UNIT-II
COMPUTER SOFTWARE

Computer Software: Application and System Software, Programming Languages and their Classification, Assemblers, Compilers and Interpreters, Process of Software Development, Operating Systems- Functions of Operating Systems, Types of Operating Systems (Batch Processing, Multitasking, Multiprogramming and Real time Systems) Database Management Systems Concepts, Types of Data Models.

SOFTWARE

Software: It is a set of computer programs that are required to enable the hardware components to work and perform their respective operations effectively. The software provides an interface between the hardware and user.

Computer Program: It is a set of logical instructions, written in computer programming language that tells the computer how to execute a particular task.

The software is divided into two types

1) System Software 2) Application Software

1) System Software: It consists of many different programs that manage and support different tasks. Depending upon the task performed, the system software can be classified into two major groups:

a) System management programs: Used for managing both the hardware and software systems.

System management program include:

- Operating system(Ex: MS DOS,WINDOWS,LINUX)
- Utility programs(Ex: Virus scanner)
- Device drivers(Ex: I/O device driver, printer driver)

b) System Development Programs: These are known as programming software allow the used to develop programs in different programming languages. Some of the system development programs include:

- Language Translators: translate the code form one language to another
- Linkers: Link the various modules and objects with the library files
- Debuggers: help in debugging the program(i.e. identifying the errors)
- Editors: Help to write the program code

Features of system software are as follows

- Close to system
- Fast in speed
- Difficult to design, Difficult to understand , Difficult to manipulate
- Less interactive
- Smaller in size
- Generally written in low-level language

OPERATING SYSTEM (OS): To make the hardware friendly to the user operating system is used. The collection of computer programs that control interaction of user and computer hardware is called Operating System. Most of the operating system are written using C language and has a lot of sub modules in it. OS manages allocation of computer resources. Usually part of the OS is stored permanently in a read-only memory (ROM) chip work as a starter of the operating system is called booting the computer.

The following is a list of some of the operating System's capabilities.

- Communicating with the computer user.
- Managing allocation of memory, of processor time, and of other resources for various task.
- Managing Input/output operations among the user programs and OS.
- Accessing data from secondary storage.
- Writing data to secondary storage.

Application Software

Application software products are designed to satisfy a particular need of a particular environment. All software applications prepared in the computer lab can come under the category of Application software. *This is further divided into two types:*

a. ***Packages*:** Collection related Applications are called "Packages".

b. ***Languages***: It is a systematical code for communication between two persons.

For example, a word-processing application MS Word, WordPerfect, or a spreadsheet application or Excel programs or a database management application such as Access or dBASE systems are well known application programs. User can also create their own application programs by using any programming language and solve a specific problem.

Examples of Application software are following

- Payroll Software
- Student Record Software
- Inventory Management Software
- Railways Reservation Software
- Microsoft Office Suite Software
- Microsoft Word, Microsoft Excel, Microsoft PowerPoint

Features of application software are as follows

- Close to user
- Easy to design, Easy to understand
- More interactive
- Slow in speed
- Generally written in high-level language
- Easy to manipulate and use
- Bigger in size and requires large storage space

COMPUTER LANGUAGES:

It is a systematical code for communication between System and user. This is in two categories.

1) Low Level Language: Machine dependable language is called "*Low level languages*". This is further divided into two types.

*a. **Machine language*:** Machine understandable language is called "Machine language". It is understandable by BINARY LANGUAGE. It is in the form of 0's and 1's.

Advantages:

i. Computer can understand directly.

Disadvantages:

i. It is very difficult to remember the codes and address of memory locations.
ii. User can't modify the program.
iii. User can't debug the program.
iv. It is machine dependent.
v. It is suitable for simple applications.

*b. **Assembly language**.* : It is a code language. It has some codes for performing specific operations. This is better than machine language.

E.g. ADD, SUB, MUL, DIV, ABS, FACT etc.

Advantages:

i. User can remember the mnemonics.
ii. It is easy to understand and develop the programs.
iii. User can modify the program and debug.
iv. It is suitable for simple applications.

Disadvantages:

i. It is machine dependent.
ii. It requires the translator program called Assembler
iii. Some codes and its operations are not easily identified by the user.

2) **High Level Language**: machine independent programming language that combines algebraic expressions and English symbols User understandable language is called "*High level language*". Because, it is in the form of English.

