a stateless protocol; cookies allow us to track the state of the application using small files stored on the user's computer. The path where the cookies are stored depends on the browser.
- A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, we can both create and retrieve cookie values.

Syntax: setcookie(name, value, expire, path, domain, secure, http only);

#### Example

```

<?php

$cookie_name = "user";
$cookie_value = "Amit";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>

<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

```

```
</body>  
</html>
```

#### 4.16.2 Session

- A session is a global variable stored on the server.
- Each session is assigned a unique id which is used to retrieve stored values.
- Whenever a session is created, a cookie containing the unique session id is stored on the user's computer and returned with every request to the server. If the client browser does not support cookies, the unique PHP session id is displayed in the URL.
- Sessions have the capacity to store relatively large data compared to cookies.
- The session values are automatically deleted when the browser is closed. If we want to store the values permanently, then we should store them in the database.
- Just like the `$_COOKIE` array variable, session variables are stored in the `$_SESSION` array variable. Just like cookies, the session must be started before any HTML tags.

#### Example

```
<?php  
// Start the session  
session_start();  
?  
<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
// Set session variables  
$_SESSION["favcolor"] = "green";  
$_SESSION["favanimal"] = "cat";  
echo "Session variables are set.";  
?>
```

```
</body>
```

```
</html>
```

#### 4.17 Object Oriented Programming with PHP

OOP stands for Object-Oriented Programming. Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions.

Object-oriented programming has several advantages over procedural programming:

- OOP is faster and easier to execute.
- OOP provides a clear structure for the programs.
- OOP helps to keep the PHP code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug.
- OOP makes it possible to create full reusable applications with less code and shorter development time.

From PHP5, we can also write PHP code in an object-oriented style.

##### Example

```
<?php  
  
class Fruit {  
  
    public $name;  
  
    public $color;  
  
    public function __construct($name, $color) {  
  
        $this->name = $name;  
  
        $this->color = $color;  
  
    }  
  
    public function intro() {  
  
        echo "The fruit is {$this->name} and the color is {$this->color}.";  
  
    }  
  
}  
  
class Strawberry extends Fruit {  
  
    public function message() {
```

```
echo "Am I a fruit or a berry? ";
}

}

$strawberry = new Strawberry("Strawberry", "red");
$strawberry->message();
$strawberry->intro();

?>
```

Chameli Devi Group of Institutions, Indore

Department of Artificial Intelligence And Data Science

Subject Notes

AD505- Internet and Web Technology

UNIT-V

Syllabus: PHP and MySQL: Basic commands with PHP examples, Connection to the server, creating the database, selecting a database, listing database, listing table names, creating a table, inserting data,

altering tables, queries, deleting the database, deleting data and tables, phpMyAdmin and database bugs.

### 5.1 BASIC COMMANDS WITH PHP EXAMPLES:

There are a lot of PHP commands available for use in various environment, especially for preparing one web application or embedding the entire server-side codebase with HTML syntax, and very easy to learn for the normal developer. Basics of PHP with commands are mentioned below:

#### 1. PHP Variables:

- **Types of Variables:** Variable always played important role in any kind of programming language. PHP also uses the declaration of the variable for assigning the value. One of the key features of the PHP variable is, it is not required to declare the type of the variable. As PHP is a weakly typed language, declare variable considering type based on the assigned value. PHP normally accepted variety types of any variable like string, integer, float, boolean, object, resource, array or NULL.
- **Name of the Variable:** Variable name in PHP always starts with \$, followed by any text or specific letter and \_. PHP variable name is case sensitive, so any capital letter variable with the same name should be considered as a new variable.
- **Scope of the variable:** Maximum variables are in the local scope. Variable declared inside the function are not available out of the function, on the same approach variable declared outside of the function are not available inside the function. It is possible to declare a global variable in PHP, in that case, need to declare that variable as global specifically, or accessing the same through the global array.

#### 2. PHP Operators:

- **Operator for assignments:** PHP normally uses one common operator for the assignment which is equal to ('='). Left of this equal sign is the variable name and right will be the assigned value.
- **Operators for arithmetic operation:** Below operators are used for performing an arithmetic operation in PHP. Operators are '+', '-', '\*', '/', '%', '++', '-'.
- **Operators for combination:** It is basically a combination of arithmetic operator and assignment operator. Combined operators are '+=' , '-=' , '\*=' , '/=' , '%=' .
- **Operators for comparison:** Comparison operators are '==' , '!=', '>', '>=' , '<' , '<=' .

- Operator for logical expression: Logical operators in PHP are ‘||’, ‘&&’, ‘and’, ‘or’, ‘xor’, ‘!’.

### 3. PHP If Else:

- Conditional Judgement: For any kind of conditional requirement in the programming logic PHP uses the ‘if else’ feature like any other programming language. The basic syntax of the ‘IF ELSE’ statement for PHP is:

```
IF [SPECIFIC CONDITION] {  
[CODE]} ELSE IF [SPECIFIC CONDITION 2] {  
[CODE]} ELSE {  
[CODE]}
```

### 4. PHP Switch:

PHP is using switch case as well, like other programming languages for avoiding the nested definition of multiple ‘IF ELSE’. Switch case considering multiple numbers of cases, and defining default are optional. The code structure of defining switch case is like below:

```
SWITCH($var) {  
CASE 'val 1'  
[CODE] Break;  
CASE 'val 2'  
[CODE] Break;  
CASE 'val 3'  
[CODE] Break;  
DEFAULT  
[CODE]}
```

### 5. PHP Loops:

- While Loop: in PHP, while loop can be executed till the mention expression is considering as true.

```
WHILE [condition or expression] {
```

```
[CODE]
```

```
}
```

- For Loop: For loop is used to execute the same code for the mention number of times.

```
FOR (exp 1, exp 2, exp 3) {
```

```
[CODE]
```

```
}
```

- Do While Loop: Similar to the while loop, the code will be executed until the true value is in while expression. The main difference with while is, the code mention inside the do at least execute one whether the expression is true or not, but while not ensure the same.

```
DO {
```

```
[CODE]
```

```
} WHILE (condition)
```

- FOREACH Loop: This loop is accepting an array as a variable and considering of executing code till the last element of the array.

```
FOREACH ($arr_var as $val) {
```

```
[CODE]
```

```
}
```

## Introduction to MySQL

- MySQL is a database system used on the web.
- MySQL is a database system that runs on a server.
- MySQL is ideal for both small and large applications.
- MySQL is very fast, reliable, and easy to use.
- MySQL uses standard SQL MySQL compiles on several platforms.
- MySQL is free to download and use.
- MySQL is developed, distributed, and supported by Oracle Corporation.
- The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

## 5.2 CONNECTION TO SERVER

In PHP user can easily do this using the `mysqli_connect()` function. All communication between PHP and the MySQL database server takes place through this connection. Here're the basic syntaxes for connecting to MySQL using MySQLi and PDO extensions:

Syntax: MySQLi, Procedural way

```
$link = mysqli_connect("hostname", "username", "password", "database");
```

Syntax: MySQLi, Object Oriented way

```
$mysqli = new mysqli("hostname", "username", "password", "database");
```

Syntax: PHP Data Objects (PDO) way

```
$pdo = new PDO("mysql:host=hostname;dbname=database", "username", "password");
```

The `hostname` parameter in the above syntax specify the hostname (e.g. `localhost`), or IP address of the MySQL server, whereas the `username` and `password` parameters specify the credentials to access the MySQL server, and the `database` parameter, if provided will specify the default MySQL database to be used when performing queries.

```
<?php  
/* Attempt MySQL server connection. Assuming user is running MySQL server with default setting */  
$link = mysqli_connect("localhost", "root", ""); // Check connection  
if($link === false){ die("ERROR: Could not connect. " . mysqli_connect_error()); } // Print host information  
  
echo "Connect Successfully. Host info: " . mysqli_get_host_info($link);  
?>
```

## 5.3 CREATING DATABASE, SELECTING A DATABASE, LISTING DATABASE

### 5.3.1 CREATE DATABASE

In this statement, user will execute the SQL query by passing it to the PHP `mysqli_query()` function to create our database.

```
<?php /* Attempt MySQL server connection. Assuming user is running MySQL server with default setting */
```

```
$link = mysqli_connect("localhost", "root", ""); // Check connection
```

```
if($link === false){ die("ERROR: Could not connect. " .
```

```
mysqli_connect_error()); }

$sql = "CREATE DATABASE demo"; // Attempt create database query execution

if(mysqli_query($link, $sql)){ echo "Database created successfully"; } else{ echo "ERROR: Could not able
to execute $sql. " . mysqli_error($link); } // Close connection mysqli_close($link);

?>
```

### 5.3.2 SELECT A DATABASE

Below script represents a code to select a database:

```
<?php

$username="root"; //MySQL User Name
$password=""; //MySQL User Password
$database= "myDB"; //MySQL Database Name
$conn=mysqli_connect("localhost",$username,$password,$database);

?>
```

### 5.3.3 LISTING DATABASES

Below represents a PHP PDO script to display all the databases present:

```
<?php

require "config.php"; //Database Connection

$result = $dbo->query("SHOW DATABASES");

while ($row = $result->fetch(PDO::FETCH_NUM))

{
```

```
echo $row[0]."<br>";  
}
```

```
?>
```

#### 5.4 LISTING TABLE NAMES

Following specifies the steps to list the table names in a database:

1. Connect to MySQL using the PDO object. In this example, database “test” is used.
2. Create our SQL statement, which is “SHOW TABLES”. This SQL statement tells MySQL to return a list of the tables that exist in our currently selected database.
3. Prepare the SQL statement.
4. Execute the SQL statement.
5. Fetch the results using fetchAll.
6. Loop through the results and print out the name of each table.

#### Example

```
<?php  
  
require "config.php"; //Database connection  
  
$result = $dbo->query("SHOW TABLES");  
  
while ($row = $result->fetch(PDO::FETCH_NUM))  
  
{  
echo $row[0]."<br>";  
}  
  
?>
```

## 5.5 CREATING A TABLE

CREATE TABLE statement is used to create a table in MySQL. Below example creates a table name "MyGuests":

```
<?php
```

```
$sql = "CREATE TABLE MyGuests (  
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
    firstname VARCHAR(30) NOT NULL,  
    lastname VARCHAR(30) NOT NULL,  
    email VARCHAR(50),  
    reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP  
);";
```

```
if (mysqli_query($conn, $sql)) {  
    echo "Table MyGuests created successfully";  
} else {  
    echo "Error creating table: " . mysqli_error($conn);  
}
```

```
?>
```

The data type specifies what type of data the column can hold. After the data type, user can specify other optional attributes for each column:

- NOT NULL - Each row must contain a value for that column, null values are not allowed.
- DEFAULT value - Set a default value that is added when no other value is passed.
- UNSIGNED - Used for number types, limits the stored data to positive numbers and zero.
- AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added.

- PRIMARY KEY - Used to uniquely identify the rows in a table. The column with the PRIMARY KEY setting is often an ID number and is often used with AUTO\_INCREMENT.

## 5.6 INSERTING DATA

After a database and a table have been created, user can start adding data to them.

Here are some syntax rules to follow:

- The SQL query must be quoted in PHP.
- String values inside the SQL query must be quoted.
- Numeric values must not be quoted.
- The word NULL must not be quoted.

The INSERT INTO statement is used to add new records to a MySQL table as follows:

```
<?php

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Aman', 'Singh', 'amansingh@example.com');

if (mysqli_query($conn, $sql)) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

?>
```

## 5.7 ALTERING TABLES

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

To add a column in a table, use the following syntax:

```
ALTER TABLE table_name ADD column_name datatype;
```

Example

```
<?php  
$sql = "ALTER TABLE MyGuests ADD DOB varchar(255);  
  
if (mysqli_query($conn, $sql)) {  
    echo "Table altered successfully";  
} else {  
    echo "Error: " . mysqli_error($conn);  
}  
?>
```

## 5.8 QUERIES

The mysqli\_query() function performs a query against the database.

Syntax: mysqli\_query(connection,query,resultmode);

Example

```
<?php  
  
$con=mysqli_connect("localhost","my_user","my_password","my_db");  
// Check connection  
if (mysqli_connect_errno())  
{  
    echo "Failed to connect to MySQL: " . mysqli_connect_error();  
}  
// Perform queries  
mysqli_query($con,"SELECT * FROM Persons");  
mysqli_query($con,"INSERT INTO Persons (FirstName,LastName,Age)  
VALUES ('Glenn','Quagmire',33)");  
mysqli_close($con);  
  
?>
```

## 5.9 DELETING DATABASE

The DROP DATABASE statement is used to delete a database.

Example

```
<?php
```

```
$sql = " DROP DATABASE myDB";  
if (mysqli_query($conn, $sql)) {  
    echo "Database deleted successfully";  
} else {  
    echo "Error deleting database !!"; mysqli_error($conn);  
}  
mysqli_close($conn);
```

```
?>
```

## 5.10 DELETING DATA AND TABLES

### 5.10.1 Deleting Data

Following steps can be performed to delete table data:

- Connect to the MySQL database by creating a new instance of the PDO object.
- Construct a DELETE statement to delete a row, multiple rows, or all rows in a table.
- Execute the DELETE statement by calling the exec() method of the PDO object or the execute() method of the PDOStatement object.

Example

```
<?php
```

```
$link = mysqli_connect("localhost", "root", "", "demo");  
  
if($link === false){ die("ERROR: Could not connect. " . mysqli_connect_error()); }
```

```

$sql = "DELETE FROM persons WHERE first_name='John'";

if(mysqli_query($link, $sql))

{ echo "Records were deleted successfully."; }

else{ echo "ERROR: Could not able to execute $sql. " . mysqli_error($link); }

mysqli_close($link);

?>

```

### 5.10.2 Deleting Tables

Following steps can be performed to delete tables:

- Connect to the MySQL database by creating a new instance of the PDO object.
- Construct a DROP statement to delete a table.
- Execute the DROP statement by calling the exec() method of the PDO object or the execute() method of the PDOStatement object.

Example

<?php

```

$sql = "DROP TABLE MyGuests";

if (mysqli_query($conn, $sql)) {
    echo "Table deleted successfully";
} else {
    echo "Error deleting table: ".mysqli_error($conn);
}

mysqli_close($conn);

```

?>

## 5.11 PHPMYADMIN AND DATABASE BUGS

### 5.11.1 PHPMYADMIN

phpMyAdmin is an application that is written in PHP. Its specific purpose is to enable users with the ability to interact with and administer MySQL servers. phpMyAdmin offers a graphical interface of the data, tables, and fields stored in the MySQL database for database administration tasks. phpMyAdmin is a free software tool written in PHP, intended to handle the administration of MySQL over the Web. phpMyAdmin supports a wide range of operations on MySQL and MariaDB. Frequently used operations (managing databases, tables, columns, relations, indexes, users, permissions, etc.) can be performed via the user interface.

Figure 5.1: phpMyAdmin Interface

Features of phpMyAdmin:

- Intuitive web interface
- Support for most MySQL features:
  - Browse and drop databases, tables, views, fields, and indexes
  - Create, copy, drop, rename and alter databases, tables, fields, and indexes
  - Maintenance server, databases, and tables, with proposals on server configuration
  - Execute, edit and bookmark any SQL statement, even batch-queries
  - Manage MySQL user accounts and privileges
  - Manage stored procedures and triggers
- Export data to various formats: CSV, SQL, XML, PDF, ISO/IEC 26300 - OpenDocument Text, Spreadsheet, Word, Latex etc.
- Administering multiple servers
- Creating graphics of your database layout in various formats
- Creating complex queries using Query-by-example (QBE)
- Searching globally in a database or a subset of it
- Transforming stored data into any format using a set of predefined functions, like displaying BLOB-data as image or download-link

### 5.11.2 DATABASE BUGS

PHP defines some constants that can be used to set the value of `error_reporting` such that only errors of certain types get reported: `E_ALL` (for all errors except strict notices), `E_PARSE` (parse errors), `E_ERROR` (fatal errors), `E_WARNING` (warnings), `E_NOTICE` (notices), and `E_STRICT` (strict notices).

While writing your PHP program, it is a good idea to use PHP-aware editors like BBEdit or Emacs. One of the special features of these editors is syntax highlighting. It changes the color of different parts of your program based on what those parts are. For example, strings are pink, keywords such as `if` and `while` are blue, comments are grey, and variables are black.

Another feature is quote and bracket matching, which helps to make sure that your quotes and brackets are balanced. When user types a closing delimiter such as `}`, the editor highlights the opening `{` that it matches.

There are the following points that need to be verified while debugging your program:

- Missing Semicolons – Every PHP statement ends with a semicolon (`;`). PHP doesn't stop reading a statement until it reaches a semicolon. If user leaves out the semicolon at the end of a line, PHP continues reading the statement on the following line.
- Not Enough Equal Signs – When user asks whether two values are equal in a comparison statement, user needs two equal signs (`==`). Using one equal sign is a common mistake.
- Misspelled Variable Names – If user misspelled a variable then PHP understands it as a new variable. Remember: To PHP, `$test` is not the same variable as `$Test`.
- Missing Dollar Signs – A missing dollar sign in a variable name is hard to see, but at least it usually results in an error message so that user knows where to look for the problem.
- Troubling Quotes – Users can have too many, too few, or the wrong kind of quotes. So, check for a balanced number of quotes.
- Missing Parentheses and curly brackets – They should always be in pairs.
- Array Index – All the arrays should start from zero instead of 1.

Moreover, handle all the errors properly and direct all trace messages into the system log file so that if any problem happens then it will be logged into the system log file and user will be able to debug that problem.

Following example shows one of the several ways to debug errors in PHP:

```
<?php
```

```
if (!$conn) {  
    die("Connection failed: " . mysqli_connect_error());  
}  
  
if (mysqli_query($conn, $sql)) {  
    echo "Record updated successfully";  
} else {  
    echo "Error updating record: " . mysqli_error($conn);  
}  
  
?>
```

**Chameli Devi Group of Institutions**

**Department of Information Technology**

**Subject Notes**

**IT405 – Database Management System**

**B.Tech, IT-4<sup>th</sup> Semester**

**Syllabus:** SQL: Data definition in SQL, update statements and views in SQL: Data storage and definitions, Data retrieval queries and update statements, Query Processing & Query Optimization: Overview, measures of query cost, selection operation, sorting, join, evaluation of expressions, transformation of relational expressions, estimating statistics of expression results, evaluation plans. Case Study of ORACLE and DB2.

**Course Objective:** Students can use SQL operations to manipulate the database and learn how to design and create a good database using functional dependencies and normalization.

**Course Outcome:** Analyze and renovate an information model into a relational database schema and to use a DDL, DML and DCL utilities to implement the schema using a DBMS.

---

---

## SQL Command

### DDL: Data Definition Language

All DDL commands are auto-committed. That means it saves all the changes permanently in the database.

Command	Description
Create	to create new table or database
Alter	for alteration
Truncate	delete data from a table
Drop	to drop a table
Rename	to rename a table

**Table 3.1: DDL Commands**

### DML: Data Manipulation Language

DML commands are not auto-committed. It means changes are not permanent to database, they can be rolled back.

Command	Description
Insert	to insert a new row
Update	to update existing row
Delete	to delete a row
Merge	merging two rows or two tables

**Table 3.2: DML Commands**

### TCL: Transaction Control Language

These commands are used to keep a check on other commands and their effect on the database. These commands can terminate changes made by other commands by rolling back to original state. It can also make changes permanent.

Command	Description
Commit	to permanently save

Rollback	to undo the change
save point	to save temporarily

**Table 3.3: TCL Commands**

### DCL: Data Control Language

Data control language provides a command to grant and take back authority.

Command	Description
Grant	grant permission of the right
Revoke	take back permission

**Table 3.4: DCL Commands**

### DQL: Data Query Language

Command	Description
Select	retrieve records from one or more table

**Table 3.5: DQL Commands**

### Create command

**create** is a DDL command used to create a table or a database.

#### Creating a Database

To create a database in RDBMS, *create* command is used.

**Syntax:** `create database database-name;`

**Example:** `create database Test;`

The above command will create a database named **Test**.

#### Creating a Table

*create* command is also used to create a table. We can specify names and datatypes of various columns along.

**Syntax:**

```
create table table-name
{
    column-name1 datatype1,
    column-name2 datatype2,
    column-name3 datatype3,
    column-name4 datatype4
};
```

create table command will tell the database system to create a new table with given table name and column information.

**Example:** create table Student (id int, name varchar, age int);

**alter command:** Used for alteration of table structures. There are various uses of *alter* command, such as,

- to add a column to the existing table
- to rename any existing column
- to change the datatype of any column or to modify its size
- to drop a column

Using alter command we can add a column to an existing table.

**Syntax:** alter table *table-name* add (*column-name* datatype);

**Example:** alter table Student add (address char);

*To add a column with Default Value*

alter command can add a new column to an existing table with default values.

**Syntax:** alter table *table-name* add (*column-name1* datatype1 default *data*);

**Example:** alter table Student add (dob date default '1-Jan-99');

**To Modify an existing Column:** Used to modify data type of an existing column.

**Syntax:** alter table *table-name* modify (*column-name* datatype);

**Example:** alter table Student modify (address varchar (30));

**To Rename a column:** Using alter command user can rename an existing column.

**Syntax:** alter table table-name rename old-column-name to column-name;

**Example:** alter table Student rename address to Location;

**To Drop a Column:** Used to drop columns also.

**Syntax:** alter table table-name drop(column-name);

**Example:** alter table Student drop(address);

#### **truncate command**

The truncate command removes all records from a table. But this command will not destroy the table's structure. When we apply truncate command on a table its Primary key is initialized.

**Syntax:** truncate table table-name

**Example:** truncate table Student;

#### **drop command**

drop query completely removes a table from the database. This command will also destroy the table structure.

**Syntax:** drop table table-name

**Example:** drop table Student;

#### **rename command**

rename command is used to rename a table.

**Syntax:** rename table old-table-name to new-table-name

**Example:** rename table Student to Student-record;

### **DML Commands**

#### **1) INSERT command**

Insert command is used to insert data into a table.

**Syntax:** INSERT into table-name values (data1, data2,)

**Example:**

Consider a table Student with following fields.

S_id	S_Name	age
------	--------	-----

INSERT into Student values(101,'Adam',15);

The above command will insert a record into Student table.

S_id	S_Name	age
101	Adam	15

## 2) UPDATE command

Update command is used to update a row of a table.

**Syntax:** UPDATE table-name set column-name = value where condition;

**Example:**

update Student set age=18 where s\_id=102;

S_id	S_Name	age
101	Adam	15
102	Alex	18
103	chris	14

**Example:**

UPDATE Student set s\_name='Abhi', age=17 where s\_id=103;

The above command will update two columns of a record.

S_id	S_Name	age
101	Adam	15
102	Alex	18
103	Abhi	17

## 3) Delete command

Delete command is used to delete data from a table. Delete command can also be used with the condition to delete a particular row.

**Syntax:** DELETE from table-name;

**Example:** DELETE from Student;

The above command will delete all the records from Student table.

### **TCL command**

Transaction Control Language (TCL) commands are used to manage transactions in the database. These are used to manage the changes made by DML statements. It also allows statements to be grouped together into logical transactions.

#### **Commit command**

Commit command is used to permanently save any transaction into the database.

**Syntax:** commit;

#### **Rollback command**

This command restores the database to last committed state. It is also used with savepoint command to jump to a save point in a transaction.

**Syntax:** rollback to save point-name;

#### **Savepoint command**

savepoint command is used to temporarily save a transaction so that you can rollback to that point whenever necessary.

**Syntax:** savepoint savepoint-name;

### **DCL command**

System: creating a session, table etc. are all types of system privilege.

Object: any command or query to work on tables comes under object privilege.

DCL defines two commands,

Grant: Gives user access privileges to the database.

Revoke: Take back permissions from the user.

**Example:** grant create session to username;

S_id	s_Name	Age	Address
101	Adam	15	Noida
102	Alex	18	Delhi
103	Abhi	17	Rohtak
104	Ankit	22	Panipat

**Table 3.6: DCL Command Example**

#### **WHERE clause**

Where clause is used to specify condition while retrieving data from the table. Where clause is used mostly with Select, Update and Delete query. If the condition specified by where clause is true then only the result from the table is returned.

#### **Syntax:**

SELECT column-name1, column-name2, column-name3, column-name N

from table-name WHERE [condition];

**Example:** SELECT s\_id, s\_name, age, address from Student WHERE s\_id=101;

#### **SELECT Query**

The Select query is used to retrieve data from a table. It is the most used SQL query. We can retrieve complete tables, or partial by mentioning conditions using WHERE clause.

#### **Syntax:**

SELECT column-name1, column-name2, column-name3, column-nameN from table-name;

**Example:** SELECT s\_id, s\_name, age from Student.

#### **Like clause**

Like clause is used as a condition in SQL query. Like clause compares data with an expression using wildcard operators. It is used to find similar data from the table.

#### *Wildcard operators*

There are two wildcard operators that are used in like clause.

Percent sign % represents zero, one or more than one character.

Underscore sign \_ represents only one character.

**Example:** SELECT \* from Student where s\_name like 'A%';

#### **Order by Clause**

Order by clause is used with a Select statement for arranging retrieved data in sorted order. The Order by clause by default sort data in ascending order. To sort data in descending order DESC keyword is used with Order by clause.

**Syntax:** SELECT column-list|\* from table-name order by asc|desc;

**Example:** SELECT \* from Emp order by salary;

#### **Group by Clause**

Group by clause is used to group the results of a SELECT query based on one or more columns. It is also used with SQL functions to group the result from one or more tables.

#### **Syntax:**

SELECT column\_name, function (column\_name)

FROM table\_name

WHERE condition

GROUP BY column\_name

Example select name, salary

from Emp

where age > 25

group by salary

## **HAVING Clause**

Having clause is used with SQL Queries to give a more precise condition for a statement. It is used to mention condition in Group based SQL functions, just like WHERE clause.

### **Syntax:**

```
select column_name, function(column_name)  
FROM table_name  
WHERE column_name condition  
GROUP BY column_name  
HAVING function (column_name) condition
```

**Example** SELECT \*

```
from sale group customer  
having sum(previous_balance) > 3000
```

## **Distinct keyword**

The distinct keyword is used with a Select statement to retrieve unique values from the table. Distinct removes all the duplicate records while retrieving from the database.

**Syntax:** SELECT distinct column-name from table-name;

**Example:** select distinct salary from Emp;

## **AND & OR operator**

AND & OR operators are used with Where clause to make more precise conditions for fetching data from database by combining more than one condition together.

**Example:**

```
SELECT * from Emp WHERE salary < 10000 AND age > 25
```

```
SELECT * from Emp WHERE salary > 10000 OR age > 25
```

## **SQL Constraints**

SQL Constraints are rules used to limit the type of data that can go into a table, to maintain the accuracy and integrity of the data inside the table.

Constraints can be divided into following two types,

**Column level constraints: limits only column data**

**Table-level constraints: limits whole table data**

Constraints are used to make sure that the integrity of data is maintained in the database.