Advantages:

i. Easy to follow.
ii. Easy to understand
iii. Easy to modify and debug.
iv. Suitable for complex applications.

Disadvantages:

i. It requires the translator program called Compiler or Interpreter.
ii. It runs programs slower with compare to low level languages

To solve a given problem using computer, user prefer high-level programming languages.

PROGRAM CHARACTERISTICS

A program has the following characteristics. These characteristics apply to programs that are written in ***any*** programming language, not just C.

1. ***Integrity.*** This refers to the accuracy of the calculations. It should be clear that all other program enhancements will be meaningless if the calculations are not carried out correctly. Thus, the integrity of the calculations is an absolute necessity in any computer program.
2. ***Clarity*** refers to the overall readability of the program, with particular emphasis on its underlying logic. If a program is clearly written, it should be possible for another programmer to follow the program logic without undue effort. It should also be possible for the original author to follow his or her own program after being away from the program for an extended period of time. One of the objectives in the design of C is the development of clear, readable programs through an orderly and disciplined approach to programming.
3. ***Simplicity***. The clarity and accuracy of a program are usually enhanced by keeping things as simple as possible, consistent with the overall program objectives. In fact, it may be desirable to sacrifice a certain amount of computational efficiency in order to maintain a relatively simple, straightforward program structure.
4. ***Efficiency*** is concerned with execution speed and efficient memory utilization. These are generally important goals, though they should not be obtained at the expense of clarity or simplicity. Many complex programs require a tradeoff between these characteristics. In such situations, experience and common sense are key factors.
5. ***Modularity.*** Many programs can be broken down into a series of identifiable subtasks. It is good programming practice to implement each of these subtasks as a separate program module. In C, such modules are written as functions. The use of a modular programming structure enhances the accuracy and clarity of a program, and it facilitates future program alterations.
6. ***Generality.*** Usually we will want a program to be as general as possible, within reasonable limits. For example, we may design a program to read in the values of certain key parameters rather than placing fixed values into the program. As a rule, a considerable amount of generality can be obtained with very little additional programming effort.

There are many high-level programming languages are available these days.

Language	Application Area	Origin of Name
FORTRAN	Scientific	FORmula TRANslation
COBOL	Business data processing	**Co**mmon **B**usiness **O**riented **L**anguage
LISP	Artificial Intelligence	**Lis**t **P**rocessing
C	System programming	Predecessor language was named B
Prolog	Artificial Intelligence	**Log**ic **Pro**gramming
Ada	Real-Time distributed sys.	Ada Augusta Byron Collaborated..
Smalltalk	GUI	Objects “talk” to one another with message
C++	Support OOP	Incremental modification of C
Java	Supports web programming	originally named “oak”

Source file is written by a programmer for the solution of a specific problem. A **source** files containing the text of a high-level language program. The software developer creates this file by using an editor or word processor.

The high-level language before it is to be executed; it must first be translated into a computer's machine language. The program that does this translation is called **COMPILER.**
During compilation grammar rules are checked.(**syntax checking**) If the program is syntactically correct, compiler saves in an object file . Object file consist of machine language instructions. Using **linker** program object file is converted into machine language code which is an executable file and ready to run.

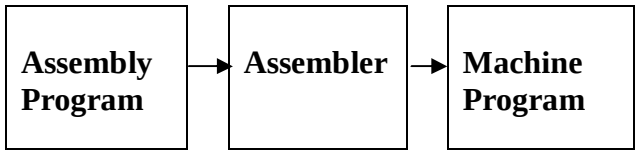
As the program executes it takes input data from one or more sources and sends results to output and /or secondary storage devices.

SYSTEM DEVELOPMENT TOOLS: The Successful development and execution of programs requires the usage of a number of tools. Some of these tools are:

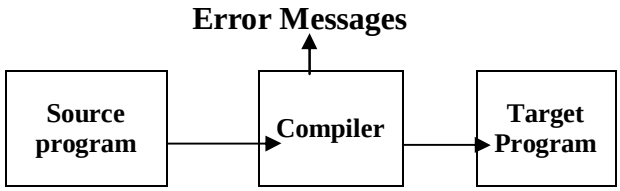
- Language translators
- Linkers
- Debuggers
- Editors

1) Language translators:

- **Assembler:** An assembler is a computer program that translates assembly language statements into machine language codes. The assembler takes each of the assembly language statements from the source code and generates a corresponding bit stream contains 1's and 0's.The output of the assembler in the form of sequence of 1's and 0's is called *Object code* or *machine code*. This machine code is finally executed by CPU to obtain results.