Following are the most used constraints that can be applied to a table:

#### **NOT NULL Constraint**

NOT NULL constraint restricts a column from having a NULL value. Once NOT NULL constraint is applied to a column, you cannot pass a null value to that column.

**Example:** CREATE table Student (s\_id int NOT NULL, Name varchar (60), Age int);

#### **UNIQUE Constraint**

UNIQUE constraint ensures that a field or column will only have unique values. A UNIQUE constraint field will not have duplicate data.

**Example:** CREATE table Student (s\_id int NOT NULL UNIQUE, Name varchar (60), Age int);

#### **Primary Key Constraint**

Primary key constraint uniquely identifies each record in a database. A Primary Key must contain unique value and it must not contain a null value.

**Example:** CREATE table Student (s\_id int PRIMARY KEY, Name varchar (60) NOT NULL, Age int);

#### **Foreign Key Constraint**

FOREIGN KEY is used to relate two tables. The foreign KEY constraint is also used to restrict actions that would destroy links between tables.

**Example:** CREATE table Order\_Detail (order\_id int PRIMARY KEY, order\_name varchar (60) NOT NULL, c\_id int FOREIGN KEY REFERENCES Customer\_Detail(c\_id));

#### **On Delete Cascade:**

This will remove the record from the child table if that value of the foreign key is deleted from the main table.

**On Delete Null:** This will set all the values in that record of child table as NULL, for which the value of the foreign key is deleted from the main table.

### CHECK Constraint

A check constraint is used to restrict the value of a column between a range. It performs check on the values, before storing them into the database. It's like condition checking before saving data into a column.

**Example:** create table Student (s\_id int NOT NULL CHECK (s\_id > 0), Name varchar (60) NOT NULL,Age int);

### SQL Functions

SQL provides many built-in functions to perform operations on data. These functions are useful while performing mathematical calculations, string concatenations, sub-strings etc. SQL functions are divided into two categories:

**Aggregate Functions:** These functions return a single value after calculating from a group of values.  
Aggregate functions includes: MAX(), MIN(), SUM(), AVG () and COUNT ()

**Scalar Functions:** Scalar functions return a single value from an input value.

UCASE (): Used to convert the value of string column to the uppercase character.

### Join in SQL

SQL Join is used to fetch data from two or more tables, which is joined to appear as a single set of data. SQL Join is used for combining column from two or more tables by using values common to both tables. Join Keyword is used in SQL queries for joining two or more tables. Minimum required condition for joining table is (n-1) where n, is a number of tables. A table can also join to itself known as, Self-Join.

Consider the following two tables:

**Table 1:** CUSTOMERS Table

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00

2   Khilan   25   Delhi   1500.00
3   kaushik   23   Kota   2000.00
4   Chaitali   25   Mumbai   6500.00
5   Hardik   27   Bhopal   8500.00
6   Komal   22   MP   4500.00
7   Muffy   24   Indore   10000.00

**Table 2:** ORDERS Table

OID	DATE	CUSTOMER_ID	AMOUNT
102   2009-10-08 00:00:00   3   3000			
100   2009-10-08 00:00:00   3   1500			
101   2009-11-20 00:00:00   2   1560			
103   2008-05-20 00:00:00   4   2060			

Now, let us join these two tables in our SELECT statement as shown below

```
SQL> SELECT ID, NAME, AGE, AMOUNT FROM CUSTOMERS, ORDERS WHERE CUSTOMERS.ID =
ORDERS.CUSTOMER_ID;
```

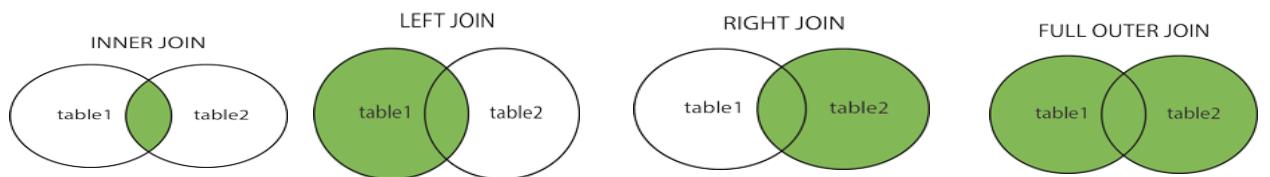
This query will produce below result

ID	NAME	AGE	AMOUNT
3   kaushik   23   3000			
3   kaushik   23   1500			
2   Khilan   25   1560			
4   Chaitali   25   2060			

Here, it is noticeable that the join is performed in the WHERE clause. Several operators can be used to join tables, such as =, <, >, <>, <=, >=, !=, BETWEEN, LIKE, and NOT; they can all be used to join tables. However, the most common operator is the equal to symbol.

Below are the different types of Join available in SQL:

- [INNER JOIN](#) returns rows when there is a match in both tables.
- [LEFT JOIN](#) returns all rows from the left table, even if there are no matches in the right table.
- [RIGHT JOIN](#) returns all rows from the right table, even if there are no matches in the left table.
- [FULL JOIN](#) returns rows when there is a match in one of the tables.
- [SELF JOIN](#) is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
- [CARTESIAN JOIN](#) returns the Cartesian product of the sets of records from the two or more joined tables.



**Figure 3.1: Join Concept**

**Example:**

Orders

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID

Customers

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country

**Table 3.7: Join Command Example**

**Inner Join:**

1. 

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```
2. 

```
SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);
```

**Left Join:**

```
SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID  
ORDER BY Customers.CustomerName;
```

**Right Join:**

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName  
FROM Orders  
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID  
ORDER BY Orders.OrderID;
```

**Full Outer Join:**

```
SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID  
ORDER BY Customers.CustomerName;
```

**Self-Join:**

```
SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City  
FROM Customers A, Customers B  
WHERE A.CustomerID <> B.CustomerID  
AND A.City = B.City  
ORDER BY A.City;
```

**Query Processing and Optimization****Query Processing**

Query Processing is a procedure of transforming a high-level query (such as SQL) into a correct and efficient execution plan expressed in low-level language. A query processing selects a most appropriate plan that is used in responding to a database request. When a database system receives a query for update or retrieval of information, it goes through a series of compilation steps, called execution plan.

### **Different phases of Query Processing are:**

- In the first phase called syntax checking phase, the system parses the query and checks that it follows the syntax rules or not.
- It then matches the objects in the query syntax with the view tables and columns listed in the system table.
- Finally, it performs the appropriate query modification. During this phase, the system validates the user privileges and that the query does not disobey any integrity rules.
- The execution plan is finally executing to generate a response.

### **Query Analyzer**

- The syntax analyzer takes the query from the users, parses it into tokens and analyses the tokens and their order to make sure they follow the rules of the language grammar.
- If an error is found in the query submitted by the user, it is rejected and an error code together with an explanation of why the query was rejected is returned to the user.

A simple form of the language grammar that could be used to implement SQL statement is given below:

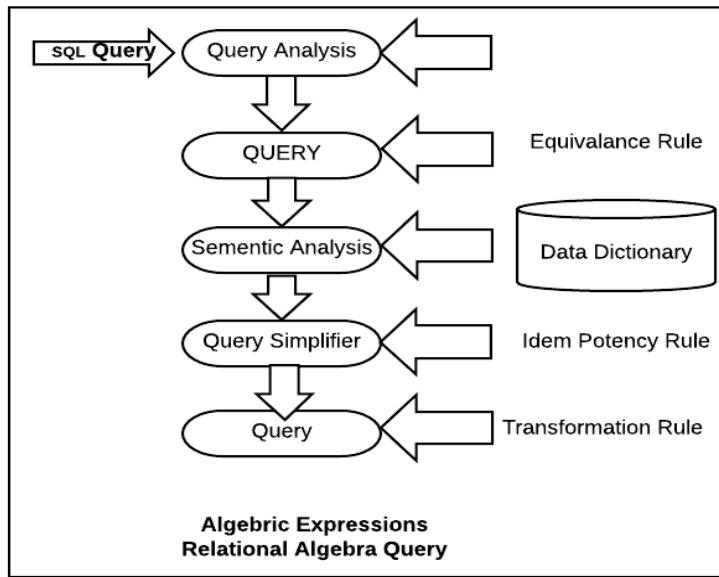
- QUERY = SELECT + FROM + WHERE
- SELECT = 'SELECT' + <COLUMN LIST>
- FROM = 'FROM' + <TABLE LIST>
- WHERE = 'WHERE' + VALUE1 OP VALUE2
- VALUE1 = VALUE / COLUMN NAME
- VALUE2 = VALUE / COLUMN NAME
- OP = >, <, >=, <=, =, <>

### **Query Decomposition**

The query decomposition is the first phase of the query processing whose aims are to transfer the high-level query into a relational algebra query and to check whether that query is syntactically and semantically correct.

- Thus, the query decomposition starts with a high-level query and transforms it into a query graph of low-level operations, which satisfy the query.

- The SQL query is decomposed into query blocks (low-level operations), which form the basic unit.
- Hence nested queries within a query are identified as separate query blocks.
- The query decomposer goes through five stages of processing for decomposition into low-level operation and translation into algebraic expressions.



**Figure 3.2: Phases of Query Processing**

### 1) Query Analysis:

- During the query analysis phase, the query is syntactically analyzed using the programming language compiler (parser).
- A syntactically legal query is then validated, using the system catalog, to ensure that all data objects (relations and attributes) referred to by the query are defined in the database.
- The type specification of the query qualifiers and result is also checked at this stage.

Let us consider the following query :

```
SELECT emp_nm FROM EMPLOYEE WHERE emp_desg>100
```

This query will be rejected because the comparison “>100” is incompatible with the data type of **emp\_desg** which is a variable character string.

### Query Tree Notations:

At the end of query analysis phase, the high-level query (SQL) is transformed into some internal representation that is more suitable for processing. This internal representation is typically a kind of query tree.

A Query Tree is a tree data structure that corresponds expression.

A Query Tree is also called a relational algebra tree.

- Leaf node of the tree, representing the base input relations of the query.
- Internal nodes of the tree representing an intermediate relation which is the result of applying an operation in the algebra.
- Root of the tree representing a result of the query.
- Sequence of operations is directed from **leaf to root**.

Query tree is executed by executing a

- The internal is then replaced by the resulting relation.
- The execution is terminated when the root relation for the query.

**Example:**

```
SELECT (P.proj_no, P.dept_no, E.name, E.add, E.dob)  
FROM PROJECT P, DEPARTMENT D, EMPLOYEE  
WHERE P.dept_no = D.d_no AND D.mgr_id = E.emp_id  
      AND P.proj_loc = 'Mumbai' ;
```

Mumbai-PROJ  $\leftarrow \sigma_{proj\_loc = 'Mumbai'}$  (PROJECT)

CONT-DEPT  $\leftarrow (Mumbai-PROJ \bowtie dept\_no = d\_no DEPARTMENT)$

PROJ-DEPT-MGR  $\leftarrow (CONT-DEPT \bowtie mgr\_id=emp\_id EMPLOYEE)$

RESULT  $\leftarrow \Pi_{proj\_no, dept\_no, name, add, dob}$  (PROJ-DEPT-MGR)

The three relations PROJECT, DEPARTMENT, EMPLOYEE is representing as a leaf nodes P, D and E, while the relational algebra operations of the represented by internal tree nodes.

- Same SQL query can have many different relational algebra expressions and hence many different query trees.
- The query parser typically generates a standard initial (canonical) query tree.

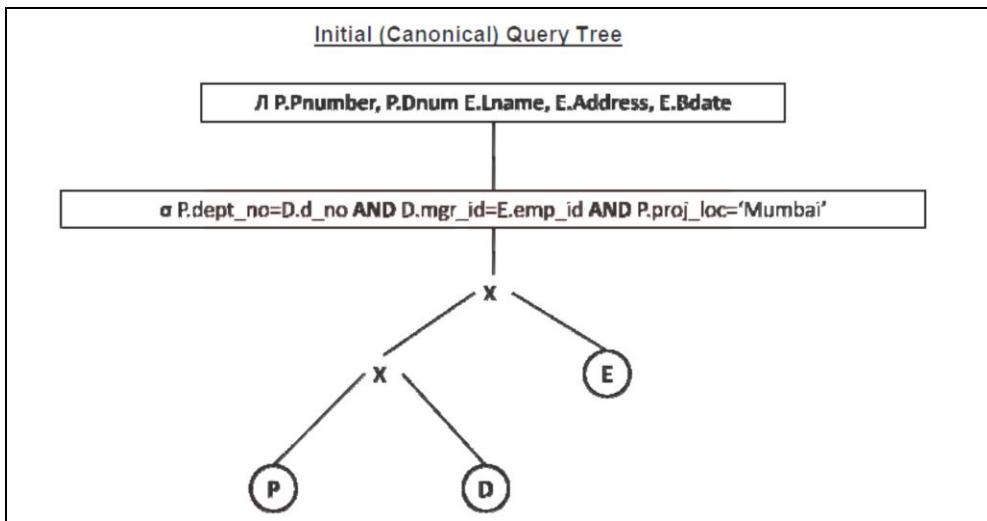


Figure 3.3: Query Tree Notation

#### Query Graph Notations:

- Query graph is sometimes also used for representation of a query. In query graph representation, the relations in the query are represented **relation nodes** are displayed as a SINGLE CIRCLE.
- The constant values from the query selection (`proj_loc = 'Mumbai'`) are represented by **constant node**, displayed as a DOUBLE CIRCLE.
- The **selection and join conditions** are represented as a **Graph Edge** (e.g. `P.dept_no = p.dept_num`).
- Finally, the attributes retrieve from the relation are displays in square brackets (`[P.proj_num, P.dept_no]`).

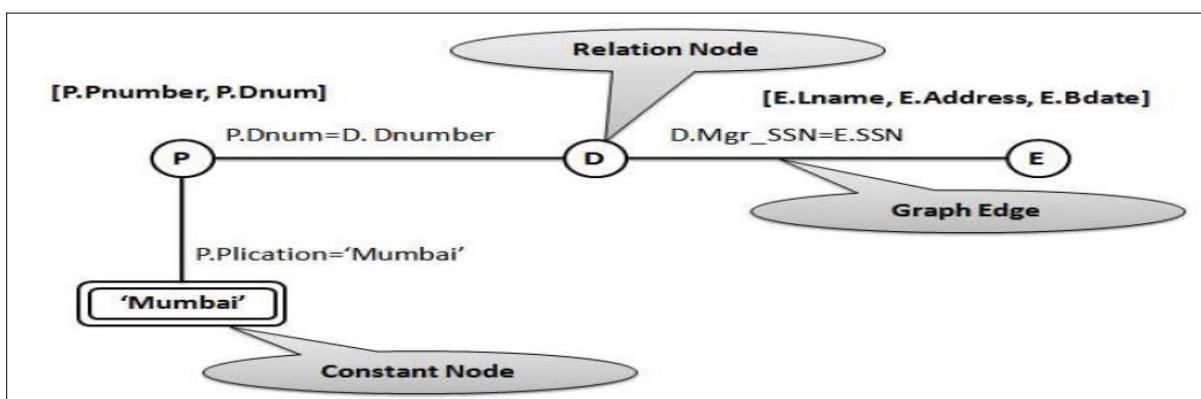


Figure 3.4: Query Graph

## 2) Query Normalization:

- The primary phase of the normalization is to avoid redundancy. The normalization phase converts the query into a normalized form that can be more easily manipulated.
- In the normalization phase, a set of equivalency rules are applied so that the projection and selection operations included on the query are simplified to avoid redundancy.
- The projection operation corresponds to the SELECT clause of SQL query and the selection operation correspond to the predicate found in WHERE clause.
- The equivalency transformation rules that are applied to SQL query is shown in following table, in which UNARYOP means UNARY operation, BINOP means BINARY operation and REL1, REL2, REL3 are the relations.

	Rule Name	Rule Description
1	Commutative of UNARY operation	UNARYOP1 UNARYOP2 REL $\leftrightarrow$ UNARYOP2 UNARYOP1 REL
2	Commutative of BINARY operation	REL1 BINOP (REL2 BINOP REL3) $\leftrightarrow$ (REL1 BINOP REL2) BINOP REL3
3.	Idempotency of UNARY operations UNARYOP1 UNARYOP2 REL	UNARYOP REL
4.	Distributivity of UNARY operations	UNARYOP1 (REL1 BINOP REL2) $\leftrightarrow$ UNARYOP (REL1) BINOP UNARYOP (REL2)
5	Factorization of UNARY operations	UNARYOP (REL1) BINOP UNARYOP (REL2) $\leftrightarrow$ UNARYOP (REL1 BINOP REL2)

Table 3.8: Query Normalization

## 3) Semantic Analyzer:

- The objective of this phase of query processing is to reduce the number of predicates.
- The semantic analyzer rejects the normalized queries that are incorrectly formulated.
- A query is incorrectly formulated if components do not contribute to the generation of the result. This happens in the case of missing join specification.
- A query is contradictory if its predicate cannot satisfy by any tuple in the relation. The semantic analyzer examines the relational calculus query (SQL) to make sure it contains only data objects that are table, columns, views, indexes that are defined in the database catalog.
- It makes sure that each object in the query is referenced correctly according to its data type.

- In the case of missing join specifications, the components do not contribute to the generation of the results, and thus, a query may be incorrect formulated.
- A query is contradictory if its predicates cannot be satisfied by any of the tuples.

For example:

( emp\_des = 'Programmer'  $\wedge$  emp\_des = 'Analyst' )

As an employee cannot be both 'Programmer' and 'Analyst' simultaneously, the above predicate on the EMPLOYEE relation is contradictory.

#### **Example of Correctness and Contradictory:**

Incorrect Formulated: - (Missing Join Specification)

```
SELECT p.projno, p.proj_location
FROM project p, viewing v, department d
WHERE d.dept_id = v.dept_id AND d.max_budj >= 8000
AND d.mgr = 'Methew'
```

Contradictory: - (**Predicate cannot satisfy**)

```
SELECT p.proj_no, p.proj_location
FROM project p, cost_proj c, department d
WHERE d.max_budj > 80000 AND d.max_budj < 50000 AND
d.dept_id = v.dept_id AND v.proj_no = p.proj_no
```

#### **4. Query Simplifier:**

The objectives of this phase are to detect redundant qualification, eliminate common sub-expressions and transform sub-graph to semantically equivalent but easier and efficiently computed form.

Commonly integrity constraints view definitions, and access restrictions are introduced into the graph at this stage of analysis so that the query must be simplified as much as possible.

Integrity constraints define constants which must hold for all state of the database, so any query that contradicts integrity constraints must be avoided and can be rejected without accessing the database.

The final form of simplification is obtaining by applying **idempotency rules** of Boolean algebra.

	Description	Rule Format
1.	PRED AND PRED = PRED	$P \wedge (P) = P$
2.	PRED AND TRUE = PRED	$P \vee \text{TRUE} = P$
3.	PRED AND FALSE = FALSE	$P \wedge \text{FALSE} = \text{FALSE}$
4.	PRED AND NOT(PRED) = FALSE	$P \wedge (\sim P) = \text{FALSE}$
5.	PRED1 AND (PRED1 OR PRED2) = PRED1	$P_1 \wedge (P_1 \vee P_2) = P_1$
6.	PRED OR PRED = PRED	$P \vee (P) = P$
7.	PRED OR TRUE = TRUE	$P \vee \text{TRUE} = \text{TRUE}$
8.	PRED OR FALSE = PRED	$P \vee \text{FALSE} = P$
9.	PRED OR NOT(PRED) = TRUE	$P \vee (\sim P) = \text{TRUE}$
10.	PRED1 OR (PRED1 AND PRED2) = PRED1	$P_1 \vee (P_1 \wedge P_2) = P_1$

Table 3.9: Boolean Algebra Rules

### 5) Query Restructuring:

- In the final stage of the query decomposition, the query can be restructured to give a more efficient implementation.
- Transformation rules are used to convert one relational algebra expression into an equivalent form that is more efficient.
- The query can now be regarded as a relational algebra program, consisting of a series of operations on relation.

### Query Optimization

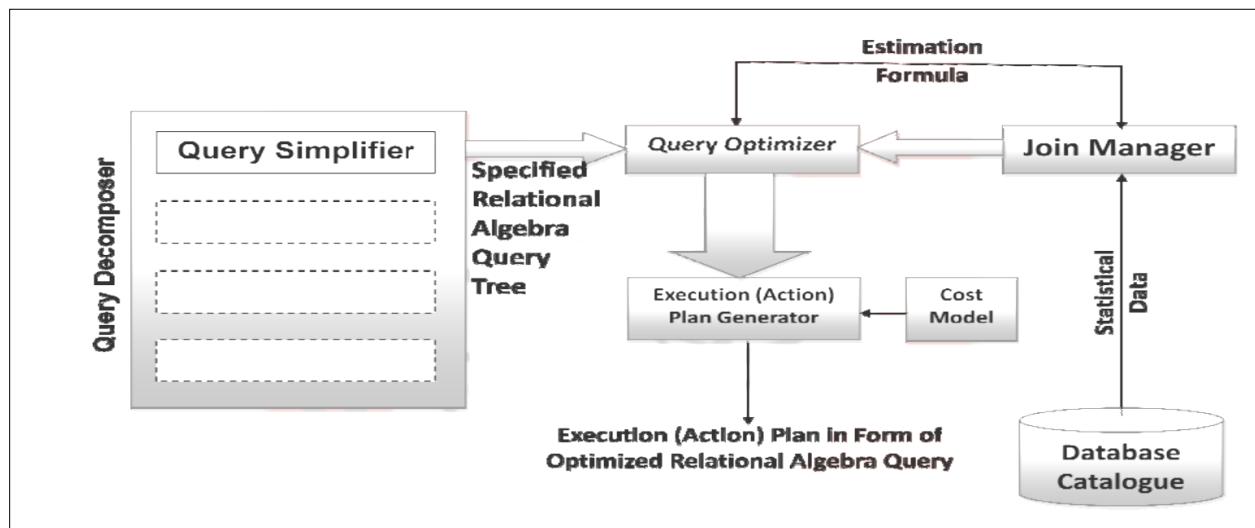


Figure 3.5: Query Optimization Phases

The primary goal of query optimization is of choosing an efficient execution strategy for processing a query.

- The query optimizer attempts to minimize the use of certain resources (mainly the number of I/O and CPU time) by selecting a best execution plan (access plan).
- A query optimization starts during the validation phase by the system to validate the user has appropriate privileges.
- Now an action plan is generated to perform the query.

Relational algebra query tree generated by the query simplifier module of query decomposer.

- Estimation formulas used to determine the cardinality of the intermediate result table.
- A cost Model
- Statistical data from the database catalogue.

The output of the query optimizer is the execution plan in form of optimized relational algebra query.

- A query typically has many possible execution strategies, and the process of choosing a suitable one for processing a query is known as Query Optimization.
- The basic issues in Query Optimization are:
  - How to use available indexes.
  - How to use memory to accumulate information and perform immediate steps such as sorting.
  - How to determine the order in which joins should be performed.
- The term query optimization does not mean giving always an optimal (best) strategy as the execution plan.
- It is just a reasonably efficient strategy for execution of the query.
- The decomposed query block of SQL is translating into an equivalent extended relational algebra expression and then optimized.

**There are two main techniques for implementing Query Optimization:**

The **first** technique is based on **Heuristic Rules** for ordering the operations in a query execution strategy. The **second** technique involves the **systematic estimation** of the cost of the different execution strategies and choosing the execution plan with the *lowest cost*.

- Semantic query optimization is used with the combination with the heuristic query transformation rules.

- It uses constraints specified on the database schema such as unique attributes and other more complex constraints, in order to modify one query into another query that is more efficient to execute.

### **1. Heuristic Rules:**

- The heuristic rules are used as an optimization technique to modify the internal representation of the query. Usually, heuristic rules are used in the form of query tree or query graph data structure, to improve its performance.
- One of the main heuristic rules is to apply SELECT operation before applying the JOIN or other BINARY operations.
- This is because the size of the file resulting from a binary operation such as JOIN is usually a multi-value function of the sizes of the input files.
- The SELECT and PROJECT reduced the size of the file and hence, should be applied before the JOIN or other binary operation.
- Heuristic query optimizer transforms the initial (canonical) query tree into final query tree using equivalence transformation rules. This final query tree is efficient to execute.

For example, consider the following relations:

Employee (EName, EID, DOB, EAdd, Sex, ESalary, EDeptNo)

Department (DeptNo, DeptName, DeptMgrID, Mgr\_S\_date)

DeptLoc (DeptNo, Dept\_Loc)

Project (ProjName, ProjNo, ProjLoc, ProjDeptNo)

WorksOn (E-ID, P-No, Hours)

Dependent (E-ID, DependName, Sex, DDOB, Relation)

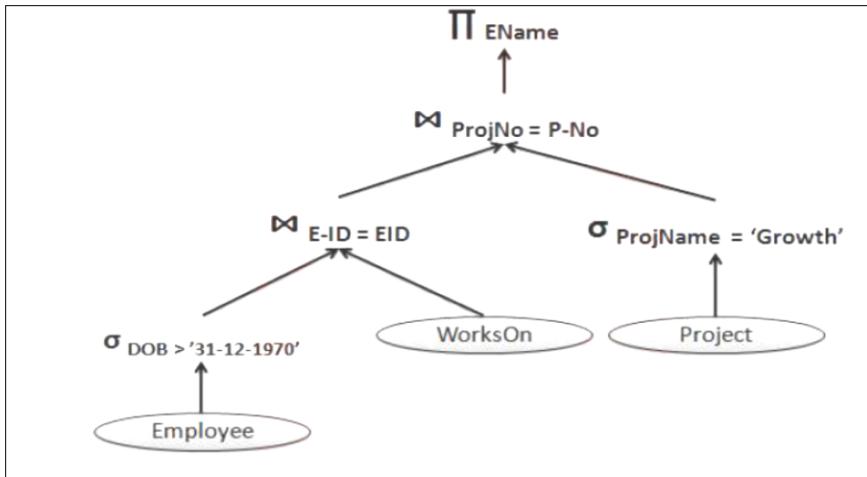
Now let us consider the query in the above database to find the name of employees born after 1970 who work on a project named 'Growth'.

SELECT EName

FROM Employee, WorksOn, Project

WHERE ProjName = 'Growth' AND ProjNo = P-No

AND EID = E-ID AND DOB > '31-12-1970';



**Figure 3.6: Heuristic Rule Example**

### General Transformation Rules:

Transformation rules are used by the query optimizer to transform one relational algebra expression into an equivalent expression that is more efficient to execute.

- A relation is considered as equivalent of another relation if two relations have the same set of attributes in a different order but representing the same information.
- These transformation rules are used to restructure the initial (canonical) relational algebra query tree attributes during query decomposition.

#### 1. Cascade of $\sigma$ :

$$\sigma c_1 \text{ AND } c_2 \text{ AND } \dots \text{ AND } c_n (R) = \sigma c_1 (\sigma c_2 (\dots (\sigma c_n (R)) \dots))$$

#### 2. Commutativity of $\sigma$ :

$$\sigma C_1 (\sigma C_2 (R)) = \sigma C_2 (\sigma C_1 (R))$$

#### 3. Cascade of $\pi$ :

$$\pi List_1 (\pi List_2 (\dots (\pi List_n (R)) \dots)) = \pi List_1 (R)$$

#### 4. Commuting $\sigma$ with $\pi$ :

$$\pi A_1, A_2, A_3, \dots, A_n (\sigma C (R)) = \sigma C (\pi A_1, A_2, A_3, \dots, A_n (R))$$

#### 5. Commutativity of $\bowtie$ AND $\times$ :

$$R \bowtie S = S \bowtie R$$

$$R \times S = S \times R$$

#### 6. Commuting $\sigma$ with $\bowtie$ or $\times$ :

If all attributes in selection condition c involved only attributes of one of the relation schemas (R).

$$\sigma c (R \bowtie S) = (\sigma c (R)) \bowtie S$$

Alternatively, selection condition c can be written as (c1 AND c2) where condition c1 involves only attributes of R and condition c2 involves only attributes of S then:

$$\sigma c (R \bowtie S) = (\sigma c_1 (R)) \bowtie (\sigma c_2 (S))$$

#### 7. Commuting $\Lambda$ with $\bowtie$ or $x$ :

- The projection list L = {A1,A2,..An,B1,B2,...Bm}.
- A1...An attribute of R and B1...Bm attributes of S.
- Join condition C involves only attributes in L then :
- $\Lambda_L (R \bowtie c S) = (\Lambda A_1, \dots, A_n (R)) \bowtie c (\Lambda B_1, \dots, B_m (S))$

#### 8. Commutativity of SET Operation:

$$- R \cup S = S \cup R$$

$$- R \cap S = S \cap R$$

Minus (R-S) is not commutative.

#### 9. Associativity of $\bowtie$ , $x$ , $\cap$ , and $\cup$ :

- If  $\emptyset$  stands for any one of these operation throughout the expression then:

$$(R \emptyset S) \emptyset T = R \emptyset (S \emptyset T)$$

#### 10. Commutativity of $\sigma$ with SET Operation:

- If  $\emptyset$  stands for any one of three operations ( $\cup$ ,  $\cap$ , and  $-$ ) then :

$$\sigma c (R \emptyset S) = (\sigma c (R)) \cup (\sigma c (S))$$

$$\Lambda c (R \emptyset S) = (\Lambda c (R)) \cup (\Lambda c (S))$$

#### 11. The $\Lambda$ operation comute with $\cup$ :

$$\Lambda L (R \cup S) = (\Lambda L(R)) \cup (\Lambda L(S))$$

#### 12. Converting a $(\sigma, x)$ sequence with $\cup$ :

$$(\sigma c (R \times S)) = (R \bowtie c S)$$

#### Heuristic Optimization Algorithm:

The Database Management System use Heuristic Optimization Algorithm that utilizes some of the transformation rules to transform an initial query tree into an optimized and efficient executable query tree.

- The steps of the heuristic optimization algorithm that could be applied during query processing and optimization are:

Step 1:

- Perform SELECT operation to reduce the subsequent processing of the relation:
- Use the transformation rule 1 to break up any SELECT operation with conjunctive condition into a cascade of SELECT operation.

Step 2:

- Perform commutativity of SELECT operation with other operation at the earliest to move each SELECT operation down to query tree.
- Use transformation rules 2, 4, 6 and 10 concerning the commutativity of SELECT with other operation such as unary and binary operations and move each select operation as far down the tree as is permitted by the attributes involved in the SELECT condition. Keep selection predicates on the same relation together.

Step 3:

- Combine the Cartesian product with subsequent SELECT operation whose predicates represents the join condition into a JOIN operation.
- Use the transformation rule 12 to combine the Cartesian product operation with subsequent SELECT operation.

Step 4:

- Use the commutativity and associativity of the binary operations.
- Use transformation rules 5 and 9 concerning commutativity and associativity to rearrange the leaf nodes of the tree so that the leaf node with the most restrictive selection operation are executed first in the query tree representation. The most restrictive SELECT operation means:
  - Either the one that produce a relation with a fewest tuples or with smallest size.
  - The one with the smallest selectivity.

Step 5:

- Perform the projection operation as early as possible to reduce the cardinality of the relation and the subsequent processing of the relation, and move the Projection operations as far down the query tree as possible.
- Use transformation rules 3, 4, 7 and 10 concerning the cascading and commuting of projection operations with other binary operation. Break down and move the projection attributes down the tree as far as needed. Keep the projection attributes in the same relation together.

Step 6:

- Compute common expression once.
- Identify sub-tree that represent group of operations that can be executed by a single algorithm.

### **Cost Estimation in Query Optimization:**

- The main aim of query optimization is to choose the most efficient way of implementing the relational algebra operations at the lowest possible cost.
- Therefore, the query optimizer should not depend solely on heuristic rules, but, it should also estimate the cost of executing the different strategies and find out the strategy with the minimum cost estimate.
- The method of optimizing the query by choosing a strategy those result in minimum cost is called cost-based query optimization.
- The cost-based query optimization uses the formula that estimate the cost for a number of options and selects the one with lowest cost and the most efficient to execute.
- The cost functions used in query optimization are estimates and not exact cost functions.
- The cost of an operation is heavily dependent on its selectivity, that is, the proportion of select operation(s) that forms the output.
- In general, the different algorithms are suitable for low or high selectivity queries. In order for query optimizer to choose suitable algorithm for an operation an estimate of the cost of executing that algorithm must be provided.
- The cost of an algorithm is depending of a cardinality of its input.
- To estimate the cost of different query execution strategies, the query tree is viewed as containing a series of basic operations which are linked in order to perform the query.
- It is also important to know the expected cardinality of an operation's output because this form the input to the next operation.

### **Cost Components of Query Execution:**

- The success of estimating the size and cost of intermediate relational algebra operations depends on the amount the accuracy of statistical data information stored with DBMS.

The cost of executing the query includes the following components:

- Access cost to secondary storage
- Storage cost
- Computation cost

- Memory uses cost
- Communication cost

#### **1. Access cost to secondary storage:**

Access cost is the cost of searching for reading and writing data blocks (containing the number of tuples) that reside on secondary storage, mainly on disk of the DBMS.

The cost of searching for tuples in the database relations depends on the type of access structures on that relation, such ordering, hashing and primary or secondary indexes.

#### **2. Storage cost:**

The storage cost is of storing any intermediate relations that are generated by the executing strategy for the query.

#### **3. Computation cost:**

Computation cost is the cost of performing in-memory operations on the data buffers during query execution.

Such operations contain searching for and sorting records, merging records for a join and performing computation on a field value.

#### **4. Memory uses cost:**

Memory uses cost is a cost pertaining to the number of memory buffers needed during query execution.

#### **5. Communication cost:**

It is the cost of transferring query and its results from the database site to the site of terminal of query organization.

Out of the above five cost components, the most important is the secondary storage access cost.

The emphasis of the cost minimization depends on the **size** and **type** of database applications.

For example, in smaller database the emphasis is on the minimizing computing cost as because most of the data in the files involved in the query can be completely stored in the main memory. For large database, the main emphasis is on minimizing the access cost to secondary device.

Chameli Devi Group of Institutions

Department of Information Technology

Subject Notes

IT405 – Database Management System

B.Tech, IT-4th Semester

Syllabus: SQL: Data definition in SQL, update statements and views in SQL: Data storage and definitions, Data retrieval queries and update statements, Query Processing & Query Optimization: Overview, measures of query cost, selection operation, sorting, join, evaluation of expressions, transformation of relational expressions, estimating statistics of expression results, evaluation plans. Case Study of ORACLE and DB2.

Course Objective: Students can use SQL operations to manipulate the database and learn how to design and create a good database using functional dependencies and normalization.

Course Outcome: Analyze and renovate an information model into a relational database schema and to use a DDL, DML and DCL utilities to implement the schema using a DBMS.

---

#### SQL Command

##### DDL: Data Definition Language

All DDL commands are auto-committed. That means it saves all the changes permanently in the database.

Command      Description

Create    to create new table or database

Alter    for alteration

Truncate      delete data from a table

Drop    to drop a table

Rename      to rename a table

Table 3.1: DDL Commands

##### DML: Data Manipulation Language

DML commands are not auto-committed. It means changes are not permanent to database, they can be rolled back.

Command      Description

Insert    to insert a new row

Update to update existing row

Delete    to delete a row

Merge merging two rows or two tables

#### Table 3.2: DML Commands

##### TCL: Transaction Control Language

These commands are used to keep a check on other commands and their effect on the database. These commands can terminate changes made by other commands by rolling back to original state. It can also make changes permanent.

Command	Description
Commit	to permanently save
Rollback	to undo the change
save point	to save temporarily

#### Table 3.3: TCL Commands

##### DCL: Data Control Language

Data control language provides a command to grant and take back authority.

Command	Description
Grant	grant permission of the right
Revoke	take back permission

#### Table 3.4: DCL Commands

##### DQL: Data Query Language

Command	Description
Select	retrieve records from one or more table

#### Table 3.5: DQL Commands

##### Create command

create is a DDL command used to create a table or a database.

##### Creating a Database

To create a database in RDBMS, create command is used.

Syntax: create database database-name;

Example: create database Test;

The above command will create a database named Test.

## Creating a Table

create command is also used to create a table. We can specify names and datatypes of various columns along.

Syntax:

```
create table table-name  
{  
    column-name1 datatype1,  
    column-name2 datatype2,  
    column-name3 datatype3,  
    column-name4 datatype4  
};
```

create table command will tell the database system to create a new table with given table name and column information.

Example: `create table Student (id int, name varchar, age int);`

alter command: Used for alteration of table structures. There are various uses of alter command, such as,

- to add a column to the existing table
- to rename any existing column
- to change the datatype of any column or to modify its size
- to drop a column

Using alter command we can add a column to an existing table.

Syntax: `alter table table-name add (column-name datatype);`

Example: `alter table Student add (address char);`

To add a column with Default Value

alter command can add a new column to an existing table with default values.

Syntax: `alter table table-name add (column-name1 datatype1 default data);`

Example: `alter table Student add (dob date default '1-Jan-99');`

To Modify an existing Column: Used to modify data type of an existing column.

Syntax: alter table table-name modify (column-name datatype);

Example: alter table Student modify (address varchar (30));

To Rename a column: Using alter command user can rename an existing column.

Syntax: alter table table-name rename old-column-name to column-name;

Example: alter table Student rename address to Location;

To Drop a Column: Used to drop columns also.

Syntax: alter table table-name drop(column-name);

Example: alter table Student drop(address);

truncate command

The truncate command removes all records from a table. But this command will not destroy the table's structure. When we apply truncate command on a table its Primary key is initialized.

Syntax: truncate table table-name

Example: truncate table Student;

drop command

drop query completely removes a table from the database. This command will also destroy the table structure.

Syntax: drop table table-name

Example: drop table Student;

rename command

rename command is used to rename a table.

Syntax: rename table old-table-name to new-table-name

Example: rename table Student to Student-record;

## DML Commands

### 1) INSERT command

Insert command is used to insert data into a table.

Syntax: `INSERT into table-name values (data1, data2,)`

Example:

Consider a table Student with following fields.

S_id	S_Name	age
------	--------	-----

`INSERT into Student values(101,'Adam',15);`

The above command will insert a record into Student table.

S_id	S_Name	age
101	Adam	15

### 2) UPDATE command

Update command is used to update a row of a table.

Syntax: `UPDATE table-name set column-name = value where condition;`

Example:

`update Student set age=18 where s_id=102;`

S_id	S_Name	age
101	Adam	15
102	Alex	18
103	chris	14

Example:

`UPDATE Student set s_name='Abhi', age=17 where s_id=103;`

The above command will update two columns of a record.

S_id	S_Name	age
101	Adam	15
102	Alex	18

### 3) Delete command

Delete command is used to delete data from a table. Delete command can also be used with the condition to delete a particular row.

Syntax: `DELETE from table-name;`

Example: `DELETE from Student;`

The above command will delete all the records from Student table.

### TCL command

Transaction Control Language (TCL) commands are used to manage transactions in the database. These are used to manage the changes made by DML statements. It also allows statements to be grouped together into logical transactions.

#### Commit command

Commit command is used to permanently save any transaction into the database.

Syntax: `commit;`

#### Rollback command

This command restores the database to last committed state. It is also used with savepoint command to jump to a save point in a transaction.

Syntax: `rollback to save point-name;`

#### Savepoint command

savepoint command is used to temporarily save a transaction so that you can rollback to that point whenever necessary.

Syntax: `savepoint savepoint-name;`

### DCL command

System: creating a session, table etc. are all types of system privilege.

Object: any command or query to work on tables comes under object privilege.

DCL defines two commands,

Grant: Gives user access privileges to the database.

Revoke: Take back permissions from the user.

Example: grant create session to username;

S_id	s_Name	Age	Address
101	Adam	15	Noida
102	Alex	18	Delhi
103	Abhi	17	Rohtak
104	Ankit	22	Panipat

Table 3.6: DCL Command Example

#### WHERE clause

Where clause is used to specify condition while retrieving data from the table. Where clause is used mostly with Select, Update and Delete query. If the condition specified by where clause is true then only the result from the table is returned.

Syntax:

```
SELECT column-name1, column-name2, column-name3, column-name N  
from table-name WHERE [condition];
```

Example: SELECT s\_id, s\_name, age, address from Student WHERE s\_id=101;

#### SELECT Query

The Select query is used to retrieve data from a table. It is the most used SQL query. We can retrieve complete tables, or partial by mentioning conditions using WHERE clause.

Syntax:

```
SELECT column-name1, column-name2, column-name3, column-nameN from table-name;  
Example: SELECT s_id, s_name, age from Student.
```

#### Like clause

Like clause is used as a condition in SQL query. Like clause compares data with an expression using wildcard operators. It is used to find similar data from the table.

#### Wildcard operators

There are two wildcard operators that are used in like clause.

Percent sign % represents zero, one or more than one character.

Underscore sign \_ represents only one character.

Example: `SELECT * from Student where s_name like 'A%';`

#### Order by Clause

Order by clause is used with a Select statement for arranging retrieved data in sorted order. The Order by clause by default sort data in ascending order. To sort data in descending order DESC keyword is used with Order by clause.

Syntax: `SELECT column-list|* from table-name order by asc|desc;`

Example: `SELECT * from Emp order by salary;`

#### Group by Clause

Group by clause is used to group the results of a SELECT query based on one or more columns. It is also used with SQL functions to group the result from one or more tables.

Syntax:

`SELECT column_name, function (column_name)`

`FROM table_name`

`WHERE condition`

`GROUP BY column_name`

`Example select name, salary`

`from Emp`

`where age > 25`

`group by salary`

## HAVING Clause

Having clause is used with SQL Queries to give a more precise condition for a statement. It is used to mention condition in Group based SQL functions, just like WHERE clause.

Syntax:

```
select column_name, function(column_name)  
FROM table_name  
WHERE column_name condition  
GROUP BY column_name  
HAVING function (column_name) condition
```

Example SELECT \*

```
from sale group customer  
having sum(previous_balance) > 3000
```

## Distinct keyword

The distinct keyword is used with a Select statement to retrieve unique values from the table. Distinct removes all the duplicate records while retrieving from the database.

Syntax: SELECT distinct column-name from table-name;

Example: select distinct salary from Emp;

## AND & OR operator

AND & OR operators are used with Where clause to make more precise conditions for fetching data from database by combining more than one condition together.

Example:

```
SELECT * from Emp WHERE salary < 10000 AND age > 25
```

```
SELECT * from Emp WHERE salary > 10000 OR age > 25
```

## SQL Constraints

SQL Constraints are rules used to limit the type of data that can go into a table, to maintain the accuracy and integrity of the data inside the table.

Constraints can be divided into following two types,

Column level constraints: limits only column data

Table-level constraints: limits whole table data

Constraints are used to make sure that the integrity of data is maintained in the database.

Following are the most used constraints that can be applied to a table:

#### NOT NULL Constraint

NOT NULL constraint restricts a column from having a NULL value. Once NOT NULL constraint is applied to a column, you cannot pass a null value to that column.

Example: CREATE table Student (s\_id int NOT NULL, Name varchar (60), Age int);

#### UNIQUE Constraint

UNIQUE constraint ensures that a field or column will only have unique values. A UNIQUE constraint field will not have duplicate data.

Example: CREATE table Student (s\_id int NOT NULL UNIQUE, Name varchar (60), Age int);

#### Primary Key Constraint

Primary key constraint uniquely identifies each record in a database. A Primary Key must contain unique value and it must not contain a null value.

Example: CREATE table Student (s\_id int PRIMARY KEY, Name varchar (60) NOT NULL, Age int);

#### Foreign Key Constraint

FOREIGN KEY is used to relate two tables. The foreign KEY constraint is also used to restrict actions that would destroy links between tables.

Example: CREATE table Order\_Detail (order\_id int PRIMARY KEY, order\_name varchar (60) NOT NULL, c\_id int FOREIGN KEY REFERENCES Customer\_Detail(c\_id));

#### On Delete Cascade:

This will remove the record from the child table if that value of the foreign key is deleted from the main table.

**On Delete Null:** This will set all the values in that record of child table as NULL, for which the value of the foreign key is deleted from the main table.

### CHECK Constraint

A check constraint is used to restrict the value of a column between a range. It performs check on the values, before storing them into the database. It's like condition checking before saving data into a column.

Example: `create table Student (s_id int NOT NULL CHECK (s_id > 0), Name varchar (60) NOT NULL,Age int);`

### SQL Functions

SQL provides many built-in functions to perform operations on data. These functions are useful while performing mathematical calculations, string concatenations, sub-strings etc. SQL functions are divided into two categories:

**Aggregate Functions:** These functions return a single value after calculating from a group of values.

Aggregate functions includes: MAX(), MIN(), SUM(), AVG () and COUNT ()

**Scalar Functions:** Scalar functions return a single value from an input value.

**UCASE ():** Used to convert the value of string column to the uppercase character.

### Join in SQL

SQL Join is used to fetch data from two or more tables, which is joined to appear as a single set of data. SQL Join is used for combining column from two or more tables by using values common to both tables. Join Keyword is used in SQL queries for joining two or more tables. Minimum required condition for joining table is  $(n-1)$  where n, is a number of tables. A table can also join to itself known as, Self-Join.

Consider the following two tables:

Table 1: CUSTOMERS Table

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00

2   Khilan   25   Delhi   1500.00
3   kaushik   23   Kota   2000.00
4   Chaitali   25   Mumbai   6500.00
5   Hardik   27   Bhopal   8500.00
6   Komal   22   MP   4500.00
7   Muffy   24   Indore   10000.00

Table 2: ORDERS Table

OID	DATE	CUSTOMER_ID	AMOUNT
102   2009-10-08 00:00:00   3   3000			
100   2009-10-08 00:00:00   3   1500			
101   2009-11-20 00:00:00   2   1560			
103   2008-05-20 00:00:00   4   2060			

Now, let us join these two tables in our SELECT statement as shown below

```
SQL> SELECT ID, NAME, AGE, AMOUNT FROM CUSTOMERS, ORDERS WHERE CUSTOMERS.ID =
ORDERS.CUSTOMER_ID;
```

This query will produce below result

ID	NAME	AGE	AMOUNT
3   kaushik   23   3000			
3   kaushik   23   1500			
2   Khilan   25   1560			
4   Chaitali   25   2060			

Here, it is noticeable that the join is performed in the WHERE clause. Several operators can be used to join tables, such as =, <, >, <>, <=, >=, !=, BETWEEN, LIKE, and NOT; they can all be used to join tables. However, the most common operator is the equal to symbol.

Below are the different types of Join available in SQL:

- INNER JOIN returns rows when there is a match in both tables.
- LEFT JOIN returns all rows from the left table, even if there are no matches in the right table.
- RIGHT JOIN returns all rows from the right table, even if there are no matches in the left table.
- FULL JOIN returns rows when there is a match in one of the tables.
- SELF JOIN is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
- CARTESIAN JOIN returns the Cartesian product of the sets of records from the two or more joined tables.

Figure 3.1: Join Concept

Example:

Orders

OrderID CustomerID EmployeeID OrderDate ShipperID

Customers

CustomerID CustomerName ContactName Address City PostalCode Country

Table 3.7: Join Command Example

Inner Join:

1. SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate

FROM Orders

INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;

2. SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName

FROM Orders

INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID)

```
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);
```

Left Join:

```
SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID  
ORDER BY Customers.CustomerName;
```

Right Join:

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName  
FROM Orders  
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID  
ORDER BY Orders.OrderID;
```

Full Outer Join:

```
SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID  
ORDER BY Customers.CustomerName;
```

Self-Join:

```
SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City  
FROM Customers A, Customers B  
WHERE A.CustomerID <> B.CustomerID  
AND A.City = B.City  
ORDER BY A.City;
```

Query Processing and Optimization

## Query Processing

Query Processing is a procedure of transforming a high-level query (such as SQL) into a correct and efficient execution plan expressed in low-level language. A query processing selects a most appropriate plan that is used in responding to a database request. When a database system receives a query for update or retrieval of information, it goes through a series of compilation steps, called execution plan.

Different phases of Query Processing are:

- In the first phase called syntax checking phase, the system parses the query and checks that it follows the syntax rules or not.
- It then matches the objects in the query syntax with the view tables and columns listed in the system table.
- Finally, it performs the appropriate query modification. During this phase, the system validates the user privileges and that the query does not disobey any integrity rules.
- The execution plan is finally executing to generate a response.

## Query Analyzer

- The syntax analyzer takes the query from the users, parses it into tokens and analyses the tokens and their order to make sure they follow the rules of the language grammar.
- If an error is found in the query submitted by the user, it is rejected and an error code together with an explanation of why the query was rejected is return to the user.

A simple form of the language grammar that could use to implement SQL statement is given below:

- QUERY = SELECT + FROM + WHERE
- SELECT = 'SELECT' + <COLUMN LIST>
- FROM = 'FROM' + <TABLE LIST>
- WHERE = 'WHERE' + VALUE1 OP VALUE2
- VALUE1 = VALUE / COLUMN NAME
- VALUE2 = VALUE / COLUMN NAME
- OP = >, <, >=, <=, =, <>

## Query Decomposition

The query decomposition is the first phase of the query processing whose aims are to transfer the high-level query into a relational algebra query and to check whether that query is syntactically and semantically correct.

- Thus, the query decomposition is start with a high-level query and transform into query graph of low-level operations, which satisfy the query.
- The SQL query is decomposed into query blocks (low-level operations), which form the basic unit.
- Hence nested queries within a query are identified as separate query blocks.
- The query decomposer goes through five stages of processing for decomposition into low-level operation and translation into algebraic expressions.

Figure 3.2: Phases of Query Processing

### 1) Query Analysis:

- During the query analysis phase, the query is syntactically analyzed using the programming language compiler (parser).
- A syntactically legal query is then validated, using the system catalog, to ensure that all data objects (relations and attributes) referred to by the query are defined in the database.
- The type specification of the query qualifiers and result is also checked at this stage.

Let us consider the following query :

```
SELECT emp_nm FROM EMPLOYEE WHERE emp_desg>100
```

This query will be rejected because the comparison “>100” is incompatible with the data type of emp\_desg which is a variable character string.

### Query Tree Notations:

At the end of query analysis phase, the high-level query (SQL) is transformed into some internal representation that is more suitable for processing. This internal representation is typically a kind of query tree.

A Query Tree is a tree data structure that corresponds expression.

A Query Tree is also called a relational algebra tree.

- Leaf node of the tree, representing the base input relations of the query.
- Internal nodes of the tree representing an intermediate relation which is the result of applying an operation in the algebra.

- Root of the tree representing a result of the query.
- Sequence of operations is directed from leaf to root.

Query tree is executed by executing a

- The internal is then replaced by the resulting relation.
- The execution is terminated when the root relation for the query.

Example:

```
SELECT (P.proj_no, P.dept_no, E.name, E.add, E.dob)
FROM PROJECT P, DEPARTMENT D, EMPLOYEE
WHERE P.dept_no = D.d_no AND D.mgr_id = E.emp_id
AND P.proj_loc = 'Mumbai' ;
```

```
Mumbai-PROJ ⊲ σ proj_loc = 'Mumbai' (PROJECT)
CONT-DEPT ⊲ (Mumbai-PROJ ⋈ dept_no = d_no DEPARTMENT)
PROJ-DEPT-MGR ⊲ (CONT-DEPT ⋈ mgr_id=emp_id EMPLOYEE)
RESULT ⊲ Π proj_no, dept_no, name, add, dob (PROJ-DEPT-MGR)
```

The three relations PROJECT, DEPARTMENT, EMPLOYEE is representing as a leaf nodes P, D and E, while the relational algebra operations of the represented by internal tree nodes.

- Same SQL query can have many different relational algebra expressions and hence many different query trees.
- The query parser typically generates a standard initial (canonical) query tree.

Figure 3.3: Query Tree Notation

Query Graph Notations:

- Query graph is sometimes also used for representation of a query. In query graph representation, the relations in the query are represented relation nodes are displayed as a SINGLE CIRCLE.

- The constant values from the query selection (proj\_loc = 'Mumbai') are represented by constant node, displayed as a DOUBLE CIRCLE.
- The selection and join conditions are represented as a Graph Edge (e.g. P.dept\_no = p.dept\_num).
- Finally, the attributes retrieve from the relation are displays in square brackets (P.proj\_num, P.dept\_no).

Figure 3.4: Query Graph

## 2) Query Normalization:

- The primary phase of the normalization is to avoid redundancy. The normalization phase converts the query into a normalized form that can be more easily manipulated.
- In the normalization phase, a set of equivalency rules are applied so that the projection and selection operations included on the query are simplified to avoid redundancy.
- The projection operation corresponds to the SELECT clause of SQL query and the selection operation correspond to the predicate found in WHERE clause.
- The equivalency transformation rules that are applied to SQL query is shown in following table, in which UNARYOP means UNARY operation, BINOP means BINARY operation and REL1, REL2, REL3 are the relations.

Rule Name	Rule Description
1. Commutative of UNARY operation	UNARYOP1 UNARYOP2 REL $\leftrightarrow$ UNARYOP2 UNARYOP1 REL
2. Commutative of BINARY operation	REL1 BINOP (REL2 BINOP REL3) $\leftrightarrow$ (REL1 BINOP REL2) BINOP REL3
3. Idem potency of UNARY operations	UNARYOP1 UNARYOP2 REL UNARYOP REL

4. Distributivity of UNARY operations       $\text{UNARYOP1 (REL1 BINOP REL2)} \leftrightarrow \text{UNARYOP (REL1) BINOP UNARYOP (REL2)}$

5. Factorization of UNARY operations       $\text{UNARIOP (REL1) BINOP UNARYOP (REL2)} \leftrightarrow \text{UNARYOP (REL1 BINOP REL2)}$

Table 3.8: Query Normalization

3) Semantic Analyzer:

- The objective of this phase of query processing is to reduce the number of predicates.
- The semantic analyzer rejects the normalized queries that are incorrectly formulated.
- A query is incorrectly formulated if components do not contribute to the generation of the result. This happens in the case of missing join specification.
- A query is contradictory if its predicate cannot satisfy by any tuple in the relation. The semantic analyzer examines the relational calculus query (SQL) to make sure it contains only data objects that are table, columns, views, indexes that are defined in the database catalog.
- It makes sure that each object in the query is referenced correctly according to its data type.
- In the case of missing join specifications, the components do not contribute to the generation of the results, and thus, a query may be incorrect formulated.
- A query is contradictory if its predicates cannot be satisfied by any of the tuples.