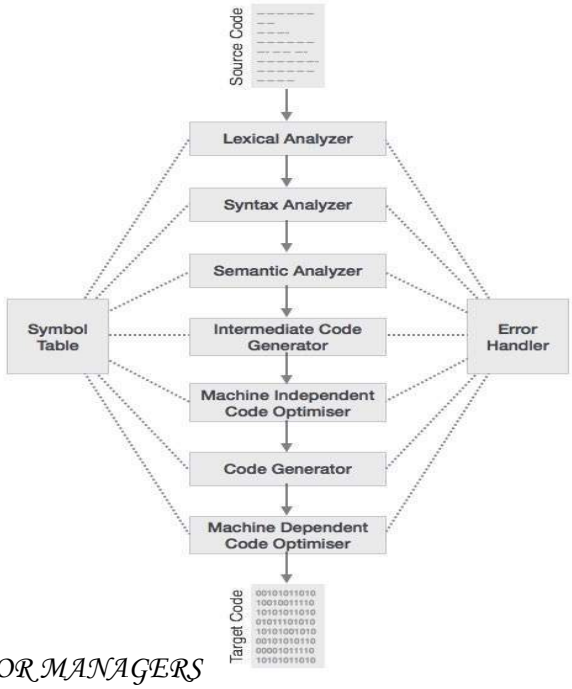


- **Compiler:** The compiler is a computer program
- that translates the source code written in a high-level language into *object code* of the machine level or low-level language. This translation process is called compilation. The entire high-level program is converted into the executable machine code file. A program that translates low-level language to a high-level one is a decompiler. Examples for compiled languages are: COBOL, FORTRAN, C, C++ etc.



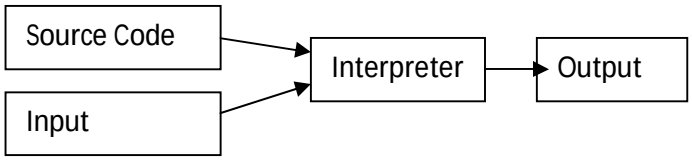
The first compiler was introduced by Grace Hopper in 1952 for A-0 programming language. In 1957, John Backus at IBM introduced the first complete compiler. Compilers are classified as two types: 1) single-pass compilers and ii) multi-pass compilers. Though Single pass compilers are generally faster than multi-pass compliers, for sophisticated optimization, multi-pass assemblers are required generate high-quality code.
The compiler treats the source program as a single unit, and executes the following steps:.

- Lexical analysis
- Syntactic analysis
- Semantic analysis
- Intermediate code Generation
- Code optimization
- Code generation



- **Interpreter:** The interpreter is a translation program that converts each high-level program statement into the corresponding machine code. This translation process carried out just before the program statement is executed. Instead of the entire program, one statement at a time is translated and executed immediately.

Ex: BASIC and PERL are interpreted languages.



Difference between compiler and interpreter:
1) Compiler translates the entire source code of high-level language program into machine code but interpreter translates only one statement of source code into machine-level language.
2) The compiled languages can be executed more efficiently and are faster than interpreter.

2) Linkers: Most of the High-level languages allow the developer to develop a large program containing multiple modules. Linker arranges or combines the object code of all the modules that have been generated by the language translator into a single executable program. The CPU of the computer is incapable of linking all the modules at the execution time and therefore, linker is combining all the modules into a single program. Linker also includes the links of various objects, which are defined in the runtime libraries.

3) Debugger: Debugger is the software that is used to detect the errors and bugs present in the programs. The Debugger locates the position of the errors in the program code with the help of Instructor Set Simulator (ISS) technique. ISS is capable of stopping the execution of a program at the point where an erroneous statement is encountered.

Debugger is divided into two types: i) Machine-level debugger and ii) symbolic debugger.

Machine-level debugger debugs the **object code** of the program and shows all the lines where bugs are detected.

The symbolic debugger debugs the **original code**. i.e., the high-level language code of the program. It shows the position of the bug in the original code of the program developed by the programmer.

The debugger performs a number of functions other than debugging, such as inserting breakpoints in the original code, tracking the value of specific variable, etc.In order to debug the program, a debugger helps perform the following tasks:

Step-by-step execution of a program

Back tracking for checking the previous steps

Stopping the execution of the program until the errors are corrected.