For example:

( emp\_des = 'Programmer'  $\wedge$  emp\_des = 'Analyst' )

As an employee cannot be both 'Programmer' and 'Analyst' simultaneously, the above predicate on the EMPLOYEE relation is contradictory.

Example of Correctness and Contradictory:

Incorrect Formulated: - (Missing Join Specification)

SELECT p.projno, p.proj\_location

FROM project p, viewing v, department d

WHERE d.dept\_id = v.dept\_id AND d.max\_budj  $\geq$  8000

AND d.mgr = 'Methew'

Contradictory: - (Predicate cannot satisfy)

```
SELECT p.proj_no, p.proj_location  
FROM project p, cost_proj c, department d  
WHERE d.max_budj >80000 AND d.max_budj < 50000 AND  
d.dept_id = v.dept_id AND v.proj_no = p.proj_no
```

#### 4. Query Simplifier:

The objectives of this phase are to detect redundant qualification, eliminate common sub-expressions and transform sub-graph to semantically equivalent but easier and efficiently computed form.

Commonly integrity constraints view definitions, and access restrictions are introduced into the graph at this stage of analysis so that the query must be simplified as much as possible.

Integrity constraints define constants which must hold for all state of the database, so any query that contradicts integrity constraints must be avoided and can be rejected without accessing the database.

The final form of simplification is obtaining by applying idempotency rules of Boolean algebra.

	Description	Rule Format
1.	PRED AND PRED	$PRED \wedge P = P$
2.	PRED AND TRUE	$PRED \wedge \text{TRUE} = PRED = P$
3.	PRED AND FALSE	$PRED \wedge \text{FALSE} = \text{FALSE}$
4.	PRED AND NOT(PRED)	$PRED \wedge \neg PRED = \text{FALSE}$
5.	PRED1 AND (PRED1 OR PRED2)	$PRED1 \wedge (PRED1 \vee PRED2) = PRED1 = P1$
6.	PRED OR PRED	$PRED \vee PRED = P$
7.	PRED OR TRUE	$PRED \vee \text{TRUE} = \text{TRUE}$
8.	PRED OR FALSE	$PRED \vee \text{FALSE} = PRED = P$
9.	PRED OR NOT(PRED)	$PRED \vee \neg PRED = \text{TRUE}$
10.	PRED1 OR (PRED1 AND PRED2)	$PRED1 \vee (PRED1 \wedge PRED2) = PRED1 = P1$

Table 3.9: Boolean Algebra Rules

#### 5) Query Restructuring:

- In the final stage of the query decomposition, the query can be restructured to give a more efficient implementation.
- Transformation rules are used to convert one relational algebra expression into an equivalent form that is more efficient.
- The query can now be regarded as a relational algebra program, consisting of a series of operations on relation.

## Query Optimization

Figure 3.5: Query Optimization Phases

The primary goal of query optimization is of choosing an efficient execution strategy for processing a query.

- The query optimizer attempts to minimize the use of certain resources (mainly the number of I/O and CPU time) by selecting a best execution plan (access plan).
- A query optimization starts during the validation phase by the system to validate the user has appropriate privileges.
- Now an action plan is generated to perform the query.

Relational algebra query tree generated by the query simplifier module of query decomposer.

- Estimation formulas used to determine the cardinality of the intermediate result table.
- A cost Model
- Statistical data from the database catalogue.

The output of the query optimizer is the execution plan in form of optimized relational algebra query.

- A query typically has many possible execution strategies, and the process of choosing a suitable one for processing a query is known as Query Optimization.
- The basic issues in Query Optimization are:
  - How to use available indexes.
  - How to use memory to accumulate information and perform immediate steps such as sorting.
  - How to determine the order in which joins should be performed.
- The term query optimization does not mean giving always an optimal (best) strategy as the execution plan.

- It is just a responsibly efficient strategy for execution of the query.
- The decomposed query block of SQL is translating into an equivalent extended relational algebra expression and then optimized.

There are two main techniques for implementing Query Optimization:

The first technique is based on Heuristic Rules for ordering the operations in a query execution strategy. The second technique involves the systematic estimation of the cost of the different execution strategies and choosing the execution plan with the lowest cost.

- Semantic query optimization is used with the combination with the heuristic query transformation rules.
- It uses constraints specified on the database schema such as unique attributes and other more complex constraints, in order to modify one query into another query that is more efficient to execute.

#### 1. Heuristic Rules:

- The heuristic rules are used as an optimization technique to modify the internal representation of the query. Usually, heuristic rules are used in the form of query tree or query graph data structure, to improve its performance.
- One of the main heuristic rules is to apply SELECT operation before applying the JOIN or other BINARY operations.
- This is because the size of the file resulting from a binary operation such as JOIN is usually a multi-value function of the sizes of the input files.
- The SELECT and PROJECT reduced the size of the file and hence, should be applied before the JOIN or other binary operation.
- Heuristic query optimizer transforms the initial (canonical) query tree into final query tree using equivalence transformation rules. This final query tree is efficient to execute.

For example, consider the following relations:

Employee (EName, EID, DOB, EAdd, Sex, ESalary, EDeptNo)

Department (DeptNo, DeptName, DeptMgrID, Mgr\_S\_date)

DeptLoc (DeptNo, Dept\_Loc)

Project (ProjName, ProjNo, ProjLoc, ProjDeptNo)

WorksOn (E-ID, P-No, Hours)

Dependent (E-ID, DependName, Sex, DDOB, Relation)

Now let us consider the query in the above database to find the name of employees born after 1970 who work on a project named 'Growth'.

```
SELECT EName  
FROM Employee, WorksOn, Project  
WHERE ProjName = 'Growth' AND ProjNo = P-No  
AND EID = E-ID AND DOB > '31-12-1970';
```

Figure 3.6: Heuristic Rule Example

#### General Transformation Rules:

Transformation rules are used by the query optimizer to transform one relational algebra expression into an equivalent expression that is more efficient to execute.

- A relation is considered as equivalent of another relation if two relations have the same set of attributes in a different order but representing the same information.
- These transformation rules are used to restructure the initial (canonical) relational algebra query tree attributes during query decomposition.

##### 1. Cascade of $\sigma$ :

$$\sigma c_1 \text{ AND } c_2 \text{ AND } \dots \text{ AND } c_n (R) = \sigma c_1 (\sigma c_2 (\dots (\sigma c_n (R)) \dots))$$

##### 2. Commutativity of $\sigma$ :

$$\sigma C_1 (\sigma C_2 (R)) = \sigma C_2 (\sigma C_1 (R))$$

##### 3. Cascade of $\pi$ :

$$\pi List_1 (\pi List_2 (\dots (\pi List_n (R)) \dots)) = \pi List_1 (R)$$

##### 4. Commuting $\sigma$ with $\pi$ :

$$\pi A_1, A_2, A_3, \dots, A_n (\sigma C (R)) = \sigma C (\pi A_1, A_2, A_3, \dots, A_n (R))$$

##### 5. Commutativity of $\bowtie$ AND $x$ :

$$R \bowtie S = S \bowtie R$$

$$R \times S = S \times R$$

##### 6. Commuting $\sigma$ with $\bowtie$ or $x$ :

If all attributes in selection condition  $c$  involved only attributes of one of the relation schemas ( $R$ ).

$$\sigma c (R \bowtie S) = (\sigma c (R)) \bowtie S$$

Alternatively, selection condition c can be written as  $(c1 \text{ AND } c2)$  where condition c1 involves only attributes of R and condition c2 involves only attributes of S then:

$$\sigma c (R \bowtie S) = (\sigma c1 (R)) \bowtie (\sigma c2 (S))$$

7. Commuting  $\Lambda$  with  $\bowtie$  or  $x$ :

- The projection list L = {A1,A2,..An,B1,B2,...Bm}.
- A1...An attribute of R and B1...Bm attributes of S.
- Join condition C involves only attributes in L then :
- $\Lambda_L (R \bowtie c S) = (\Lambda_{A1,..An}(R)) \bowtie c (\Lambda_{B1,..Bm}(S))$

8. Commutativity of SET Operation:

$$- R \cup S = S \cup R$$

$$- R \cap S = S \cap R$$

Minus ( $R-S$ ) is not commutative.

9. Associativity of  $\bowtie$ ,  $x$ ,  $\cap$ , and  $\cup$ :

- If  $\emptyset$  stands for any one of these operation throughout the expression then:

$$(R \emptyset S) \emptyset T = R \emptyset (S \emptyset T)$$

10. Commutativity of  $\sigma$  with SET Operation:

- If  $\emptyset$  stands for any one of three operations ( $\cup, \cap$ , and -) then :

$$\sigma c (R \emptyset S) = (\sigma c (R)) \emptyset (\sigma c (S))$$

$$\Lambda c (R \emptyset S) = (\Lambda c (R)) \emptyset (\Lambda c (S))$$

11. The  $\Lambda$  operation comute with  $\cup$ :

$$\Lambda L (R \cup S) = (\Lambda L(R)) \cup (\Lambda L(S))$$

12. Converting a  $(\sigma, x)$  sequence with  $\cup$ :

$$(\sigma c (R x S)) = (R \bowtie c S)$$

Heuristic Optimization Algorithm:

The Database Management System use Heuristic Optimization Algorithm that utilizes some of the transformation rules to transform an initial query tree into an optimized and efficient executable query tree.

- The steps of the heuristic optimization algorithm that could be applied during query processing and optimization are:

**Step 1:**

- Perform SELECT operation to reduce the subsequent processing of the relation:
- Use the transformation rule 1 to break up any SELECT operation with conjunctive condition into a cascade of SELECT operation.

**Step 2:**

- Perform commutativity of SELECT operation with other operation at the earliest to move each SELECT operation down to query tree.
- Use transformation rules 2, 4, 6 and 10 concerning the commutativity of SELECT with other operation such as unary and binary operations and move each select operation as far down the tree as is permitted by the attributes involved in the SELECT condition. Keep selection predicates on the same relation together.

**Step 3:**

- Combine the Cartesian product with subsequent SELECT operation whose predicates represents the join condition into a JOIN operation.
- Use the transformation rule 12 to combine the Cartesian product operation with subsequent SELECT operation.

**Step 4:**

- Use the commutativity and associativity of the binary operations.
- Use transformation rules 5 and 9 concerning commutativity and associativity to rearrange the leaf nodes of the tree so that the leaf node with the most restrictive selection operation are executed first in the query tree representation. The most restrictive SELECT operation means:
  - Either the one that produce a relation with a fewest tuples or with smallest size.
  - The one with the smallest selectivity.

**Step 5:**

- Perform the projection operation as early as possible to reduce the cardinality of the relation and the subsequent processing of the relation, and move the Projection operations as far down the query tree as possible.
- Use transformation rules 3, 4, 7 and 10 concerning the cascading and commuting of projection operations with other binary operation. Break down and move the projection attributes down the tree as far as needed. Keep the projection attributes in the same relation together.

**Step 6:**

- Compute common expression once.
- Identify sub-tree that represent group of operations that can be executed by a single algorithm.

### Cost Estimation in Query Optimization:

- The main aim of query optimization is to choose the most efficient way of implementing the relational algebra operations at the lowest possible cost.
- Therefore, the query optimizer should not depend solely on heuristic rules, but, it should also estimate the cost of executing the different strategies and find out the strategy with the minimum cost estimate.
- The method of optimizing the query by choosing a strategy those result in minimum cost is called cost-based query optimization.
- The cost-based query optimization uses the formula that estimate the cost for a number of options and selects the one with lowest cost and the most efficient to execute.
- The cost functions used in query optimization are estimates and not exact cost functions.
- The cost of an operation is heavily dependent on its selectivity, that is, the proportion of select operation(s) that forms the output.
- In general, the different algorithms are suitable for low or high selectivity queries. In order for query optimizer to choose suitable algorithm for an operation an estimate of the cost of executing that algorithm must be provided.
- The cost of an algorithm is depending of a cardinality of its input.
- To estimate the cost of different query execution strategies, the query tree is viewed as containing a series of basic operations which are linked in order to perform the query.
- It is also important to know the expected cardinality of an operation's output because this form the input to the next operation.

### Cost Components of Query Execution:

- The success of estimating the size and cost of intermediate relational algebra operations depends on the amount the accuracy of statistical data information stored with DBMS.

The cost of executing the query includes the following components:

- Access cost to secondary storage
- Storage cost
- Computation cost
- Memory uses cost
- Communication cost

#### 1. Access cost to secondary storage:

Access cost is the cost of searching for reading and writing data blocks (containing the number of tuples) that reside on secondary storage, mainly on disk of the DBMS.

The cost of searching for tuples in the database relations depends on the type of access structures on that relation, such ordering, hashing and primary or secondary indexes.

2. Storage cost:

The storage cost is of storing any intermediate relations that are generated by the executing strategy for the query.

3. Computation cost:

Computation cost is the cost of performing in-memory operations on the data buffers during query execution.

Such operations contain searching for and sorting records, merging records for a join and performing computation on a field value.

4. Memory uses cost:

Memory uses cost is a cost pertaining to the number of memory buffers needed during query execution.

5. Communication cost:

It is the cost of transferring query and its results from the database site to the site of terminal of query organization.

Out of the above five cost components, the most important is the secondary storage access cost.

The emphasis of the cost minimization depends on the size and type of database applications.

For example, in smaller database the emphasis is on the minimizing computing cost as because most of the data in the files involved in the query can be completely stored in the main memory. For large database, the main emphasis is on minimizing the access cost to secondary device.

**Chameli Devi Group of Institutions**

**Department of Information Technology**

**Subject Notes**

**IT405 – Database Management System**

**B.Tech, IT-4<sup>th</sup> Semester**

**Syllabus:** Relational Model: Structure of relational databases, Domains, Relations, Relational algebra fundamental operators and syntax, relational algebra queries, Entity-Relationship model :Basic concepts, Design process, constraints, Keys, Design issues, E-R diagrams, weak entity sets, extended E-R features – generalization, specialization and aggregation

**Course Objective:** To study of different data model and conceptual design using ER diagram

**Course Outcome (CO 2):** Analyze the physical and logical database designs, database modelling, relational, hierarchical, and network models

## Unit II

**Relational Model:** The relational model represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as tables. However, the physical storage of the data is independent of the way the data are logically organized.

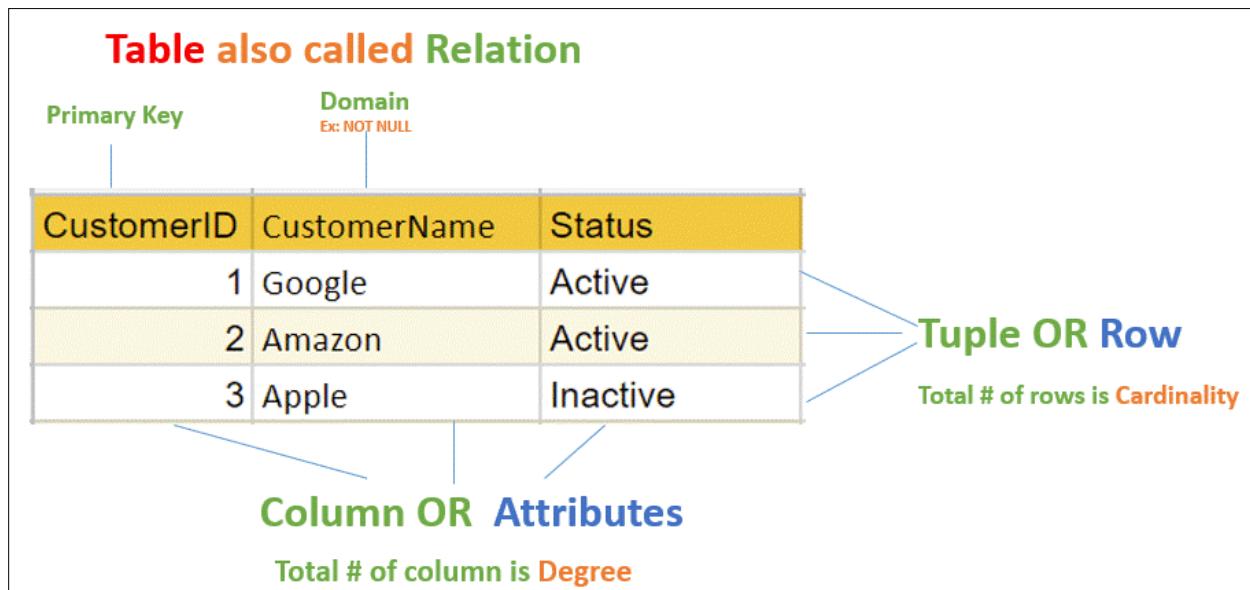


Figure 2.1: Relational Database Structure

- **Relations:** In relational data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where rows represents records and columns represent the attributes.

- Tuple: A single row of a table, which contains a single record for that relation is called a tuple.
- Relation instance: A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.
- Relation schema: A relation schema describes the relation name (table name), attributes, and their names.
- Relation key: Each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.
- Attribute domain: Every attribute has some pre-defined value scope, known as attribute domain.

**Relational Algebra:** Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either unary or binary. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

In the formal relational model terminology, a **row is called a tuple**, a **column header is called an attribute**, and the table is called a **relation**. The data type describing the types of values that can appear in each column is represented by a domain of possible values. We now define these terms—domain, tuple, attribute, and relation formally.

### Domains, Attributes, Tuples, and Relations

**Deposit Relation**

bname	Account	Ename	Balance
Bhanwarkuwan	SBI1200	Ram	5000
Tilak Nagar	SBI1238	Amit	1000

**Customer Relation**

Ename	Street	City
Ramesh	MG road	Indore
Jhon	RNT Marg	Indore

- It has four attributes.
- For each attribute there is a permitted set of values, called the **domain** of that attribute.
- E.g. the domain of *bname* is the set of all branch names.

Let  $D_1$  denote the domain of *bname*, and  $D_2, D_3$  and  $D_4$  the remaining attributes' domains respectively.

A domain D is a set of atomic values. By atomic we mean that each value in the domain is indivisible as far as the formal relational model is concerned. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain, to help in interpreting its values. Some examples of domains follow:

- `Usa_phone_numbers`. The set of ten-digit phone numbers valid in the United States.
- `Local_phone_numbers`. The set of seven-digit phone numbers valid within a particular area code in the United States. The use of local phone numbers is quickly becoming obsolete, being replaced by standard ten-digit numbers.

A relation schema R, denoted by  $R(A_1, A_2, \dots, A_n)$ , is made up of a relation name R and a list of attributes,  $A_1, A_2, \dots, A_n$ . Each attribute  $A_i$  is the name of a role played by some domain D in the relation schema R. D is called the domain of  $A_i$  and is denoted by  $\text{dom}(A_i)$ . A relation schema is used to describe a relation R is called the name of this relation. The degree (or arity) of a relation is the number of attributes n of its relation schema.

Using the datatype of each attribute, the definition is sometimes written as:

`STUDENT (Name: string, Ssn: string, Homophone: string, Address: string, Office phone: string, Age: integer, Gpa: real)`

### **Characteristics of Relations**

**Ordering of Tuples in a Relation.** A relation is defined as a set of tuples. Mathematically, elements of a set have no order among them hence, tuples in a relation do not have any particular order.

**Ordering of Values within a Tuple and an Alternative Definition of a Relation.** According to the preceding definition of a relation, an n-tuple is an ordered list of n values, so the ordering of values in a tuple and hence of attributes in a relation schema is important.

**Values and NULLs in the Tuples.** Each value in a tuple is an atomic value that is, it is not divisible into components within the framework of the basic relational model. Hence, composite and multivalued attributes are not allowed. An important concept is that of NULL values, which are used to represent the values of attributes that may be unknown or may not apply to a tuple.

### **Relational Model Notation**

A relation schema R of degree n is denoted by R (A<sub>1</sub>, A<sub>2</sub>, ..., A<sub>n</sub>).

- The uppercase letters Q, R, S denote relation names.
- The lowercase letters q, r, s denotes relation states.
- The letters t, u, v denotes tuples.
- In general, the name of a relation schema such as STUDENT also indicates the current set of tuples in that relation—the current relation state—whereas STUDENT (Name, Ssn, ...) refers only to the relation schema.
- An attribute A can be qualified with the relation name R to which it belongs by using the dot notation R.A—for example, STUDENT. Name or STUDENT. Age. This is because the same name may be used for two attributes in different relations.

### **Relational Model Constraints and Relational Database Schemas**

Constraints on databases can generally be divided into three main categories:

1. Constraints that are inherent in the data model. We call these inherent model-based constraints or implicit constraints. Example: In the relational model, no two tuples in a relation can be duplicates. Why? Because a relation is a set of tuples, as opposed to a bag/multiset or a sequence.
2. Constraints that can be directly expressed in schemas of the data model, typically by specifying them in the DDL. We call these schema-based constraints or explicit constraints.
3. Constraints that cannot be directly expressed in the schemas of the data model, and hence must be expressed and enforced by the application programs. We call these application-based or semantic constraints or business rules.

### **Types of Schema based Constraints**

#### **Domain Constraints**

Domain constraints specify that within each tuple, the value of each attribute A must be an atomic value from the domain  $\text{dom}(A)$ .

#### **Key Constraints and Constraints on NULL Values**

In the formal relational model, a relation is defined as a set of tuples. By definition, all elements of a set are distinct; hence, all tuples in a relation must also be distinct.

A super key SK specifies a uniqueness constraint that no two distinct tuples in any state  $r$  of  $R$  can have the same value for SK. Every relation has at least one default super key—the set of all its attributes. A super key can have redundant attributes, however, so a more useful concept is that of a key, which has no redundancy.

A relation schema may have more than one key. In this case, each of the keys is called a **candidate key**. For example, the CAR relation has (**Licence\_no** **Eng\_sr\_no** **Model** **Make\_year** model) two candidate keys: **License\_number** and **Engine\_serial\_number**. It is common to designate one of the candidate keys as the **primary key** of the relation. This is the candidate key whose values are used to identify tuples in the relation.

### Relational Databases and Relational Database Schemas

The definitions and constraints we have discussed so far apply to single relations and their attributes. A relational database usually contains many relations, with tuples in relations that are related in various ways. In this section, we define a relational database and a relational database schema.

A relational database schema  $S$  is a set of relation schemas  $S = \{R_1, R_2, \dots, R_m\}$  and a set of integrity constraints  $IC$ . A relational database state  $DB$  of  $S$  is a set of relation states  $DB = \{r_1, r_2, \dots, r_m\}$  such that each  $r_i$  is a state of  $R_i$  and such that the  $r_i$  relation states satisfy the integrity constraints specified in  $IC$ . Below Figure shows a relational database schema that we call **COMPANY** = {EMPLOYEE, DEPARTMENT, DEPT\_LOCATIONS, PROJECT, WORKS\_ON, DEPENDENT}. The underlined attributes represent primary keys.

**EMPLOYEE**

Fname	Minit	Lname	Ssn	Bdata	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	-----	-------	---------	-----	--------	-----------	-----

**DEPARTMENT**

DNAME	Dnumber	Mgr_ssn	Mgr_str_Date
-------	---------	---------	--------------

**DEPT\_ LOCATIONS**

Dnumber	DLocation
---------	-----------

**PROJECT**

Pname	Pnumber	Plocation	Dnum
-------	---------	-----------	------

Figure 2.2: Relational Database

## Integrity, Referential Integrity, and Foreign Keys

The entity integrity constraint states that no primary key value can be NULL. This is because the primary key value is used to identify individual tuples in a relation. Having NULL values for the primary key implies that we cannot identify some tuples. For example, if two or more tuples had NULL for their primary keys, we may not be able to distinguish them if we try to reference them from other relations.

The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.

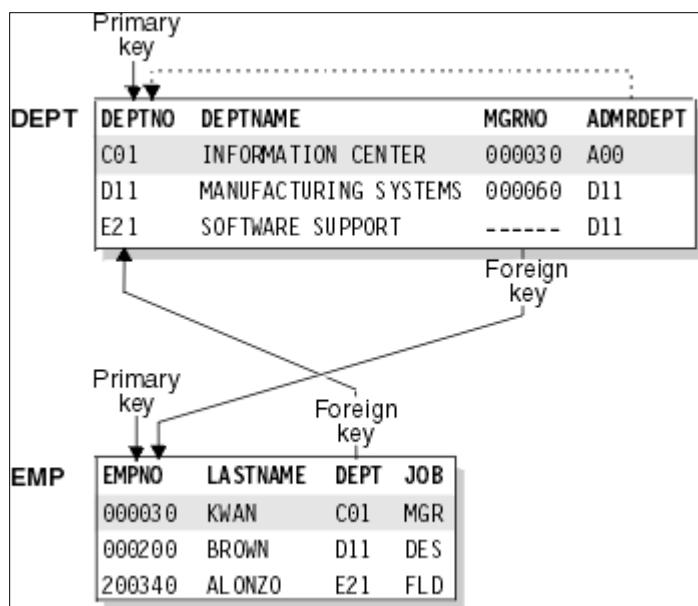


Figure 2.3: Primary & Referential Integrity

The relational algebra is essential for several reasons. First, it provides a formal foundation for relational model operations. Second, and perhaps more important, it is used as a basis for implementing and optimizing queries in the query processing and optimization modules that are integral parts of relational database management systems (RDBMSs).

BASIS FOR COMPARISON	RELATIONAL ALGEBRA	RELATIONAL CALCULUS
Basic	Relational Algebra is a Procedural language.	Relational Calculus is Declarative language.
States	Relational Algebra states how to obtain the result.	Relational Calculus states what result we have to obtain.
Order	Relational Algebra describes the order in which operations have to be performed.	Relational Calculus does not specify the order of operations.
Domain	Relational Algebra is not domain dependent.	Relation Calculus can be domain dependent.
Related	It is close to a programming language.	It is close to the natural language.

Table 2.1: Difference between Calculus and Algebra

The fundamental operations of relational algebra are as follows –

- Select
- Project
- Union
- Set different
- Cartesian product
- Rename

We will discuss all these operations in the following sections.

### Select Operation ( $\sigma$ )

It selects tuples that satisfy the given predicate from a relation.

#### Notation – $\sigma(p)(r)$

Where  $\sigma$  stands for selection predicate and  $r$  stands for relation.  $p$  is propositional logic formula which may use connectors like and, or, and not. These terms may use relational operators like –  $=, \neq, \geq, <, >, \leq$ .

For example –

$\sigma_{\text{subject}} = \text{"database"}(\text{Books})$

Output – Selects tuples from books where subject is 'database'.

$\sigma_{\text{subject}} = \text{"database"} \text{ and } \text{price} = \text{"450"}(\text{Books})$

Output – Selects tuples from books where subject is 'database' and 'price' is 450.

$\sigma_{\text{subject}} = \text{"database"} \text{ and } \text{price} = \text{"450"} \text{ or } \text{year} > \text{"2010"}(\text{Books})$

Output – Selects tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

### **Project Operation ( $\Pi$ )**

It projects column(s) that satisfy a given predicate.

Notation –  $\Pi_{A1, A2, An}(r)$

Where  $A1, A2, An$  are attribute names of relation  $r$ .

Duplicate rows are automatically eliminated, as relation is a set.

For example –

$\Pi_{\text{subject}, \text{author}}(\text{Books})$

Selects and projects columns named as subject and author from the relation Books.

### **Union Operation ( $U$ )**

It performs binary union between two given relations and is defined as –

$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$

Notation –  $r \cup s$

Where  $r$  and  $s$  are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold –

$r$ , and  $s$  must have the same number of attributes.

Attribute domains must be compatible.

Duplicate tuples are automatically eliminated.

$\Pi_{\text{author}}(\text{Books}) \cup \Pi_{\text{author}}(\text{Articles})$

Output – Projects the names of the authors who have either written a book or an article or both.

### **Set Difference ( $-$ )**

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation –  $r - s$

Finds all the tuples that are present in  $r$  but not in  $s$ .

$\Pi_{\text{author}}(\text{Books}) - \Pi_{\text{author}}(\text{Articles})$

Output – Provides the name of authors who have written books but not articles.

### **Cartesian Product (X)**

Combines information of two different relations into one.

Notation –  $r \times s$

Where  $r$  and  $s$  are relations and their output will be defined as –

$$r \times s = \{ q t \mid q \in r \text{ and } t \in s \}$$

$\sigma_{\text{author}} = 'tt'(\text{Books} \times \text{Articles})$

Output – Yields a relation, which shows all the books and articles written by tt.

### **Rename Operation ( $\rho$ )**

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter rho  $\rho$ .

### **Notation – $\rho_x(E)$**

Where the result of expression  $E$  is saved with name of  $x$ .

Additional operations are –

- Set intersection
- Assignment
- Natural join

We can define the three operations UNION, INTERSECTION, and SET DIFFERENCE

on two union-compatible relations  $R$  and  $S$  as follows:

- UNION: The result of this operation, denoted by  $R \cup S$ , is a relation that includes all tuples that are either in  $R$  or in  $S$  or in both  $R$  and  $S$ . Duplicate tuples are eliminated.
- INTERSECTION: The result of this operation, denoted by  $R \cap S$ , is a relation that includes all tuples that are in both  $R$  and  $S$ .
- SET DIFFERENCE (or MINUS): The result of this operation, denoted by  $R - S$ , is a relation that includes all tuples that are in  $R$  but not in  $S$ .

The set operations UNION, INTERSECTION, and MINUS. (a) Two union-compatible relations.

(b) STUDENT  $\cup$  INSTRUCTOR. (c) STUDENT  $\cap$  INSTRUCTOR. (d) STUDENT – INSTRUCTOR.

(e) INSTRUCTOR – STUDENT.

**CARTESIAN PRODUCT** operation—also known as CROSS PRODUCT or CROSS JOIN—which is denoted by  $\times$ . This is also a binary set operation, but the relations on which it is applied do not have to be union compatible. The JOIN operation, denoted by  $\bowtie$ , is used to combine related tuples from two relations into single “longer” tuples. This operation is very important for any relational database with more than a single relation because it allows us to process relationships among relations.

$\text{DEPT\_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{Mgr\_ssn}=\text{Snn}} \text{EMPLOYEE}$

$\text{RESULT} \leftarrow \Pi_{\text{Dname}, \text{Lname}} (\text{DEPT\_MGR})$

(a) STUDENT	INSTRUCTOR	(b)																																																		
<table border="1"> <thead> <tr> <th>Fn</th> <th>Ln</th> </tr> </thead> <tbody> <tr><td>Susan</td><td>Yao</td></tr> <tr><td>Ramesh</td><td>Shah</td></tr> <tr><td>Johnny</td><td>Kohler</td></tr> <tr><td>Barbara</td><td>Jones</td></tr> <tr><td>Amy</td><td>Ford</td></tr> <tr><td>Jimmy</td><td>Wang</td></tr> <tr><td>Ernest</td><td>Gilbert</td></tr> </tbody> </table>	Fn	Ln	Susan	Yao	Ramesh	Shah	Johnny	Kohler	Barbara	Jones	Amy	Ford	Jimmy	Wang	Ernest	Gilbert	<table border="1"> <thead> <tr> <th>Fname</th> <th>Lname</th> </tr> </thead> <tbody> <tr><td>John</td><td>Smith</td></tr> <tr><td>Ricardo</td><td>Browne</td></tr> <tr><td>Susan</td><td>Yao</td></tr> <tr><td>Francis</td><td>Johnson</td></tr> <tr><td>Ramesh</td><td>Shah</td></tr> </tbody> </table>	Fname	Lname	John	Smith	Ricardo	Browne	Susan	Yao	Francis	Johnson	Ramesh	Shah	<table border="1"> <thead> <tr> <th>Fn</th> <th>Ln</th> </tr> </thead> <tbody> <tr><td>Susan</td><td>Yao</td></tr> <tr><td>Ramesh</td><td>Shah</td></tr> <tr><td>Johnny</td><td>Kohler</td></tr> <tr><td>Barbara</td><td>Jones</td></tr> <tr><td>Amy</td><td>Ford</td></tr> <tr><td>Jimmy</td><td>Wang</td></tr> <tr><td>Ernest</td><td>Gilbert</td></tr> <tr><td>John</td><td>Smith</td></tr> <tr><td>Ricardo</td><td>Browne</td></tr> <tr><td>Francis</td><td>Johnson</td></tr> </tbody> </table>	Fn	Ln	Susan	Yao	Ramesh	Shah	Johnny	Kohler	Barbara	Jones	Amy	Ford	Jimmy	Wang	Ernest	Gilbert	John	Smith	Ricardo	Browne	Francis	Johnson
Fn	Ln																																																			
Susan	Yao																																																			
Ramesh	Shah																																																			
Johnny	Kohler																																																			
Barbara	Jones																																																			
Amy	Ford																																																			
Jimmy	Wang																																																			
Ernest	Gilbert																																																			
Fname	Lname																																																			
John	Smith																																																			
Ricardo	Browne																																																			
Susan	Yao																																																			
Francis	Johnson																																																			
Ramesh	Shah																																																			
Fn	Ln																																																			
Susan	Yao																																																			
Ramesh	Shah																																																			
Johnny	Kohler																																																			
Barbara	Jones																																																			
Amy	Ford																																																			
Jimmy	Wang																																																			
Ernest	Gilbert																																																			
John	Smith																																																			
Ricardo	Browne																																																			
Francis	Johnson																																																			
(c)	(d)	(e)																																																		
<table border="1"> <thead> <tr> <th>Fn</th> <th>Ln</th> </tr> </thead> <tbody> <tr><td>Susan</td><td>Yao</td></tr> <tr><td>Ramesh</td><td>Shah</td></tr> </tbody> </table>	Fn	Ln	Susan	Yao	Ramesh	Shah	<table border="1"> <thead> <tr> <th>Fn</th> <th>Ln</th> </tr> </thead> <tbody> <tr><td>Johnny</td><td>Kohler</td></tr> <tr><td>Barbara</td><td>Jones</td></tr> <tr><td>Amy</td><td>Ford</td></tr> <tr><td>Jimmy</td><td>Wang</td></tr> <tr><td>Ernest</td><td>Gilbert</td></tr> </tbody> </table>	Fn	Ln	Johnny	Kohler	Barbara	Jones	Amy	Ford	Jimmy	Wang	Ernest	Gilbert	<table border="1"> <thead> <tr> <th>Fname</th> <th>Lname</th> </tr> </thead> <tbody> <tr><td>John</td><td>Smith</td></tr> <tr><td>Ricardo</td><td>Browne</td></tr> <tr><td>Francis</td><td>Johnson</td></tr> </tbody> </table>	Fname	Lname	John	Smith	Ricardo	Browne	Francis	Johnson																								
Fn	Ln																																																			
Susan	Yao																																																			
Ramesh	Shah																																																			
Fn	Ln																																																			
Johnny	Kohler																																																			
Barbara	Jones																																																			
Amy	Ford																																																			
Jimmy	Wang																																																			
Ernest	Gilbert																																																			
Fname	Lname																																																			
John	Smith																																																			
Ricardo	Browne																																																			
Francis	Johnson																																																			

Figure 2.4: Cartesian product

#### Variations of JOIN: The EQUIJOIN and NATURAL JOIN

The most common use of JOIN involves join conditions with equality comparisons only. Such a JOIN, where the only comparison operator used is  $=$ , is called an EQUIJOIN.

The standard definition of NATURAL JOIN requires that the two join attributes (or each pair of join attributes) have the same name in both relations. If this is not the case, a renaming operation is applied first.

$\text{PROJ\_DEP DEPT\_LOCS} \leftarrow \text{DEPARTMENT} * \text{DEPT\_LOCATIONS}.$

<b>DEPT_LOCS</b>				
Dname	Dnumber	Mgr_ssn	Mgr_start_date	Location
Headquarters	1	888665555	1981-06-19	Houston
Administration	4	987654321	1995-01-01	Stafford
Research	5	333445555	1988-05-22	Bellaire
Research	5	333445555	1988-05-22	Sugarland
Research	5	333445555	1988-05-22	Houston

Figure 2.5: Join Example

### The DIVISION Operation

The DIVISION operation, denoted by  $\div$ , is useful for a special kind of query that sometimes occurs in database applications. An example is Retrieve the names of employees who work on all the projects that ‘John Smith’ works on.

$\text{SMITH} \leftarrow \sigma_{\text{Fname} = 'Jhon' \text{ AND } \text{Lname} = 'SMITH'} (\text{EMPLOYEE})$

$\text{SMITH\_PNOS} \leftarrow \pi_{\text{pno}} (\text{WORKS\_ON} \bowtie_{\text{Essn} = \text{Ssn}} \text{SMITH})$

	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{<\text{selection condition}>}(\text{R})$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{<\text{attribute list}>}(\text{R})$

THETA JOIN	Produces all combinations of tuples from R and R1 2 that satisfy the join condition.	R1 <join condition>
EQUIJOIN	Produces all the combinations of tuples from R1 and R2 that satisfy a join condition with only equality comparisons.	R1 <join condition>
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	R1*<join condition> R2, OR R1* (<join attributes 1>), $\cup_{\text{join}}$
UNION	Produces a relation that includes all the tuples in R1 or R2 or both R1 and R2; R1 and R2 must be union-compatible.	R1 $\cup$ R2
INTERSECTION	Produces a relation that includes all the tuples in both R1 and R2; R1 and R2 must be union-compatible.	R1 $\cap$ R2
DIFFERENCE	Produces a relation that includes all the tuples in R1 that are not in R2; R1 and R2 must be union-compatible.	R1 – R2
CARTESIAN PRODUCT	Produces a relation that has the attributes of R1 and R2 and includes as tuples all possible combinations of tuples from R1 and R2.	R1 $\times$ R2
DIVISION	Produces a relation R(X) that includes all tuples t[X] in R1(Z) that appear in R1 in combination with every tuple from R(Y), where Z = X $\cup$ Y.	R1(Z) $\div$ R2(Y)

Table 2.2: Various Operations of Relational Algebra

Example on Relational Algebra

EMPLOYEE										
Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno	
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5	
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5	
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4	
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4	
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5	
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5	
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4	
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1	

DEPARTMENT				DEPT_LOCATIONS			
Dname	Dnumber	Mgr_ssn	Mgr_start_date	Dnumber	Dlocation		
Research	5	333445555	1988-05-22	1	Houston		
Administration	4	987654321	1995-01-01	4	Stafford		
Headquarters	1	888665555	1981-06-19	5	Bellaire		
				5	Sugarland		
				5	Houston		

WORKS_ON				PROJECT			
Essn	Pno	Hours		Pname	Pnumber	Plocation	Dnum
123456789	1	32.5		ProductX	1	Bellaire	5
123456789	2	7.5		ProductY	2	Sugarland	5
666884444	3	40.0		ProductZ	3	Houston	5
453453453	1	20.0		Computerization	10	Stafford	4
453453453	2	20.0		Reorganization	20	Houston	1
333445555	2	10.0		Newbenefits	30	Stafford	4
333445555	3	10.0					
333445555	10	10.0					
333445555	20	10.0					
999887777	30	30.0					
999887777	10	10.0					
987987987	10	35.0					
987987987	30	5.0					
987654321	30	20.0					
987654321	20	15.0					
888665555	20	NULL					

DEPENDENT				
Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Figure 2.6: Relational Algebra

Query 1. Retrieve the name and address of all employees who work for the 'Research' department.

RESEARCH\_DEPT  $\leftarrow \sigma_{Dname = 'Research'}(DEPARTMENT)$

RESEARCH\_EMPS  $\leftarrow (RESEARCH\_DEPT \bowtie_{Dnumber = Dno} EMPLOYEE)$

RESULT  $\leftarrow \pi_{Fname, Lname, Address}(RESEARCH_EMPS)$

Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

```

STAFFORD_PROJs ← σ Plocation = 'stafford'(PROJECT)
CONTR_DEPTS ← (STAFFORD_PROJs ⋈ Dnum=Dnumber DEPARTMENT)
PROJ_DEPT_MGRS ← (CONTR_DEPTS ⋈ Mgr_ssn = Ssn EMPLOYEE)
RESULT ← π Pnumber, Dnum, Lname, Address, Bdate(PROJ_DEPT_MGRS)

```

Query 3. Find the names of employees who work on all the projects controlled by department number 5.

```

DEPT5_PROJS ← ρ(Pno)(πPnumber(σDnum=5(PROJECT)))
EMP_PROJ ← ρ(Ssn, Pno)(πEssn, Pno(WORKS_ON))
RESULT_EMP_SSNS ← EMP_PROJ ÷ DEPT5_PROJS
RESULT ← πLname, Fname(RESULT_EMP_SSNS * EMPLOYEE)

```

Query 4. List the names of all employees with two or more dependents.

Strictly speaking, this query cannot be done in the basic (original) relational algebra. We have to use the AGGREGATE FUNCTION operation with the COUNT aggregate function. We assume that dependents of the same employee have distinct Dependent\_name values.

```

T1(Ssn, No_of_Dependents) ← Εssn 3 COUNT Dependent_name(DEPENDENT)
T2 ← σ No_of_Dependent ≥ 2(T1)
RESULT ← π Lname, Fname(T2 * EMPLOYEE)

```

ID	Name	Dept_name	Salary
10101	Shrinivasan	Comp.sci	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstin	Physics	95000

## Instructor Relation

instructor relation where the instructor is in the “Physics”

department, we write:

dept name =“Physics” (instructor )

## Relational Calculus

In contrast to Relational Algebra, Relational Calculus is a non-procedural query language, that is, it tells what to do but never explains how to do it.

Relational calculus exists in two forms –

Tuple Relational Calculus (TRC) Filtering variable ranges over tuples

### Notation – {T | Condition}

Returns all tuples T that satisfies a condition.

For example –

{ T.name | Author(T) AND T.article = 'database' }

Output – Returns tuples with 'name' from Author who has written article on 'database'.

TRC can be quantified. We can use Existential ( $\exists$ ) and Universal Quantifiers ( $\forall$ ).

For example –

{ R |  $\exists T \in \text{Authors}(T.article='database' \text{ AND } R.name=T.name)$  }

Output – The above query will yield the same result as the previous one.

### Domain Relational Calculus (DRC)

In DRC, the filtering variable uses the domain of attributes instead of entire tuple values (as done in TRC, mentioned above).

Notation –

{ a<sub>1</sub>, a<sub>2</sub>, a<sub>3</sub>, ..., a<sub>n</sub> | P (a<sub>1</sub>, a<sub>2</sub>, a<sub>3</sub>, ..., a<sub>n</sub>) }

Where a<sub>1</sub>, a<sub>2</sub> are attributes and P stands for formulae built by inner attributes.

For example –

{< article, page, subject > |  $\in$  TP  $\wedge$  subject = 'database'}

Output – Yields Article, Page, and Subject from the relation TP where subject is database.

Just like TRC, DRC can also be written using existential and universal quantifiers. DRC also involves relational operators.

The expression power of Tuple Relation Calculus and Domain Relational Calculus is equivalent to Relational Algebra.

### E-R Diagram

An ER schema diagram for the COMPANY database

### Entities and Attributes

**Entities and Their Attributes.** The basic object that the ER model represents is an **entity**, which 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).

**Composite versus Simple (Atomic) Attributes.** Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meanings. For example, the Address attribute of the EMPLOYEE entity.

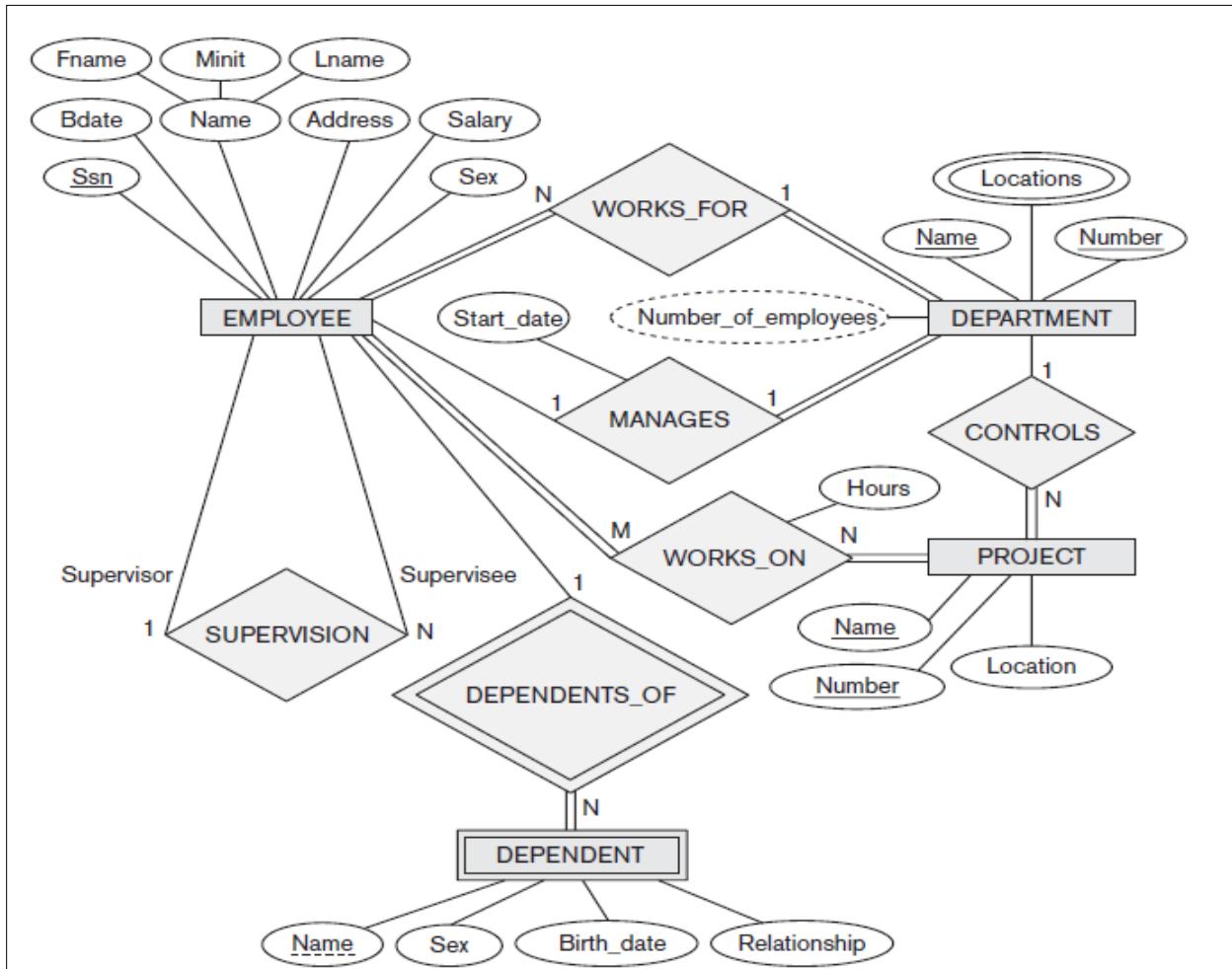


Figure 2.7: E-R Diagram

**Single-Valued versus Multivalued Attributes.** Most attributes have a single value for a particular entity; such attributes are called **single-valued**. For example, Age.

A **multivalued** attribute may have lower and upper bounds to constrain the *number of values* allowed for each individual entity. For example, the Colors attribute of a car.

**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.

Initial Conceptual Design of the COMPANY Database covers

Relationship & type

## **Cardinality**

### **Weak Entity**

**Participation Constraints:** - The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type.

Company schema, with structural constraints specified using (min, max) notation and role names.

## **Enhanced Entity-Relationship (EER) Model**

Semantic data modeling concepts that were incorporated into conceptual data models such as the ER Model. ER model can be enhanced to include these concepts, leading to the **Enhanced ER (EER)** model.

**Subclasses:** - An entity type is used to represent both a type of entity and the entity set or collection of entities of that type that exist in the database. For example, the entity type EMPLOYEE describes the type

(that is, the attributes and relationships) of each employee entity, and also refers to the current set of EMPLOYEE entities in the COMPANY database.

**Super classes:** - We call each of these subgroupings a subclass or subtype of the EMPLOYEE entity type, and the EMPLOYEE entity type is called the superclass or super type for each of these subclasses.

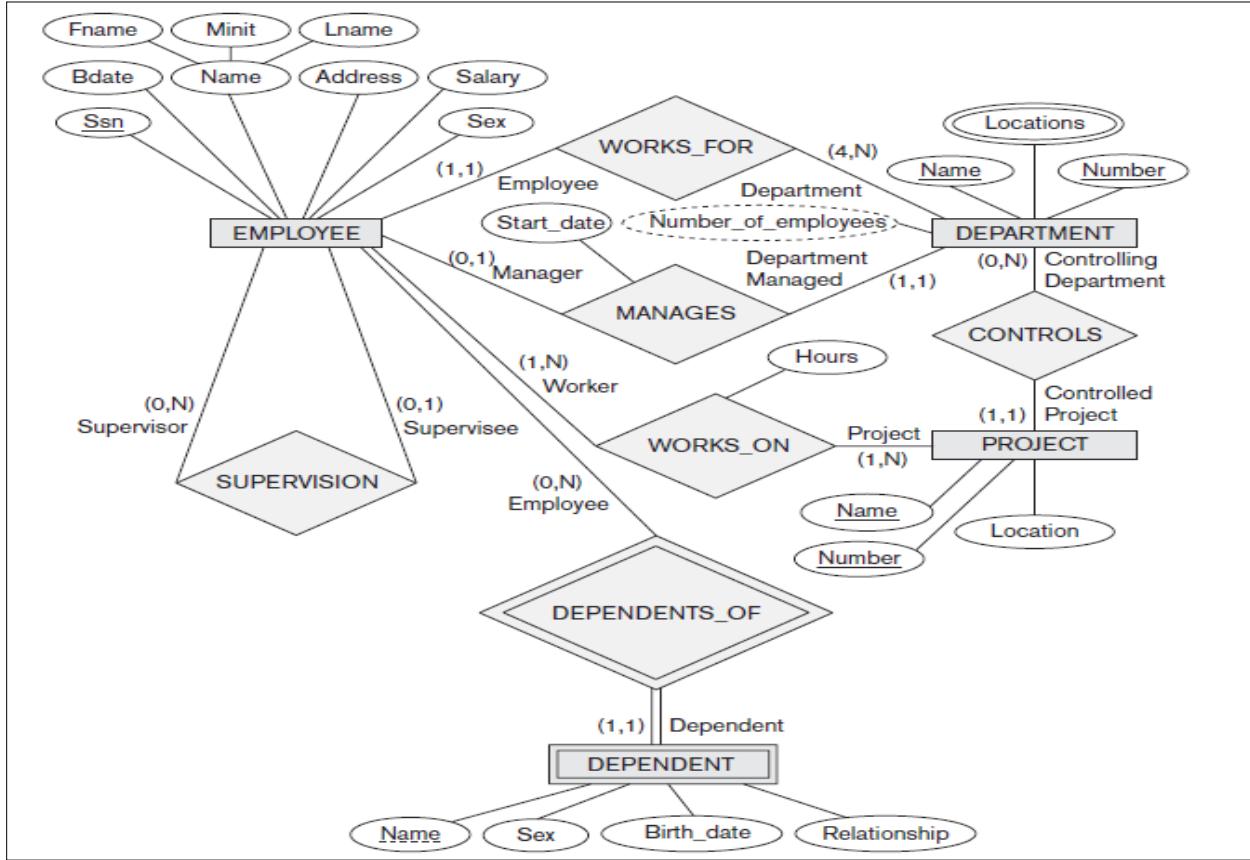


Figure 2.8: Extended E-R Model

### Specialization

**Specialization** is the process of defining a set of *subclasses* of an entity type this entity type is called the **superclass** of the specialization. The set of subclasses that forms a specialization is defined on the basis of some distinguishing characteristic of the entities in the superclass. For example, the set of subclasses {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of the superclass EMPLOYEE that distinguishes among employee entities based on the *job type* of each employee entity.

### Steps for Specialization

- Define a set of subclasses of an entity type.
- Establish additional specific attributes with each subclass.
- Establish additional specific relationship types between each subclass and other entity types or other subclasses.

EER diagram notation to represent subclasses and specialization. Three specializations of EMPLOYEE:  
{SECRETARY, TECHNICIAN, ENGINEER}

{MANAGER}

{HOURLY\_EMPLOYEE, SALARIED\_EMPLOYEE}

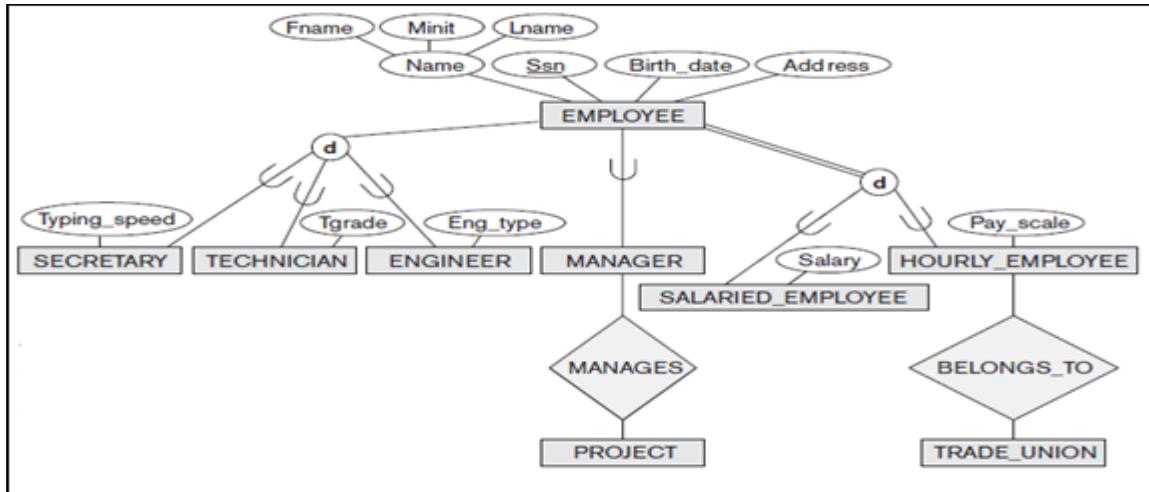


Figure 2.9: Generalization

### Generalization

We can think of a *reverse process* of abstraction in which we suppress the differences among several entity types, identify their common features, and **generalize** them into a single **superclass** of which the original entity types are special **subclasses**. For example, consider the entity types CAR and TRUCK.