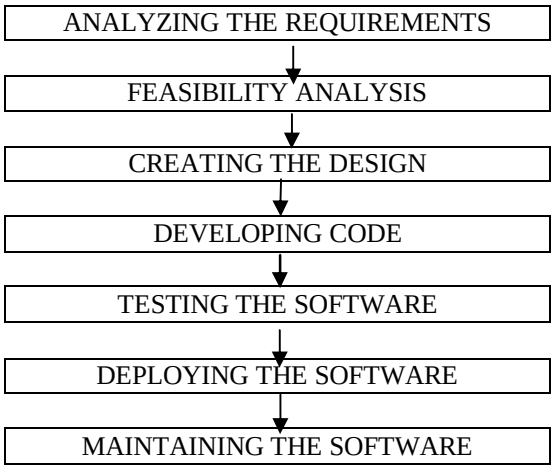
4) Editors: Editor is a special program the allows the user to work with text in a computer system. It is used for the documentation purpose and enables us to edit the information present in an existing document or a file. The editor enables us to perform various editing operations such as copy, cut and paste while editing the text. The editors are divided into following categories:

- Text editor: It is used to edit plain text.
- Digital audio editor: It is used to edit the information related to the audio components of a multimedia application.
- Graphics editor: It is used to edit the information related to graphical objects.
- Binary editor: It is used to edit the digital data or binary data (1's and 0's).
- HTML editor: It is used to edit the information included in the web pages.
- Source Code editor: It is used to edit the source code of a program written in a programming language such as C, C++ and java.

THE SOFTWARE DEVELOPMENT METHOD:

Programming is a problem solving activity. Programmers use the following software development methods sequence for developing efficient Software:

- Specifying the problem requirement (**requirements**)
- Analyze the problem (**analysis**)
- Designing an algorithm to solve the problem (**design)**
- Implement the algorithm (**Coding**)
- Test and verify the completed program(**Testing and validating**)
- Documentation and maintenance



1. ***Specify the problem requirements:*** forces you to state the problem clearly and unambiguously and to gain a clear understanding of what is required for its solution. Find more information from the person who posed the problem. If the requirements are not properly understood, then the software is bound to fall short of end user's expectation. There should be continuous interaction between the software development team and end users. The task of requirement analysis is typically performed by a business analyst.
2. ***Analyze the problem:*** analyzing the problem involves identifying the problem.
   a. inputs, that is the data you have to work with
   b. outputs, the desired results
   c. any additional requirements or constraints on the solution.

   At this stage, we should also determine the required format in which the results should be displayed and develop a list of problem variables and their relationships. These relationships may be expressed as formulas.

   Read the problem statement carefully, first, to obtain a clear idea of the problem and second, to determine the inputs and outputs. Otherwise it may lead to wrong problem solving.

   The process of modeling a problem by extracting the essential variables and their relationships is called abstraction.
3. ***Design the algorithm to solve the problem*:**

   Requires developing a list of steps called a algorithm to solve the problem and to then verify that the algorithm solves the problem as intended.

   Writing the algorithm is often the most difficult part of the problem solving process.

   Use top-down approach for writing the algorithm. In top-down design (also called divide and conquer) first list the major steps, or sub-problems, that need to be solved. Then solve the problem by solving its sub-problems.

   Most computer algorithms consists of at least the following sub-problems

   1. Get the data 2. Perform the computation 3. Display the results

Algorithm refinement: development of a detailed list of steps to solve a particular step in the original algorithm.

Desk checking: the step-by-step simulation of the computer execution of an algorithm.

4. Implementing the algorithm:

At this stage, one can begin to code the detailed program designs into program instructions of a given language. If all the previous steps have been completed with due diligence, this coding should be almost 'automatic'. The chances are high that a fairly successful program will result first time around. Although it may still contain bugs, these should be fewer and relatively easy to identify and correct.

5. Test and verify the completed program:

Testing and verifying the program requires testing the completed program to verify that it works as desired. Don't rely on just one test case. Run the program several times using different sets of data to make sure that it works correctly for every situation provided for in the algorithm.

There are two important activities that are performed while testing, they are verification and validation. Verification is a process of checking software based on some pre-defined specifications. Validation is testing the product to ascertain whether it meets user requirements.