Generalization. (a) Two entity types, CAR and TRUCK. (b) Generalizing CAR and TRUCK into the superclass VEHICLE.

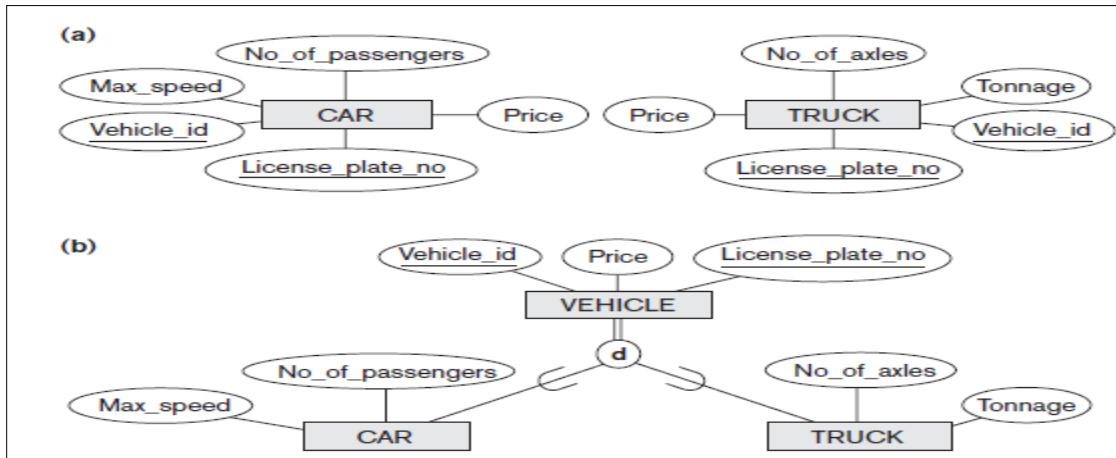


Figure 2.10: Specialization

### Inheritance

We use all the above features of ER-Model in order to create classes of objects in object-oriented programming. The details of entities are generally hidden from the user; this process known as abstraction.

Inheritance is an important feature of Generalization and Specialization. It allows lower-level entities to inherit the attributes of higher-level entities.

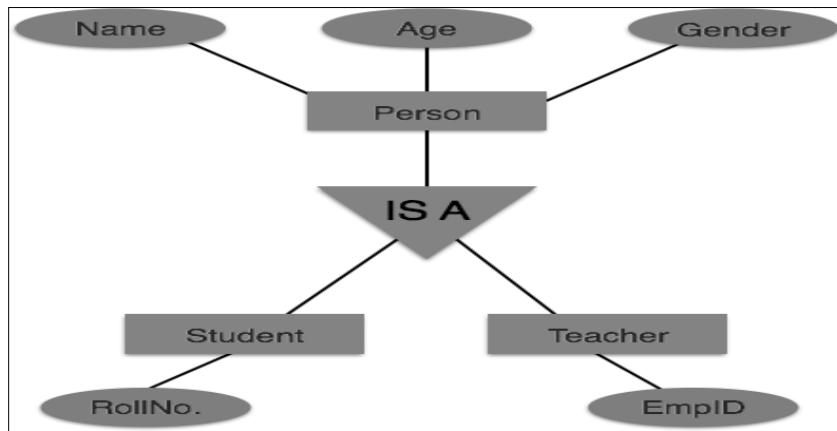


Figure 2.11: Inheritance

For example, the attributes of a Person class such as name, age, and gender can be inherited by lower-level entities such as Student or Teacher.

### Aggregation

Aggregation is a process when the relation between two entities is treated as a single entity. Here the relation between Center and Course is acting as an Entity in relation with Visitor.

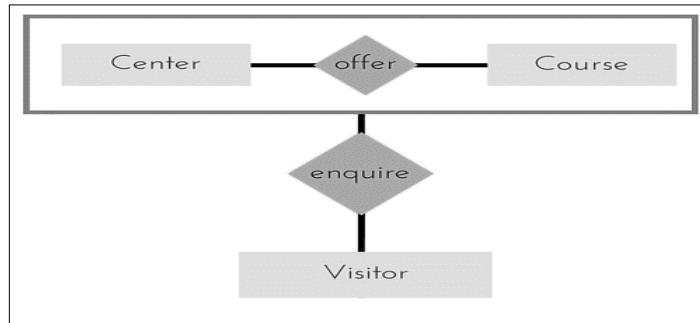


Figure 2.12: Aggregation

**Subject Code: AD403**

**Subject Name: Software Engineering**

**UNIT-I**

---

**Course Objective- To Introduce Software Development Lifecycle and Various Software Process Models**

**Course Outcome-Understand Various Software applications and remember different process model used in software development.**

**Syllabus:** Introduction to Software Engineering Software Development Life Cycles, SDLC Models: Waterfall, V-Model, Prototype Model, Incremental, Evolutionary, RAD, Spiral. Project Planning, Metrics for Project Size Estimation, Project Estimation Techniques, Requirements Gathering and Analysis, Software Requirements Specification (SRS). Software Product and Process Characteristics, Software Process Models, Evolutionary Process Models and Agile processes. Software Process customization and improvement, CMM, Product and Process Metrics, Functional and Non-functional requirements, Requirement Sources and Elicitation Techniques, Analysis Modeling for Function-oriented and Object-oriented software development, Use case Modeling, System and Software Requirement Specifications, Requirement Validation, Traceability.

**Introduction**

**Software: -**

Software is nothing but collection of computer programs and related documents that are planned to provide desired features, functionalities, and better performance.

**Software** is more than just a program code. A program is an executable code, which serves some computational purpose. Software is collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called **software product**.

**Engineering** on the other hand, is all about developing products, using well-defined, scientific principles and methods.

**Software Engineering:** The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

Software product classified in 2 classes:

1. **Generic software:** developed to solution whose requirements are very common stable and well understood by software engineer.
2. **Custom software:** developed for a single customer according to their specification.

#### **Need of Software Engineering: -**

The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working.

**Large software** - It is easier to build a wall than to a house or building, likewise, as the size of software become large engineering has to step to give it a scientific process.

**Scalability-** If the software process were not based on scientific and engineering concepts, it would be easier to re-create new software than to scale an existing one.

**Cost-** As hardware industry has shown its skills and huge manufacturing has lower down the price of computer and electronic hardware. But the cost of software remains high if proper process is not adapted.

**Dynamic Nature-** The always growing and adapting nature of software hugely depends upon the environment in which user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where software engineering plays a good role.

**Quality Management-** Better process of software development provides better and quality software product.

#### **Software Development Life Cycle/Process model/ Software Development Life Cycle: -**

Software Development Life Cycle is a well-defined, structured sequence of stages in software engineering to develop the intended software product.

**Definition:** Software Development Life Cycle (SDLC) is a process used by software industry to design, develop and test high quality software. The SDLC aims to produce high-quality software that meets or exceeds customer expectations, reaches completion within times and costestimates.

#### **What is SDLC?**

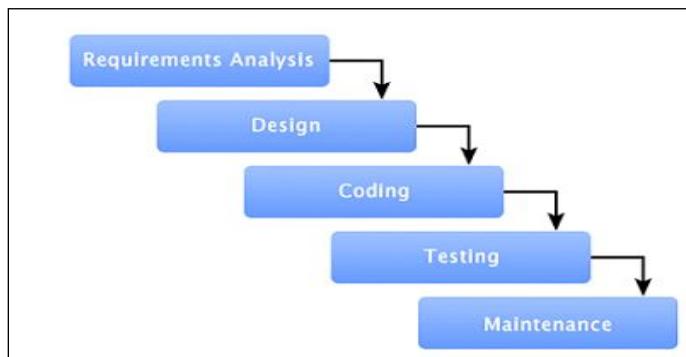
SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace, and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

#### **Basic Activities That Can Be Carried Out in Life Cycle Model Are:**

A few of software development paradigms or process models are defined in Figure 1.1:

##### **1. Waterfall model or linear sequential model or classic life cycle model: -**

Sometimes called the classic life cycle or the waterfall model, the linear sequential model suggests a systematic, sequential approach<sup>5</sup> to software development that begins at the system level and progresses through analysis, design, coding, testing, and maintenance.



**Figure 1.1: Waterfall Model**

**Software requirements analysis:** The requirement's gathering process is intensified and focused specifically on software. To understand the nature of the programs to be built, the software engineer ("analyst") must understand the information domain for the software, as well as required function, behavior, performance, and interface.

**Design:** Software design is a multi-step process that focuses on four distinct attributes of a program: data structure, software architecture, interface representations, and procedural (algorithmic) detail.

**Code generation:** The design must be translated into a machine-readable form. The code generation step performs this task. If design is performed in a detailed manner, code generation can be accomplished mechanically.

**Testing:** Once code has been generated, program testing begins. The testing process focuses on the logical internals of the software, ensuring that all statements have been tested, and on the functional externals; that is, conducting tests to uncover errors and ensure that defined input will produce actual results that agree with required results.

**Maintenance:** Software will undoubtedly undergo change after it is delivered to the customer (a possible exception is embedded software). Change will occur because errors have been encountered, because the software must be adapted to accommodate changes in its external environment

### **Advantages of waterfall model: -**

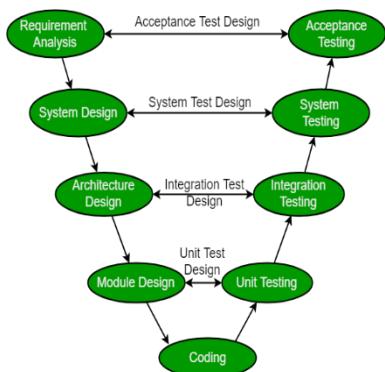
- This model is simple and easy to understand and use.
- Waterfall model works well for smaller projects where requirements are very well understood.
- Documentation is produced at every stage of the software's development. This makes understanding the product designing procedure, simpler.
- After every major stage of software coding, testing is done to check the correct running of the code. help us to control schedules and budgets.

### **Disadvantages of waterfall model: -**

- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.
- High amounts of risk and uncertainty.

### **V-Model**

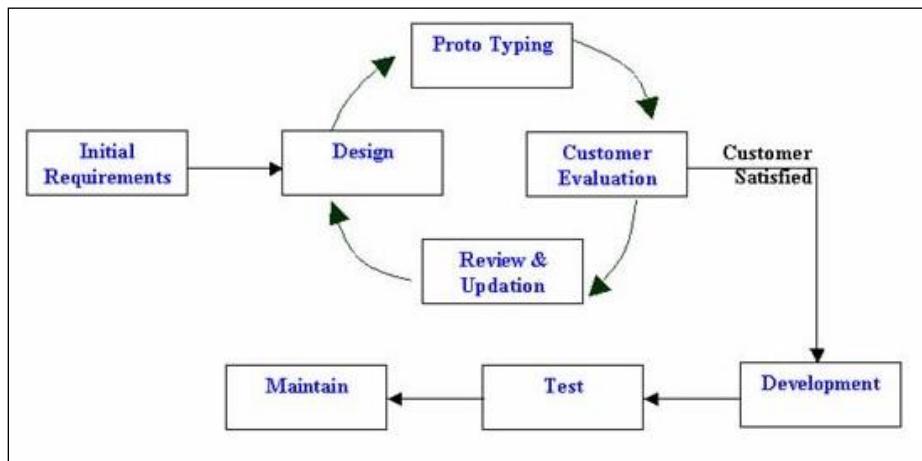
- The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as **Verification and Validation model**.
- The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.
- Under the V-Model, the corresponding testing phase of the development phase is planned in parallel. So, there are Verification phases on one side of the 'V' and Validation phases on the other side. The Coding Phase joins the two sides of the V-Model.



**Figure 1.2: V Model**

### **Prototype Model: -**

A prototype is a toy implementation of the system. A prototype usually exhibits limited functional capabilities, low reliability, and inefficient performance compared to the actual software. A prototype is usually built using several shortcuts. A prototype usually turns out to be a very crude version of the actual system. The Figure 1.2 is shown as below-



**Figure1.3 Prototype Model**

### **Need for a prototype in software development: -**

There are several uses of a prototype. An important purpose is to illustrate the input data formats, messages, reports, and the interactive dialogues to the customer. This is a valuable mechanism for gaining better understanding of the customer's needs:

- how the screens might look like
- how the user interface would behave
- how the system would produce outputs

### **Advantages of Prototyping Model: -**

When prototype is shown to the user, he gets a proper clarity and 'feel' of the functionality of the software, and he can suggest changes and modifications and increasing user confidence.

- When client is not confident about the developer's capabilities, he asks for a small prototype to be built. Based on this model, he judges capabilities of developer.

- It reduces risk of failure, as potential risks can be identified early, and mitigation steps can be taken.
- Iteration between development team and client provides a very good and conductive environment during project.

#### **Disadvantages of Prototyping Model: -**

- Once we get proper requirements from client after showing prototype model, it may be of no use. That is why, sometimes we refer to the prototype as "Throw-away" prototype.
- It is a slow process.
- Too much involvement of client is not always preferred by the developer.
- Too many changes can disturb the rhythm of the development team.

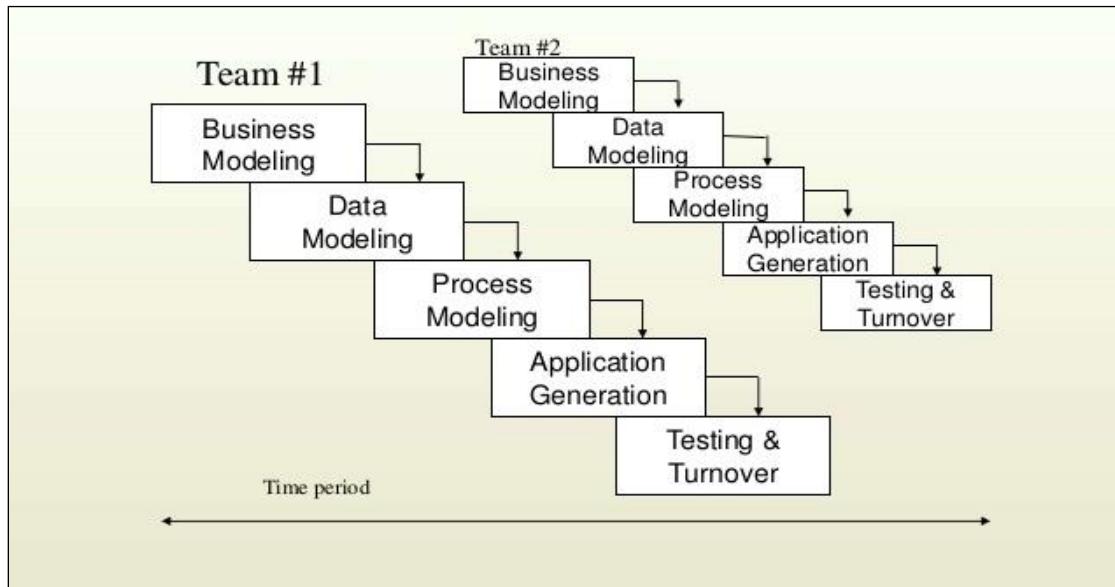
#### **RAPID APPLICATION MODEL**

Rapid application development (RAD) is a software development methodology that uses minimal planning in favor of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product. In RAD model, the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery.

If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a “fully functional system” within very short time periods (e.g., 60 to 90 days). Used primarily for information systems applications, the RAD approach encompasses

1. **Business modeling:** - The information flow among business functions is modeled in a way that answers the following questions: What information drives the business process? What information is generated? Who generates it? Where does the information go? Who processes it?
2. **Data modeling:** - The information flow defined as part of the business modeling phase is refined into a set of data objects that are needed to support the business. The characteristics (called *attributes*) of each object are identified and the relationships between these objects defined.
3. **Process modeling:** - The data objects defined in the data modeling phase are transformed to achieve the information flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.
4. **Application generation:** - RAD assumes the use of fourth generation techniques. Rather than creating software using conventional third generation programming languages the RAD process works to reuse existing program components (when possible) or create reusable components (when necessary). In all cases, automated tools are used to facilitate construction of the software.

5. **Testing and turnover:** - The RAD process emphasizes reuse; many of the program components have already been tested. This reduces overall testing time. However, new components must be tested, and all interfaces must be fully exercised.



**Figure 1.4 RAD Model**

#### **Advantages of the RAD model:**

- Reduced development time.
- Increases re usability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of integration issues.

#### **Disadvantages of RAD model:**

- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modeling skills
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

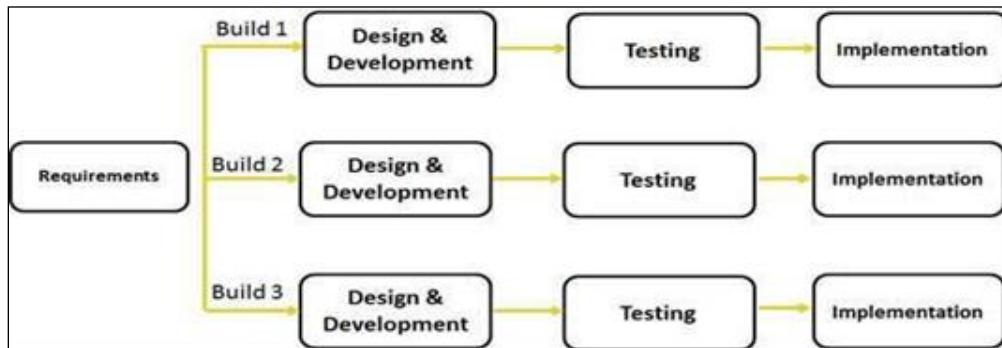
### **When to use RAD model:**

- RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
- It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
- RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

### **Iterative Model design:**

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made, and new functional capabilities are added.

Following is the pictorial representation of Iterative and Incremental model shown in Figure 1.4:



**Figure: 1.5 Iterative Model**

### **Iterative Model Application:**

Like other SDLC models, Iterative and incremental development has some specific applications in the software industry. This model is most often used in the following scenarios:

- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- There is a time to the market constraint.

- A new technology is being used and is being learnt by the development team while working on the project.

### **Evolutionary Process Model is of 2 types**

- Incremental Model And
- Spiral Model

#### **Incremental Model:**

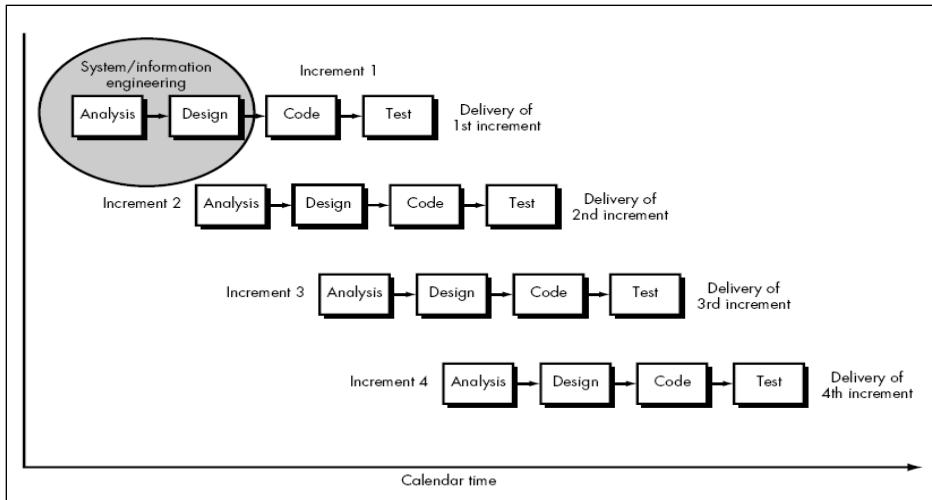
The incremental model combines elements of the linear sequential model (applied repetitively) with the iterative philosophy of prototyping. the incremental model applies linear sequences in a staggered fashion as calendar time progresses. Each linear sequence produces a deliverable “increment” of the software.

#### **Advantages of Incremental model: -**

- Generates working software quickly and early during the software life cycle.
- This model is more flexible – less costly to change scope and requirements.
- It is easier to test and debug during a smaller iteration.
- In this model customer can respond to each built.
- Lowers initial delivery cost.
- Easier to manage risk because risky pieces are identified and handled during it'd iteration.
- There is low risk for overall project failure.
- Customer does not have to wait until the entire system is delivered.

#### **Disadvantages of Incremental model: -**

- Needs good planning and design at the management a technical level.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- Total cost is higher than waterfall.
- Time foundation create problem to complete the project.



**Figure 1.6 Incremental Model**

#### **When to use the Incremental model:**

- This model can be used when the requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some details can evolve with time.
- There is a need to get a product to the market early.
- A new technology is being used
- Resources with needed skill set are not available
- There are some high-risk features and goals.

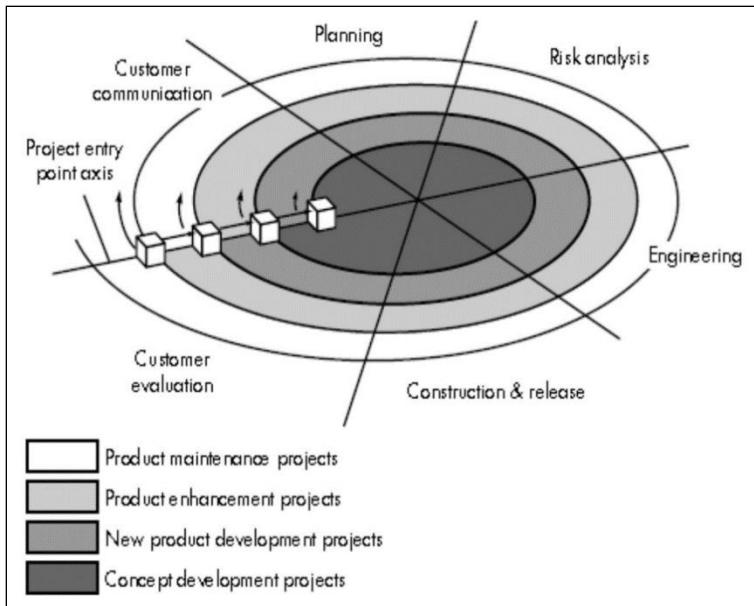
#### **Spiral Model (Boehm's Model)**

The spiral model, originally proposed by Boehm, is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model.

Typically, there are between three and six task regions. In blow figure depicts a spiral model that contains six task regions:

- **Customer communication**—tasks required to establish effective communication between developer and customer.
- **Planning**—tasks required to define resources, timelines, and other project related information.
- **Risk analysis**—tasks required to assess both technical and management risks.
- **Engineering**—tasks required to build one or more representations of the application.

- **Construction and release**—tasks required to construct, test, install, and provide user support (e.g., documentation and training).
- **Customer evaluation**—tasks required to obtain customer feedback based on evaluation of the software representations created during the engineering stage and implemented during the installation stage.



**Figure:1.7 Spiral Model**

#### **Advantages of Spiral model: -**

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added later.
- Software is produced early in the software life cycle.

#### **Disadvantages of Spiral model: -**

- It can be a costly model to use.
- Risk analysis requires highly specific expertise.
- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects.

### **When to use Spiral model:**

- When costs and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment unwise because of potential changes to economic priorities.
- Users are unsure of their needs.

### **LOC and FP estimation**

Lines of code (LOC) and function points (FP) were described as measures from which productivity metrics can be computed.

LOC and FP estimation are distinct estimation techniques.

It is a software metric used to measure the size of a computer program by counting the number of lines in the text of the program's source code. It is used to predict the amount of effort that will be required to develop a program, as well as to estimate programming productivity or effort once the software is produced.

The problems of lines of code (LOC):-

- Different languages lead to different lengths of code.
- It is not clear how to count lines of code.
- A report, screen, or GUI generator can generate thousands of lines of code in minutes.
- Depending on the application, the complexity of code is different.

A Function Point (FP) is a unit of measurement to express the amount of business functionality, an information system (as a product) provides to a user. FPs measure software size.

The Five Components of Function Points are:-

1. Internal Logical Files
2. External Interface Files
3. External Inputs
4. External Outputs

## 5. External Inquiries

### **Empirical models - COCOMO**

The COCOMO is defined as Constructive Cost Model. It is Based on water fall process model. COCOMO consists of a hierarchy of three increasingly detailed and accurate forms.

There are three forms of the COCOMO

- Basic COCOMO (macro estimation) which gives an initial rough estimate of man months and Development time.
- Intermediate COCOMO which gives a more detailed estimate for small to medium sized projects.
- Detailed COCOMO (micro estimation) which gives a more detailed estimate for large projects.

The first level, Basic COCOMO is good for quick, early, rough order of magnitude estimates of software costs, but its accuracy is limited due to its lack of factors to account for difference in project attributes (Cost Drivers).Intermediate COCOMO takes these Cost Drivers into account and Detailed COCOMO additionally accounts for the influence of individual project phases.

**Basic COCOMO** computes software development effort (and cost) as a function of program size. Program size is expressed in estimated thousands of lines of code (KLOC).

COCOMO applies to three classes of software projects:

- Organic projects - "small" teams with "good" experience working with "less than rigid" requirements.
- Semi-detached projects - "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements.
- Embedded projects - developed within a set of "tight" constraints (hardware, software, operational).

Basic COCOMO is good for quick estimate of software costs. However it does not account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and so on.

COCOMO has three different models that reflect the complexities:

- Basic Model – this model would be applied early in a projects development. It will provide a rough estimate early on that should be refined later on with one of the other models.

- Intermediate Model – this model would be used after you have more detailed requirements for a project.
- Advanced Model – when design of the project is complete you can apply this model to further refine your estimate.

Within each of these models there are also three different modes. The mode you choose will depend on your work environment, and the size and constraints of the project itself. The modes are:

- Organic – this mode is used for “relatively small software teams developing software in a highly familiar, in-house environment”.
- Embedded – operating with tight constraints where the product is strongly tied to a “complex of hardware, software, regulations and operational procedures”.
- Semi-detached – an intermediate stage somewhere in between organic and embedded. Projects are usually of moderate size of up to 300,000 lines of code.

#### Equations Used

There are two main equations that are used to calculate effort and schedule time (measured in months). They are:

$$\text{Equation 1} \quad PM = a(KDSI)b * EAF \quad \text{Equation 2} \quad TDEV = c(PM)d$$

Where:

- PM is effort in person-months
- EAF is the effort adjustment factor
- TDEV is the schedule time
- KDSI is the number of lines of code (in thousands)
- a, b, c, and d are all constants based on the mode you are using (refer to Table 1)

Model	A	b	c	D
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

**Table 1.1. List of Constants Based on Mode**

#### Capability Maturity Model (CMM): -

The Software Engineering Institute (SEI) has developed a comprehensive model predicated on a set of software engineering capabilities that should be present as organizations reach different levels of process maturity. To determine an organization's current state of process maturity, the

SEI uses an assessment that results in a five-point grading scheme.

**Level 1: Initial.** The software process is characterized as ad hoc and occasionally even chaotic. Few processes are defined, and success depends on individual effort.

**Level 2: Repeatable.** Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

**Level 3: Defined.** The software process for both management and engineering activities is documented, standardized, and integrated into an organization wide software process. All projects use a documented and approved version of the organization's process for developing and supporting software. This level includes all characteristics defined for level 2.

**Level 4: Managed.** Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled using detailed measures. This level includes all characteristics defined for level 3.

**Level 5: Optimizing.** Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies. This level includes all characteristics defined for level 4.

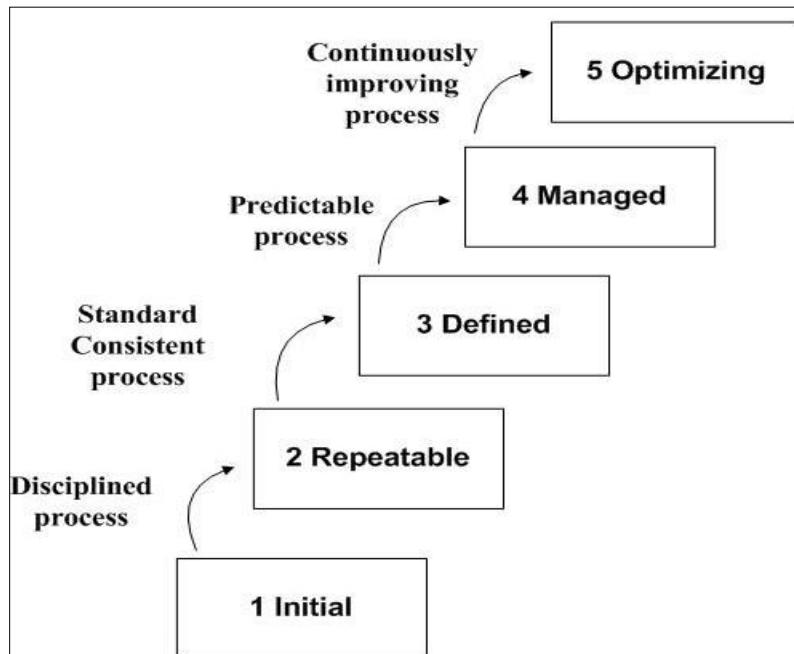


Figure:1.8 CMM Level

## Unified Process Agile Development Model

**Agile Process:**-The word ‘agile’ means able to think quickly and clearly.

In business, ‘agile’ is used for describing ways of planning and doing work wherein it is understood that making changes as needed is an important part of the job.

Agile development model is also a type of Incremental model. Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly tested to ensure software quality is maintained. It is used for time critical applications. Extreme Programming (XP) is currently one of the most well-known agile development life cycle models.



**Figure 1.9 Agile Model**

**1. Requirements gathering:** In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

**2. Design the requirements:** When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

**3. Construction/ iteration:** When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

**4. Testing:** In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

**5. Deployment:** In this phase, the team issues a product for the user's work environment.

**6. Feedback:** After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

#### **Advantages of Agile model:**

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers, and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Even late changes in requirements are welcomed

#### **Disadvantages of Agile model:**

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what outcome that they want.

### **Extreme Programming**

Extreme Programming (XP) is an agile software development framework that aims to produce higher quality software, and higher quality of life for the development team. XP is the most specific of the agile frameworks regarding appropriate engineering practices for software development.

Extreme Programming is based on the following values –

- Communication
- Simplicity
- Feedback
- Courage
- Respect

Extreme Programming takes the effective principles and practices to extreme levels.

- Code reviews are effective as the code is reviewed all the time.
- Testing is effective as there is continuous regression and testing.
- Design is effective as everybody needs to do refactoring daily.
- Integration testing is important as integrate and test several times a day.
- Short iterations are effective as the planning game for release planning and iteration planning.

Types of Requirements: -

User Requirements: It is a collection of statement in natural language and description of the services the

• system provides and its operational limitation. It is written for customer. System Requirement: It is a structured document that gives the detailed description of the system services

.• It is written as a contract between client and contractor

Software Requirement Specification: - SRS is a document created by system analyst after the requirements are collected from various stakeholders. SRS defines how the intended software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc. The requirements received from client are written in natural language. It is the responsibility of system analyst to document the requirements in technical language so that they can be comprehended and useful by the software development team. SRS should come up with following features:

User Requirements are expressed in natural language.

- Technical requirements are expressed in structured language, which is used inside the organization
- Design description should be written in Pseudo code.
- Format of Forms and GUI screen prints.
- Conditional and mathematical notations for DFDs etc

.• Software Requirements: - We should try to understand what sort of requirements may arise in the requirement elicitation phase and what kinds of requirements are expected from the software system. Broadly software requirements should be categorized in two categories:

1. Functional Requirement
2. Non Functional Requirement

**FUNCTIONAL REQUIREMENTS:** It should describe all requirement functionality or system services. The customer should provide statement of service. it should be clear how the system should be reacting to particular input and how a particular system should behave in particular situation. Functional requirement is heavily depending upon the type of software expected users and the type of system where the software is used. It describes system services in detail.

**NON-FUNCTIONAL REQUIREMENTS:** Requirements, which are not related to functional aspect of software, fall into this category. They are implicit or expected characteristics of software; which users make assumption of. Non-functional are more critical than functional requirement if the non-functional requirement do not meet then the complete system is of no use.

**REQUIREMENT SOURCES AND ELICITATION TECHNIQUES:** Requirements Sources:- In a typical system, there will be many sources of requirements and it is essential that all potential sources are identified and evaluated for their impact on the system.

1. Goals: -
2. Domain knowledge
3. System stakeholders:
4. The operational environment:
5. The organizational environment

**Elicitation techniques:** - When the requirements sources have been identified the requirements, engineer can start eliciting requirements from them. It also means requirement discovery. This subtopic concentrates on techniques for getting human stakeholders to articulate their requirements. This is a very difficult area and the requirements engineer needs to be sensitized to the fact that (for example) users may have difficulty describing their tasks, may leave important information unstated, or may be unwilling or unable to cooperate. It is particularly important to understand that elicitation is not a passive activity and that even if cooperative and articulate stakeholders are available, the requirements engineer has to work hard to elicit the right information. A number of techniques will be covered, but the principal one

Interviews:-

Scenarios:

Prototypes:

Facilitated meetings

Observation

## ANALYSIS MODELING FOR FUNCTION ORIENTED AND OBJECT ORIENTED SOFTWARE DEVELOPMENT :

Analysis model: - The analysis model must achieve three primary objectives: To describe what the customer requires(analysis)

- To establish a basis for the creation of software with a combination of text and design are used to represent
- the software requirement. To define a set of requirements that can be validated once the software is built.

The elements of analysis model: - At the core of the model lies the data dictionary—a repository that contains descriptions of all data objects consumed or produced by the software. Three different diagrams surround the core. The entity relation diagram (ERD) depicts relationships between data objects. The ERD is the notation that is used to conduct the data modeling activity. The attributes of each data object noted in the ERD can be described using a data object description. The data flow diagram (DFD) serves two purposes: (1) to provide an indication of how data are transformed as they move through the system and (2) to depict the functions (and sub functions) that transform the data flow

### **USE Case Modelling**

The Use-case model is defined as a model which is used to show how users interact with the system in order to solve a problem. As such, the use case model defines the user's objective, the interactions between the system and the user, and the system's behavior required to meet these objectives.

Various model elements are contained in use-case model, such as actors, use cases, and the association between them.

### Components of Basic Model

There are various components of the basic model:

1. Actor
2. Use Case
3. Associations

## Actor

Usually, actors are people involved with the system defined on the basis of their roles. An actor can be anything such as human or another external system.

## Use Case

The use case defines how actors use a system to accomplish a specific objective. The use cases are generally introduced by the user to meet the objectives of the activities and variants involved in the achievement of the goal.

## Associations

Associations are another component of the basic model. It is used to define the associations among actors and use cases they contribute in. This association is called communicates-association.

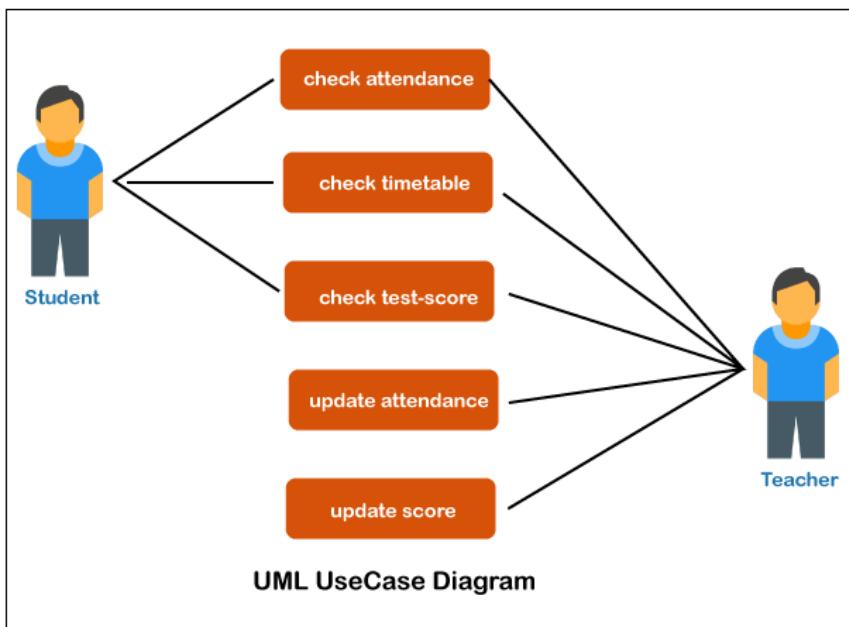


Figure 1.10-Use Case Model

## Software Requirement Specification

**Software Requirement Specification [SRS]:** - A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase. The Software Requirements Specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description, a detailed functional description, a representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria, and other information pertinent to requirements.

Characteristics of SRS:

- Correct: Requirement must be correctly mentioned and realistic by nature.
- Unambiguous: Transparent and plain SRS must be written

- Complete: To make the SRS complete I should be specified what a software designer wants to create on software.
- Consistent: If there are no conflicts in the specified requirement then SRS is said to be consistent
- Stability: The SRS must contain all the essential requirement. Each requirement must be clear and
- Downloaded from be.rgpvnotes.in Page no: 10 Follow us on facebook to get real-time updates from RGPV explicit.
- Verifiable: the SRS should be written in such a manner that the requirement that is specified within it
- must be satisfied by the software.
- Modifiable: It can easily modify according to user requirement.
- Traceable: If origin of requirement is properly given of the requirement are correctly mentioned then
- such a requirement is called as traceable requirement.

**REQUIREMENTS VALIDATION:** The work products produced as a consequence of requirements engineering are assessed for quality during a validation step. Requirements validation examines the specification to ensure that all system requirements have been stated unambiguously; that inconsistencies, omissions, and errors have been detected and corrected; and that the work products conform to the standards established for the process, the project, and the product. The primary requirements validation mechanism is the formal technical review. The review team includes system engineers, customers, users, and other stakeholders who examine the system specification looking for errors in content or interpretation, areas where clarification may be required, missing information, inconsistencies , conflicting requirements, or unrealistic (unachievable) requirements

**TRACEABILITY:** Traceability is a property of an element of documentation or code that indicates the degree to which it can be traced to its origin or "reason for being". Traceability also indicates the ability to establish a predecessor-successor relationship between one work product and another. A work product is said to be traceable if it can be proved that it complies with its specification. For example, a software design is said to be traceable if it satisfies all the requirements stated in the software requirements specification. Examples of traceability include:

- External source to system requirements

- System requirements to software requirements
- Software requirements to high level design
- High level design to detailed design
- Detailed design to code
- Software requirement to test case.

**Chameli Devi Group of Institutions**

**Department of Information Technology**

**Subject Notes**

**AD 402 – Database Management System**

**B.Tech, AD-4<sup>th</sup>Semester**

**Syllabus:** Basic Concepts: Introduction to DBMS, File system vs DBMS, Advantages of database systems, Database System architecture, Data models, Schemas and instances, Data independence, Functions of DBA and designer, Entities and attributes, Entity types, Key attributes, Relationships, Defining the E-R diagram of database.

---

**Course Objective:** To understand fundamental knowledge of file system, database concepts and use of relational database.

---

**Course Outcome (CO1):** Compare file system and DBMS and explain how DBMS is better than traditional File

Processing Systems.

---

## **Unit I**

### **Basic Concept & Introduction:**

The database is a collection of related data. Database management system is software designed to assist the maintenance and utilization of large-scale collection of data. DBMS came into existence in 1960 by Charles. Integrated data store which is also called as the first general-purpose DBMS. Again in 1960 IBM brought IMS-Information management system. In 1970 Edgar Codd at IBM came with a new database

called RDBMS. In 1980 then came SQL Architecture- Structure Query Language. In 1980 to 1990 there were advances in DBMS,e.g. DB2, ORACLE.

#### **Data:**

- Data is raw fact or figures or entity.
- When activities in the organization take place, the effect of these activities needs to be recorded which is known as Data.

#### **Information:**

Processed data is called information

The purpose of data processing is to generate the information required for carrying out the business activities.

Below are the different functions of DBMS:

- **Data Capture:** Collection of data from the place of origination like taking attendance in the class.
- **Data Classification:** Categorization of captured data based on different dimensions like size, type, segregation of data images, audio, video.
- **Data Storage:** The categorized data must be stored, and it must maintain data persistence and integrity of it like we are storing details of students and it should remain same for years.
- **Data Arranging:** Arrangement of the data in the database in a proper manner that will help the user to understand how we are going to use it.
- **Data Retrieval:** Data required for retrieval so there must be mechanism through which they can be retrieved efficiently.
- **Data Maintenance:** It is the database to keep it up-to-date. One of the important feature of the DBMS through which data can be updated and remain useful for the user.
- **Data Verification:** Before storing the data, it must be checked based on the syntax and semantics to avoid errors. To keep data persistence, it is mandatory to verify it before the storage and must store in the database the way it supports.
- **Data Editing:** Providing tools that facilitate to perform change and update operation on DBMS.

- **Data Transcription:** Providing facility to transfer one form to another. User requirement is changes very frequently and in the digital world expectations from the users are also very high. So, DBMS must provide facility to change it from one form to other.

### **Database:**

A database is basically a set of data and it contains interrelated data. The database contains set of algorithm and rules through which data can be stored in it in a systematic manner. So, all the data fields must be related to each other. For example, where it is used suppose an organization from attendance do salary calculation and different allowances are given to them can be done through the Database. To keep a record nowadays from billing an item, to take attendance in the classroom database is required.

### **Evolution of DBMS: -**

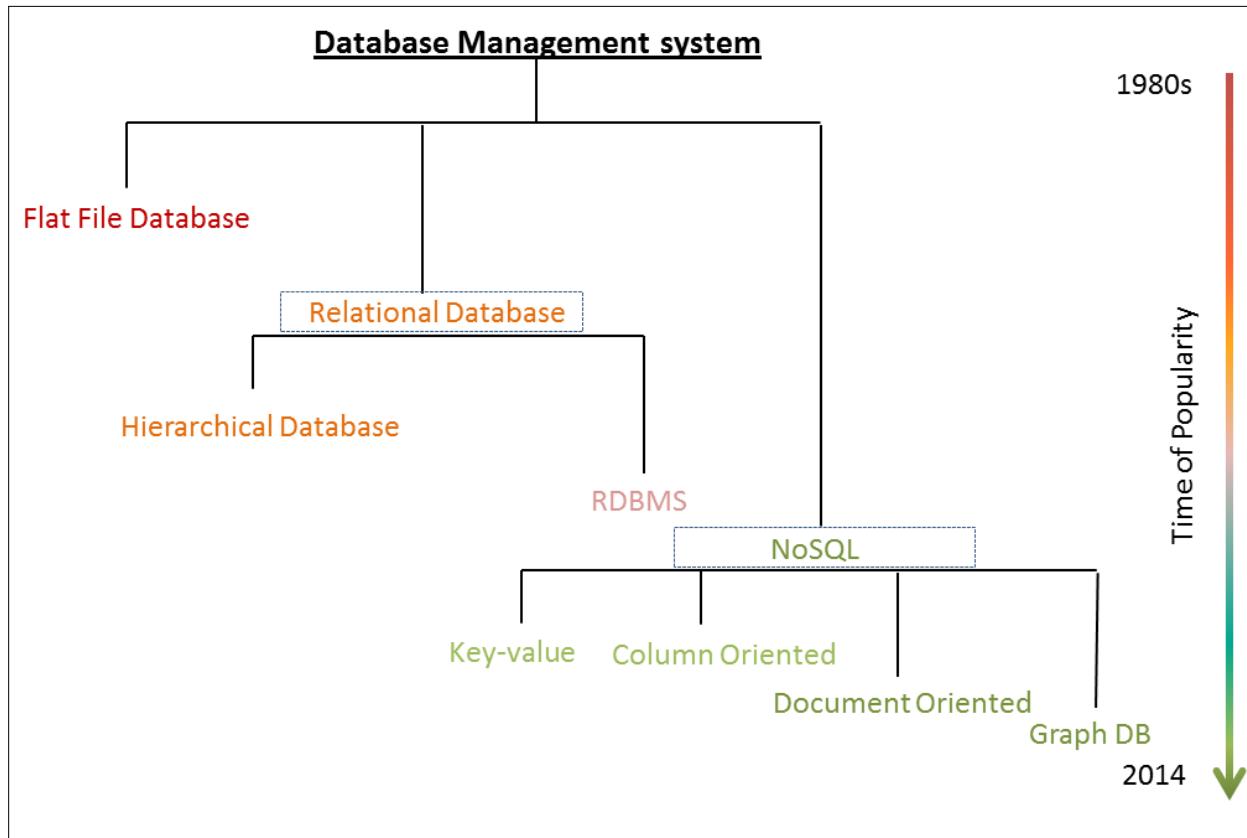


Figure 1.1: Evolution of DBMS

### **File Oriented Approach:**

Computer System comes into the picture to store the records and after performing computation producing information. As compared to the manual work done by the human being they are more

accurate and fast and secure also. Such system stores a group of records like department wise or class wise record and each group is having separate files, or for a category. Such arrangements called file processing system and in which each branch or department is having its own file and a dedicated application designed for to perform an operation on it. This is mainly to keep a record of the students like fees details, result details, and contact details and whenever some information is required then through application program it can be accessed. Still, file processing approach of data storage and access is used but it has some drawbacks.

### **Database Management System:**

A Database Management System (DBMS) is a collection of program that enables user to create and maintain a database.

The DBMS is hence a general-purpose software system that facilitates the process of defining constructing and manipulating database for various applications.

**Table 1.1: Comparison between File System & DBMS**

<b>DBMS</b>	<b>File System</b>
DBMS is a collection of data and user is not required to write the procedures for managing the database.	File system is a collection of data. Any management with the file system, user has to write the procedures
DBMS provides an abstract view of data that hides the details.	File system gives the details of the data representation and Storage of data.
DBMS is efficient to use since there are wide varieties of sophisticated techniques to store and retrieve the data.	In File system storing and retrieving of data cannot be done efficiently.
DBMS takes care of Concurrent access using some form of locking.	Concurrent access to the data in the file system has many problems like  a. Reading the file while other deleting some information, updating some information
DBMS has crash recovery mechanism DBMS protects user from the effects of system failures.	File system doesn't provide crash recovery mechanism.  Eg. While we are entering some data into the file if System crashes then content of the file is lost.
DBMS has a good protection mechanism.	Protecting a file under file system is very difficult.

A database management system is, well, a system used to manage databases.

## **RDBMS:**

A relational database management system is a database management system used to manage relational databases. A relational database is one where tables of data can have relationships based on primary and foreign keys.

## **Advantages of DBMS:**

Due to its centralized nature, the database system can overcome the disadvantages of the file system-based system

- **Concurrent use:** A database system provides facility to access database several users concurrently. Let us understand it by taking an example that a movie online booking system database employee of different branches access the database in a concurrent manner. Each employee can handle customers at their individual desk, Able to see the seats available for the booking from the interface provided to them.
- **Structured data:** One of the important features of the database system is not only to store the data and providing access to the database. But it also provides details about the data and how to access and use it. Taking an example when you are going for an online form of exam then details about each and every field is given, what type of data it accepts and format details also.
- **Data Independence:** Another important characteristic of the DBMS is data independence in which application through which user can interact with the user is not dependent on the physical data storage. So, if there is any change in the application program it will not affect the data stored in the database.
- **Integrity:** This characteristic of the DBMS deals with one of the basic properties of the data called integrity, in which if some data is saved in the database and later retrieved from the database it must be same. This also covers restriction on the unauthorized access of the data that can make changes in the data sets.
- **Transaction of Data:** A transaction is a set of actions that are done in a database to transfer it from one consistent state to another. If it is not handled properly it may result in inconsistent state and loss of data, so DBMS has certain constraints to maintain the basic properties of transaction like atomicity, integrity, isolation, and durability.
- **Data Persistence:** This is one of the basic characteristics of the database because data can be retained in the database for years and it should be in the same condition. In the banking system, a user will open his account and lifelong maintain it, or a user has opted an LIC policy, or mediclaim policy.

## **A modern DBMS has the following characteristics:**

**Real-world entity:** A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use students as an entity and their age as an attribute.

**Relation-based tables:** DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.

**Isolation of data and application:** A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.

**Less redundancy:** DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.

**Consistency:** Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state.

**Query Language:** DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.

**ACID Properties:** DBMS follows the concepts of Atomicity, Consistency, Isolation, and Durability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database.

**Multiuser and Concurrent Access:** DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.

**Multiple views:** DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.

**Security:** Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage.

### **DBMS Architecture:**

The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent **n** modules, which can be independently modified, altered, changed, or replaced.

In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.

### 3-tier Architecture:

**Database (Data) Tier:** At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.

**Application (Middle) Tier:** At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.

**User (Presentation) Tier:** End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

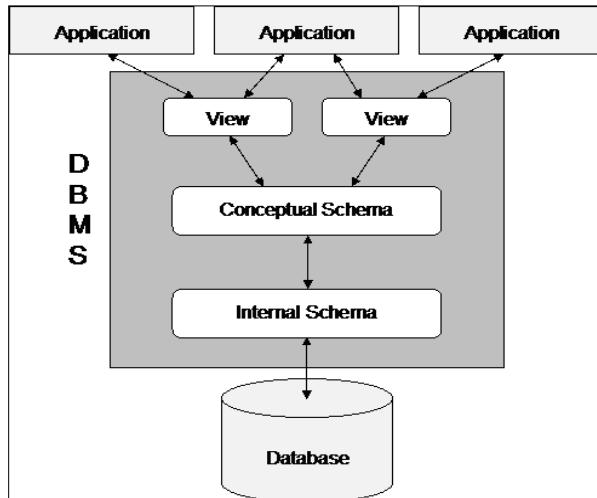


Figure 1.2: DBMS Three Tier Architecture

### DBMS Architecture Component:

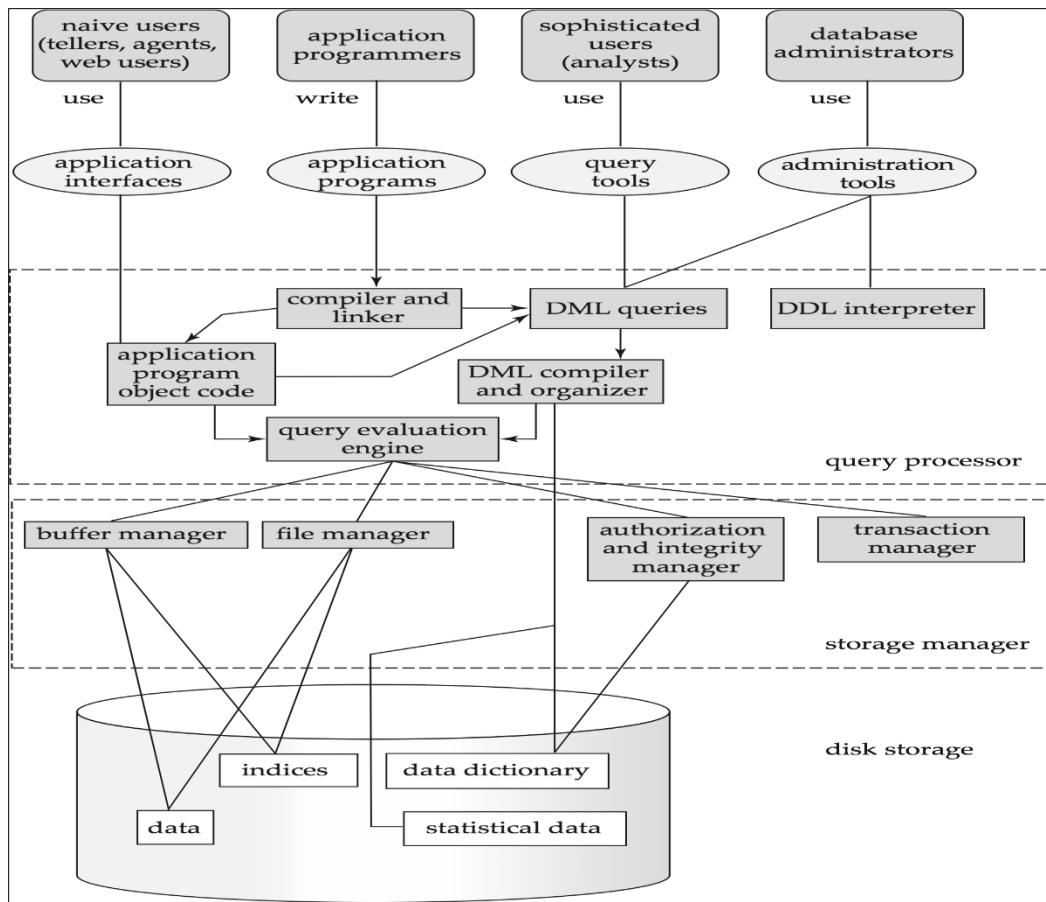


Figure 1.3: DBMS Architecture Component

#### Database Users:

Users are differentiated by the way they expect to interact with the system:

#### Application programmers:

Application programmers are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces.