As a result newly developed software is tested and installed successfully in target environment. Software documentation is handed over to users to make it operational.

6. Maintain and update the program:

Maintaining and updating the programs involves modifying a program to remove previously undetected errors and to keep it up to date as government regulations or company policies change.

A disciplined approach is essential to create programs that are easy to read, understand and maintain. This includes Documentation. Documentation is the process of collecting, organizing and maintaining, in written the complete information of the program for future references.

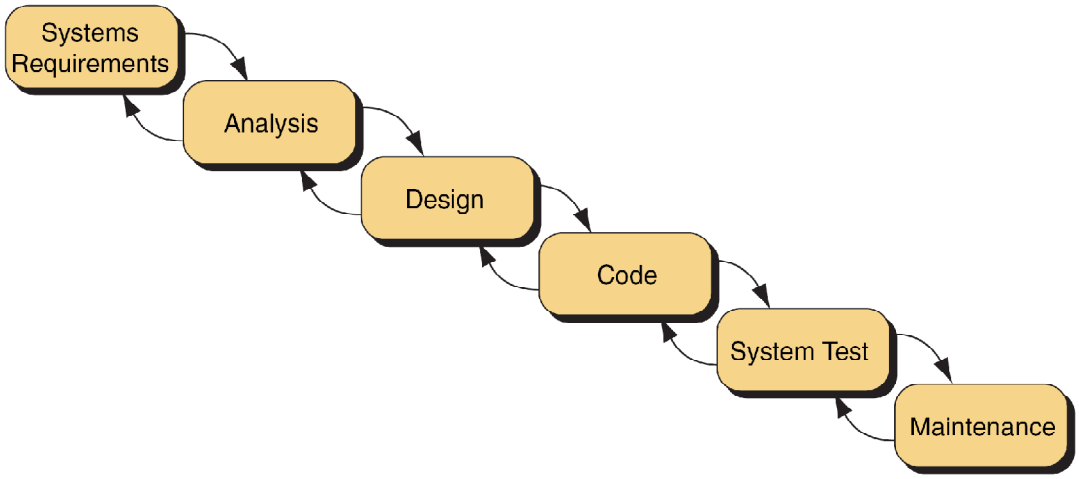


Fig: WATER FALL MODEL OF SOFTWEARE DEVELOPMENT

SOFTWARE ENGINEERING: is the establishment and use of sound engineering methods and principles to obtain software that is reliable and that works on real machines.

OPERATING SYSTEM:

An operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently.

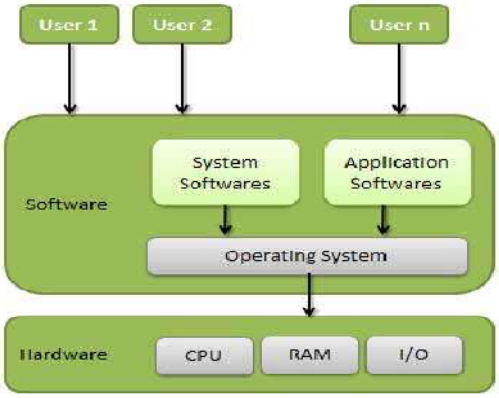
In technical terms, it is software which manages hardware. An operating System controls the allocation of resources and services such as memory, processors, devices and information.

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.

FUNCTIONS 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.

- ✓ Keeps tracks of primary memory i.e. what part of it are in use by whom, what part are not in use.
- ✓ In multiprogramming, OS decides which process will get memory when and how much.
- ✓ Allocates the memory when the process requests it to do so.
- ✓ De-allocates the memory when the process no longer needs it or has been terminated.

Processor Management

In multiprogramming environment, OS decides which process gets the processor when and how much time. This function is called process scheduling. Operating System does the following activities for processor management.

- ✓ Keeps tracks of processor and status of process. Program responsible for this task is known as traffic controller.
- ✓ Allocates the processor (CPU) to a process.
- ✓ De-allocates processor when processor is no longer required.

Device Management

OS manages device communication via their respective drivers. Operating System 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. 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 Activities

Following are some of the important activities that Operating System does.