Rapid application development (RAD) tools are tools that enable an application programmer to construct forms and reports without writing a program.

#### Sophisticated users:

Sophisticated users interact with the system without writing programs. Instead, they form their requests in a database query language.

**Specialized users:**

Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework.

**Naïve users:**

Naïve users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.

**Database Administrator:**

Coordinates all the activities of the database system. The database administrator has a good understanding of the enterprise's information resources and needs.

**Query Processor:**

The query processor will accept query from user and solves it by accessing the database.

Parts of Query processor:

**DDL interpreter**

This will interpret DDL statements and fetch the definitions in the data dictionary.

**DML compiler**

a. This will translates DML statements in a query language into low level instructions that the query evaluation engine understands.

b. A query can usually be translated into any of a number of alternative evaluation plans for same query result DML compiler will select best plan for query optimization.

**Query evaluation engine**

This engine will execute low-level instructions generated by the DML compiler on DBMS.

**Storage Manager/Storage Management:**

A storage manager is a program module which acts like interface between the data stored in a database and the application programs and queries submitted to the system.

**The storage manager components include:**

**Authorization and integrity manager:** Checks for integrity constraints and authority of users to access data.

**Transaction manager:** Ensures that the database remains in a consistent state although there are system failures.

**File manager:** Manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

**Buffer manager:** It is responsible for retrieving data from disk storage into main memory. It enables the database to handle data sizes that are much larger than the size of main memory.

Data structures implemented by storage manager.

**Data files:** Stored in the database itself.

**Data dictionary:** Stores metadata about the structure of the database.

**Indices:** Provide fast access to data items.

#### **Data Models:**

Data models define how the logical structure of a database is modelled. Data Models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside the system.

The very first data model could be flat datamodels, where all the data used are to be kept in the same plane. Earlier data models were not so scientific; hence they were prone to introduce lots of duplication and update anomalies.

**There are many kinds of data models. Some of the most common ones include:**

- Hierarchical database model
- Relational model
- Network model
- Object-oriented database model
- Entity-relationship model
- Object-relational model

The most common model, the relational model stores data into tables, also known as relations, each of which consists of columns and rows. Each column lists an attribute of the entity in question, such as

price, zip code, or birth date. Together, the attributes in a relation are called a domain. A particular attribute or combination of attributes is chosen as a primary key that can be referred to in other tables, when it's called a foreign key.

Each row, also called a tuple, includes data about a specific instance of the entity in question, such as a particular employee.

The model also accounts for the types of relationships between those tables, including one-to-one, one-to-many, and many-to-many relationships. Here's an example:

#### Relational Model:

The diagram illustrates a relational model with a table labeled "table (relation)". The table has five columns: SID, SName, SAge, SClass, and SSection. The rows are labeled 1101 through 1105 and represent tuples. An arrow points from the word "attributes" to the first column. Another arrow points from the word "tuple" to the second row. A vertical arrow points down from the word "column" to the third column. The table data is as follows:

SID	SName	SAge	SClass	SSection
1101	Alex	14	9	A
1102	Maria	15	9	A
1103	Maya	14	10	B
1104	Bob	14	9	A
1105	Newton	15	10	B

Figure 1.4: Relational Model

Within the database, tables can be normalized, or brought to comply with normalization rules that make the database flexible, adaptable, and scalable. When normalized, each piece of data is atomic, or broken into the smallest useful pieces.

Relational databases are typically written in Structured Query Language (SQL). The model was introduced by E.F. Codd in 1970.

The main highlights of this model are –

- Data is stored in tables called relations.

- Relations can be normalized.
- In normalized relations, values saved are atomic values.
- Each row in a relation contains a unique value.
- Each column in a relation contains values from a same domain.

### Hierarchical Model:

The hierarchical model organizes data into a tree-like structure, where each record has a single parent or root. Sibling records are sorted in a particular order. That order is used as the physical order for storing the database. This model is good for describing many real-world relationships.

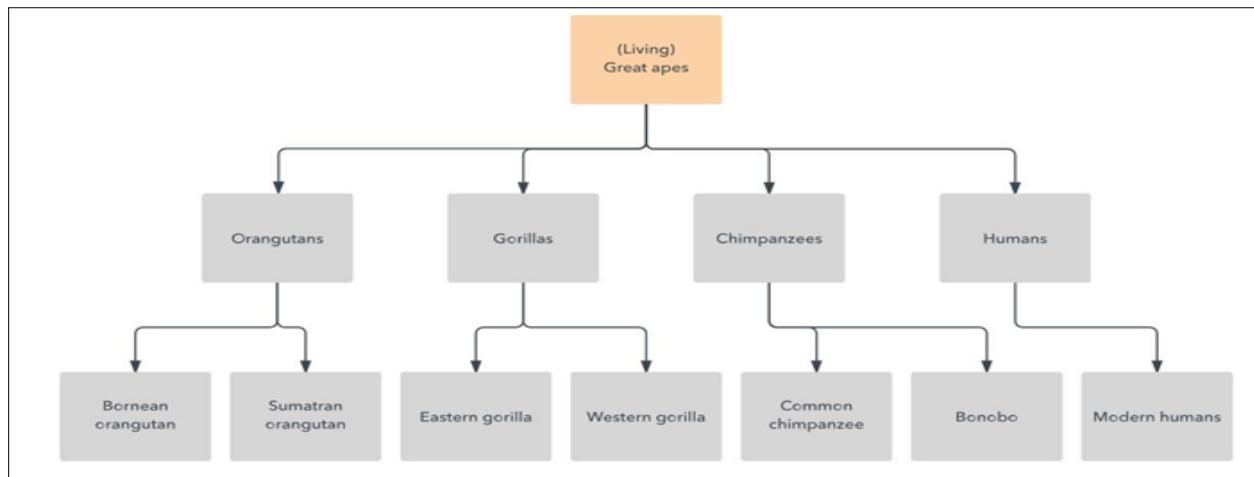


Figure 1.5: Hierarchical Model

### Network Model:

The network model builds on the hierarchical model by allowing many-to-many relationships between linked records, implying multiple parent records. Based on mathematical set theory, the model is constructed with sets of related records. Each set consists of one owner or parent record and one or more member or child records. A record can be a member or child in multiple sets, allowing this model to convey complex relationships.

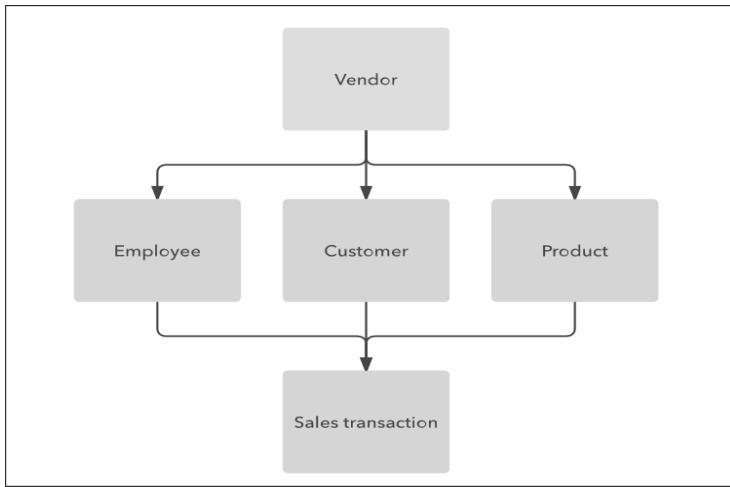


Figure 1.6: Network Model

#### **Object-Oriented Database Model:**

This model defines a database as a collection of objects, or reusable software elements, with associated features and methods. There are several kinds of object-oriented databases:

A multimedia database incorporates media, such as images, that could not be stored in a relational database.

A hypertext database allows any object to link to any other object. It's useful for organizing lots of disparate data, but it's not ideal for numerical analysis.

The object-oriented database model is the best known post-relational database model, since it incorporates tables, but isn't limited to tables. Such models are also known as hybrid database models.

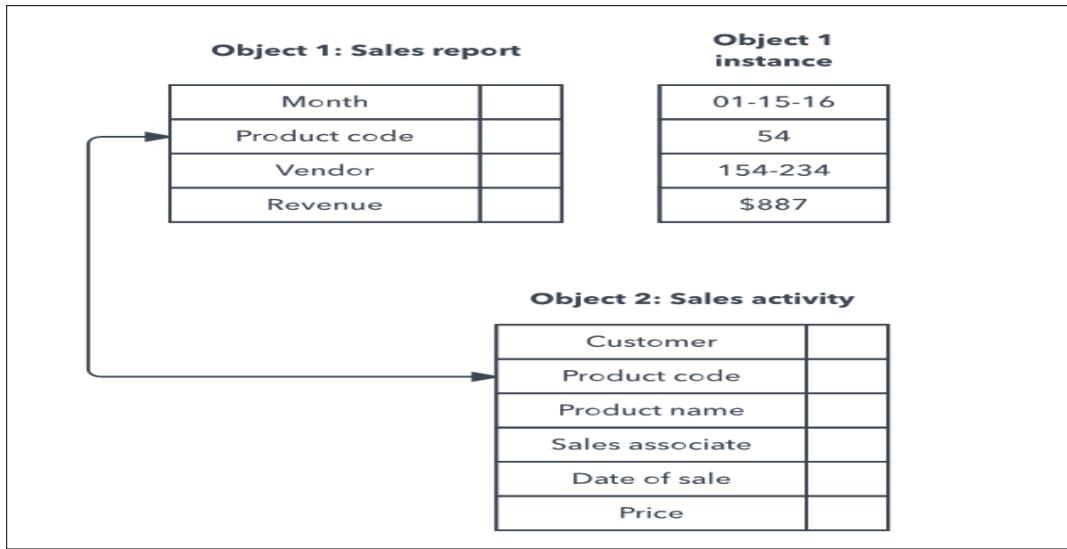


Figure 1.7: Object Oriented Database Model

### Object-Relational Model:

This hybrid database model combines the simplicity of the relational model with some of the advanced functionality of the object-oriented database model. In essence, it allows designers to incorporate objects into the familiar table structure.

Languages and call interfaces include SQL3, vendor languages, ODBC, JDBC, and proprietary call interfaces that are extensions of the languages and interfaces used by the relational model.

### Entity-Relationship Model:

This model captures the relationships between real-world entities much like the network model, but it isn't as directly tied to the physical structure of the database. Instead, it's often used for designing a database conceptually.

Here, the people, places, and things about which data points are stored are referred to as entities, each of which has certain attributes that together make up their domain. The cardinality, or relationships between entities, are mapped as well.

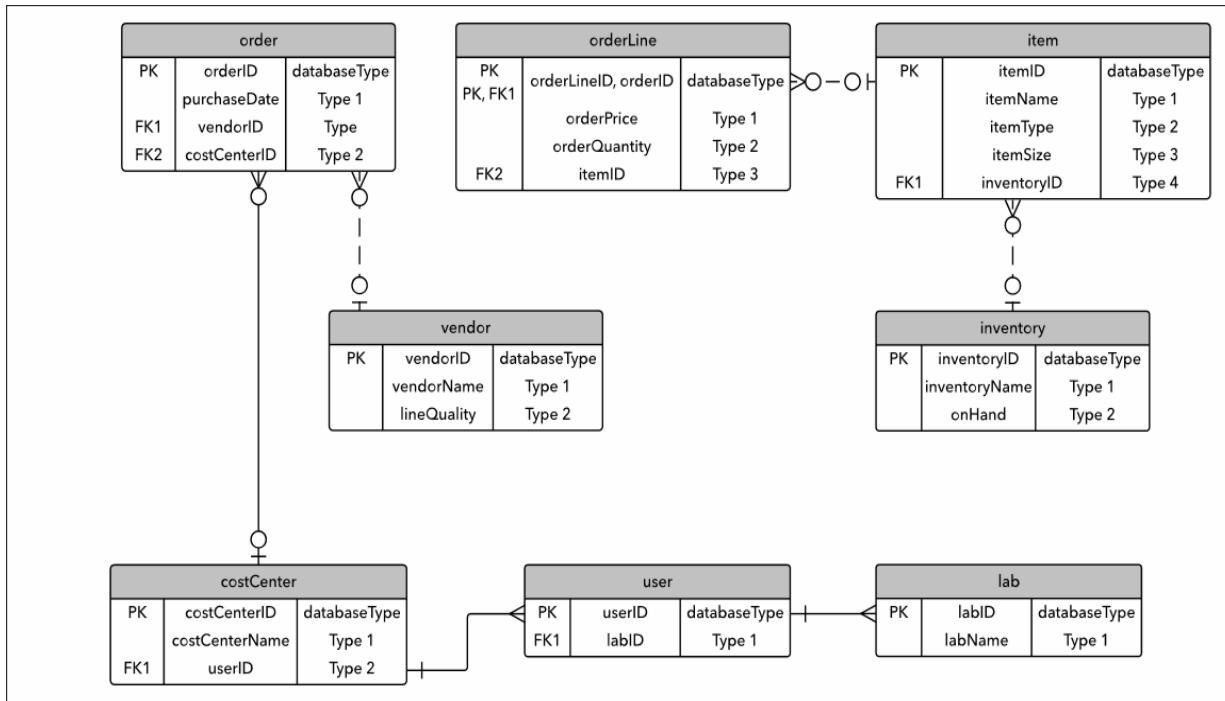


Figure 1.8: E-R Diagram

### NoSQL Database Models:

In addition to the object database model, other non-SQL models have emerged in contrast to the relational model: The graph database model, which is even more flexible than a network model, allowing any node to connect with any other. The multi valued model, which breaks from the relational model by allowing attributes to contain a list of data rather than a single data point.

The document model, which is designed for storing and managing documents or semi-structured data, rather than atomic data.

### Schema and Instance in DBMS:

Design of a database is called the schema. Schema is of three types:

- **Physical schema:** The design of a database at physical level is called physical schema, how the data stored in blocks of storage is described at this level.
- **Logical schema:** Design of database at logical level is called logical schema, programmers and database administrators work at this level, at this level data can be described as certain types of data records gets stored in data structures, however the internal details such as implementation of data structure is hidden at this level (available at physical level).
- **View schema:** Design of database at view level is called view schema. This generally describes end user interaction with database systems.

**Instances:** The data stored in database at a particular moment of time is called instance of database. Database schema defines the variable declarations in tables that belong to a particular database the value of these variables at a moment of time is called the instance of that database.

### **Data Independence:**

A database system normally contains a lot of data in addition to users' data. For example, it stores data about data, known as metadata, to locate and retrieve data easily. It is rather difficult to modify or update a set of metadata once it is stored in the database. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users.

### **Logical Data Independence:**

Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation.

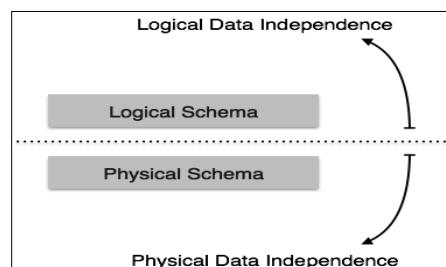


Figure 1.9: Data Independence

Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.

### **Physical Data Independence:**

All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.

For example, in case we want to change or upgrade the storage system itself – suppose we want to replace hard-disks with SSD – it should not have any impact on the logical data or schemas.

Requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job.

### DBA (Database Administrator)

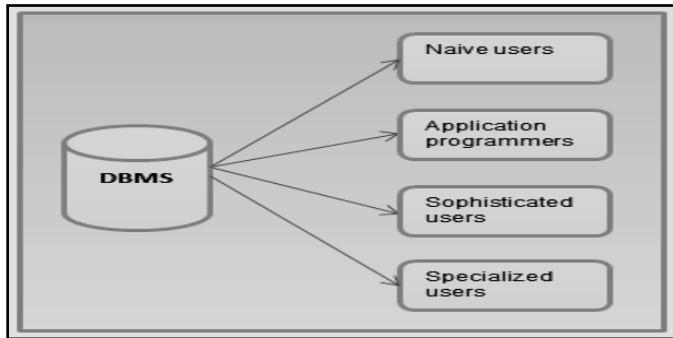


Figure 1.10: DBA

A data administration (also known as a database administration manager, data architect, or information center manager) is a high-level function responsible for the overall management of data resources in an organization. In order to perform its duties, the DA must know a good deal of system analysis and programming.

**These are the functions of a data administrator (not to be confused with database administrator functions):**

1. Data policies, procedures, standards
2. Planning- development of organization's IT strategy, enterprise model, cost/benefit model, design of database environment, and administration plan.
3. Data conflict (ownership) resolution
4. Data analysis- Define and model data requirements, business rules, operational requirements, and maintain corporate data dictionary
5. Internal marketing of DA concepts
6. Managing the data repository

### E-R Diagram

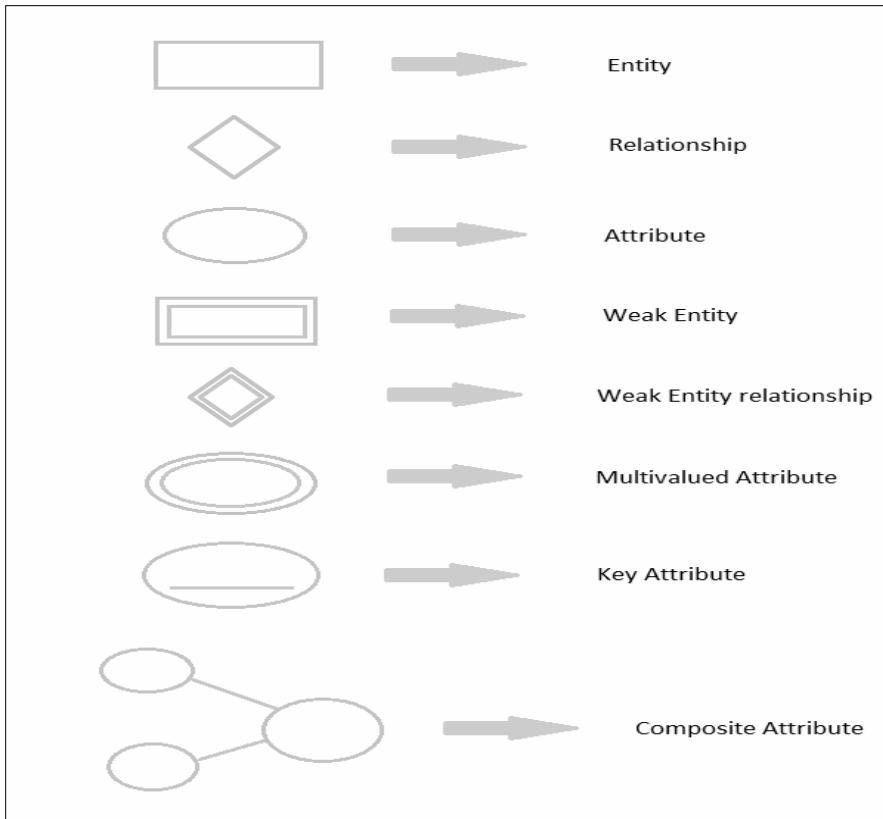


Figure 1.11: E-R Diagram

ER Model is represented by means of an ER diagram. Any object, for example, entities, attributes of an entity, relationship sets, and attributes of relationship sets, can be represented with the help of an ER diagram.

### **Entity:**

Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

EX. Student, Faculty, Course

- **Strong Entity:** - This type of entities has key attributes set that are used to create a primary key. Like student entity has Roll no as a key attribute.
- **Weak Entity:** - This type of Entities does not have any primary key attribute and they are dependent on some other entity which has the prime key attribute. For example, Employee child is a weak entity which is dependent on Employee Entity.

- **Composite Entity:** - This type of entities is used to replace many to many relationships and that relationship is replaced by Entity. It is of three type one-to-one cardinality, one-to-many cardinality, many-to-many cardinality.

### **Attributes:**

Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).

**Derived attributes:** - This is the attributes that can be derived from other attributes. Like Age can be derived from present date and Date of Birth.

**Composite Attribute:** - This is the attributes that can be created with the combination of two other attributes. Like a combination of first name and Last name will give the full name.

**Single-value attribute:** - This is the attributes which have single value when they are converted in the form of a table. Like DOB of a person.

**Multi-Valued Attribute:** - This is the attributes which have multiple values for the instance When an Entity is converted to a table and that attribute allows multiple values for it. Like a person have two contact numbers.

### **Relationship:**

A Relationship describes relations between **entities**. Relationship is represented using diamonds.

There are three types of relationship that exist between Entities.

- Binary Relationship
- Recursive Relationship
- Ternary Relationship

### **Binary Relationship**

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

- **One to One:** 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 relationship.

- **One to Many:** It reflects business rule that one entity is associated with many number of same entity.

The example for this relation might sound a little weird, but this means that one student can enroll to many courses, but one course will have one Student.

The arrows in the diagram describes that one student can enroll for only one course.

- **Many to One:** 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.

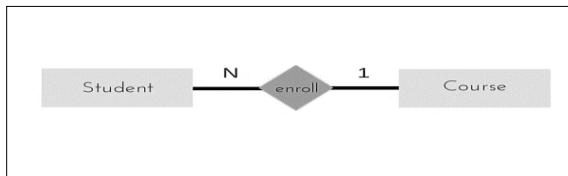


Figure 1.12: Many to One Relationship

- **Many to Many:**

The above diagram represents that many students can enroll for more than one courses.

### Recursive Relationship

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

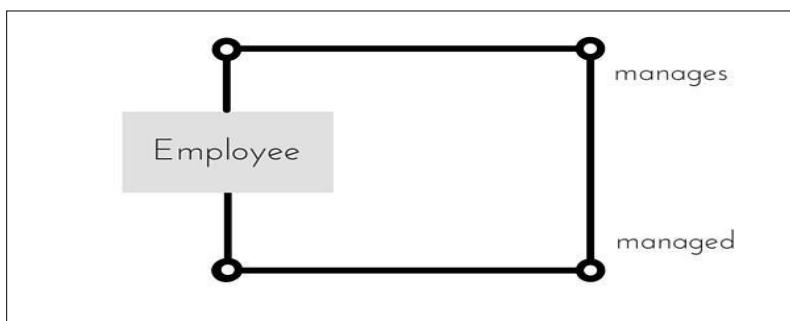


Figure 1.13: Recursive Relationship

### Ternary Relationship

Relationship of degree three is called Ternary relationship.

### Defining the E-R diagram of database:

Here we are going to design an Entity Relationship (ER) model for a college database.

we have the following statements.

- A college contains many departments
- Each department can offer any number of courses
- Many instructors can work in a department
- An instructor can work only in one department
- For each department, there is a Head
- An instructor can be head of only one department
- Each instructor can take any number of courses
- A course can be taken by only one instructor
- A student can enroll for any number of courses
- Each course can have any number of students

### **Step 1: Identify the Entities**

What are the entities here?

From the statements given, the entities are

Department

Course

Instructor

Student

### **Step 2: Identify the relationships**

- One department offers many courses. But one particular course can be offered by only one department. hence the cardinality between department and course is One to Many (1:N)
- One department has multiple instructors. But instructor belongs to only one department. Hence the cardinality between department and instructor is One to Many (1:N)
- One department has only one head and one head can be the head of only one department. Hence the cardinality is one to one. (1:1)
- One course can be enrolled by many students and one student can enroll for many courses. Hence the cardinality between course and student is Many to Many (M:N)
- One course is taught by only one instructor. But one instructor teaches many courses. Hence the cardinality between course and instructor is Many to One (N :1)

### Step 3: Identify the key attributes

- "Department\_Name" can identify a department uniquely. Hence Department Name is the key attribute for the Entity "Department".
- Course\_ID is the key attribute for "Course" Entity.
- Student\_ID is the key attribute for "Student" Entity.
- Instructor\_ID is the key attribute for "Instructor" Entity.

### Step 4: Identify other relevant attributes

- For the department entity, other attributes are location
- For course entity, other attributes are course\_name, duration
- For instructor entity, other attributes are first\_name, last name, phone
- For student entity, first\_name, last\_name, phone

### Step 5: Draw complete ER diagram

By connecting all these details, we can now draw ER diagram as given below.

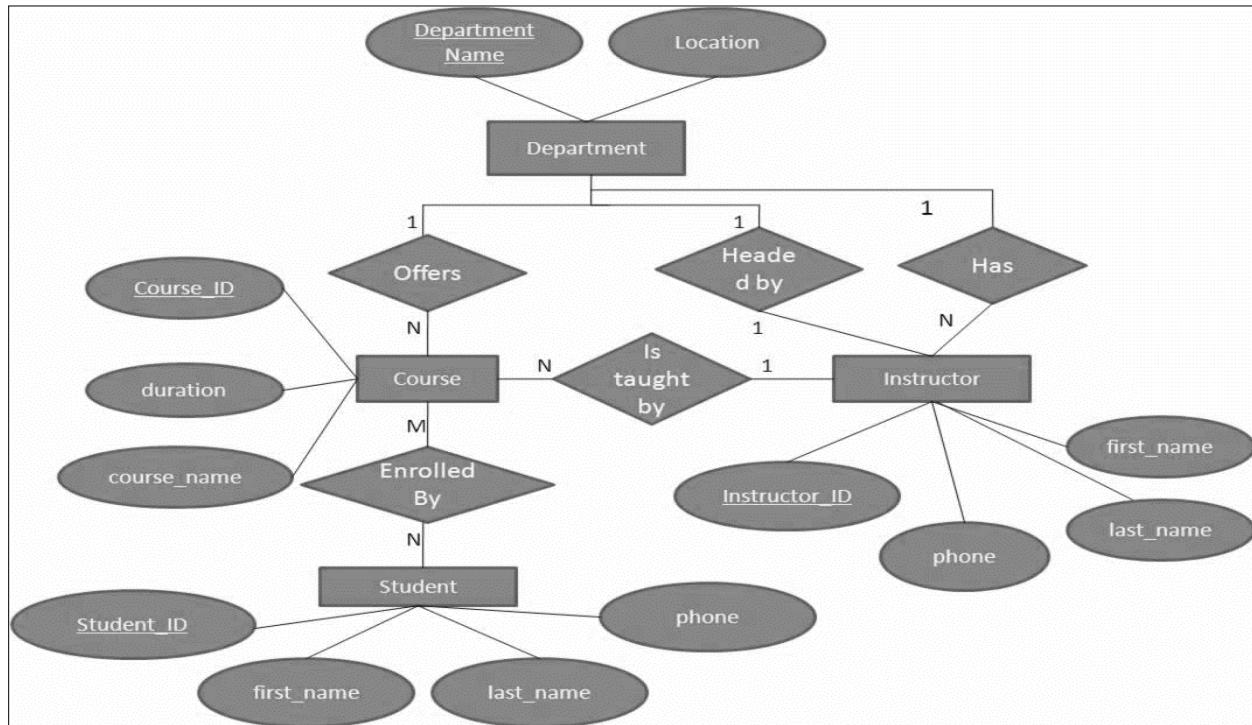


Figure 1.14: Defining E-R diagram of Database