- ✓ **Security:** By means of password and similar other techniques, preventing 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 and users**: Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

TYPES OF OPERATING SYSTEMS:

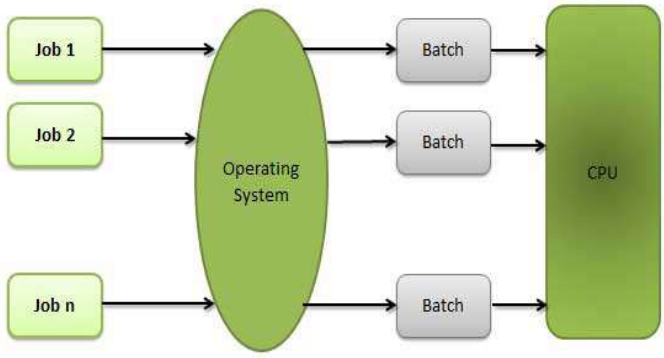
Operating systems are there from the very first computer generation. Operating systems keep evolving over the period of time. Following are few of the important types of operating system which are most commonly used.

- ➢ Batch processing
- ➢ Multitasking
- ➢ Multiprogramming
- ➢ Real Time System
- ➢ Distributed Environment

Batch processing:

Batch processing is a technique in which Operating System collects one programs and data together in a batch before processing starts. Operating system does the following activities related to batch processing.

- ✓ OS defines a job which has predefined sequence of commands, programs and data as a single unit.
- ✓ OS keeps a number of jobs in memory and executes them without any manual information.
- ✓ Jobs are processed in the order of submission i.e. first come first served fashion.
- ✓ When 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.



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 finished without any manual intervention.

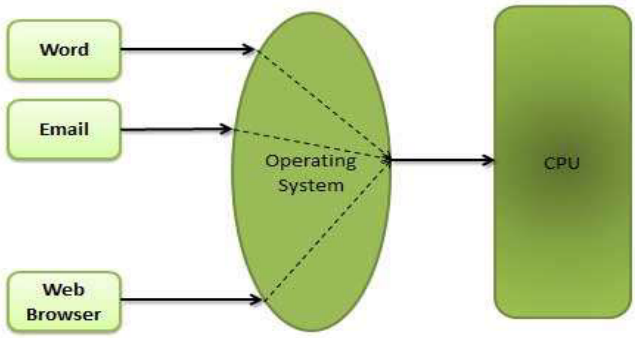
Disadvantages

- ➢ 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 refers to term where 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. Operating system 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.
- ✓ Operating System 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.



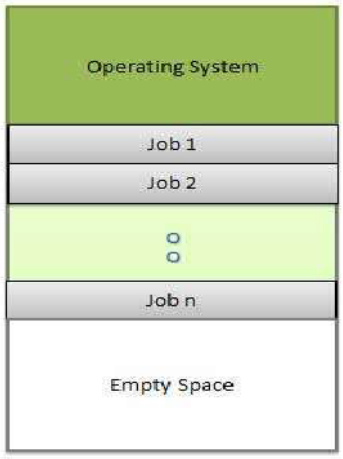
- ✓ A time-shared operating system uses concept of CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared CPU.
- ✓ Each user has at least one separate program in memory.

Multiprogramming

When two or more programs are residing in memory at the same time, then sharing the processor is referred to the multiprogramming. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.

Operating system 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 job in the memory.
- ✓ Multiprogramming operating system monitors the state of all active programs and system resources using memory management programs to ensures that the CPU is never idle unless there are no jobs



Following figure shows the memory layout for a multiprogramming system.

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.

Real Time System

Real time systems represents are usually dedicated embedded systems. 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

Distributed environment refers to multiple independent CPUs or processors in a computer system. Operating system does the following activities related to distributed environment.

- ➢ 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.
- ➢ OS manages the communications between the processors. They communicate with each other through various communication lines.

DATABASE MANAGEMENT SYSTEM CONCEPTS

DATA: Data is the raw material from which useful information is derived. Data is a collection of facts which is unorganized but can be made organized into useful information.

DATABASE: A database is a collection of information that is organized so that it can easily be accessed, managed, and updated.

DBMS: The database management system (DBMS), is a computer software program that is designed as the means of managing all databases that are currently installed on a system hard drive or network.

DBMS is a collection of interrelated data and a set of programs to access those data.

Data management is a discipline that focuses on the proper generation, storage and retrieval of data. Different types of database management systems exist, with some of them designed for the oversight and proper control of databases that are configured for specific purposes.

In database management system (DBMS), data files are the files that store the database information, whereas other files, such as index files and data dictionaries, store administrative information, known as metadata.

Some DBMS examples include MySQL, PostgreSQL, Microsoft Access, SQL Server, FileMaker, Oracle, RDBMS, dBASE, Clipper, and FoxPro. Since there are so many database management systems available, it is important for there to be a way for them to communicate with each other. For this reason, most database software comes with an Open Database Connectivity (ODBC) driver that allows the database to integrate with other databases.

DATABASE SYSTEM APPLICATIONS:

Database Management Systems:

1. A database management system (DBMS), or simply a database system (DBS), consists of
   - A collection of interrelated and persistent data (usually referred to as the database (DB)).
   - A set of application programs used to access, update and manage that data (which form the data management system (MS)).
2. The goal of a DBMS is to provide an environment that is both convenient and efficient to use in
   - Retrieving information from the database.
   - Storing information into the database.
3. Databases are usually designed to manage large bodies of information. This involves
   - Definition of structures for information storage (data modeling).
   - Provision of mechanisms for the manipulation of information (file and systems structure, query processing).
   - Providing for the safety of information in the database (crash recovery and security).
   - Concurrency control if the system is shared by users

Databases are widely used. Here are some representative applications.

- **Banking:** For customer information, accounts, loans, and banking transactions.
- **Airlines:** For reservations and schedule information.
- **Universities:** For students information, course registration and grades
- **Credit card transactions**: For purchases on credit card and generation of monthly statements.
- **Telecommunication:** For keeping records of calls made, generating monthly bills maintaining balances n prepaid calling cards and storing the information about the communication networks.
- **Finance:** For storing information about sales, and purchases of financial instruments such as stocks and bonds;
- **Sales:** For customer, product and purchase information.
- **On-line retailers:** For sales data noted above plus on-line order tacking;
- **Manufacturing:** For management of supply chain and tracking production of items in factories, inventories of items in warehouses and stores and order for items.
- **Human resources:** For information about employees, salaries, payroll taxes, and for generation of pay checks.

Characteristics of Data in a database:

- *SHARED*-Data in a database are shared among different user's and applications
- *PERSISTENCE*-Data in a database exists permanently in the sense that datacan live beyond the scope of the process that created it.
- *CORRECTNESS*-Data should be correct
- *SECURITY*-Data should be protected from un-authorized access.
- *CONSISTENCY*-When ever more than one data element in a database represents real-world values, the values should be protected from unauthorized access
- *NON*-REDUNDANCY-No two data items in a database should represent the same real world entity.

ROLE AND ADVANTAGE OF DBMS:

The DBMS serves as the intermediary between the user and the database. Having a DBMS between the end user's application and the database offers some important advantages. First, the DBMS enables the data in the database *to be shared* among multiple applications or users. Second, the DBMS *integrates* the many different users views of the data into a single allen compassing data repository.

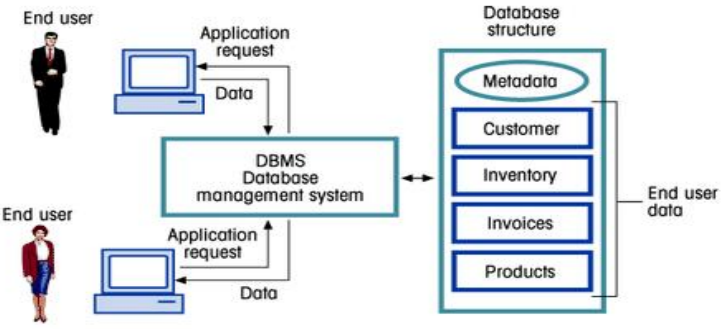


FIGURE 1.2 THE DBMS MANAGES THE INTERACTION BETWEEN THE END USER AND THE DATABASE

DBMS provides advantages such as:

- ***Improved data sharing****:* The DBMS helps creates an environment in which end users have better access to more data and better-managed data
- ***Improved data security****:* The more the user access the data, the greater the risks of data security; A DBMS provides a framework for better enforcement of data privacy and security policies.
- ***Better data integration****:* Wider access to well-managed data promotes an integrated view of the organization's operation and a clearer view of the big picture.
- ***Minimized data inconsistency****:* Data inconsistency exists when different versions of the same data appear in different places. The probability of data inconsistency is greatly reduced in a properly designed database.
- ***Improved data access****:* The DBMS makes it possible to produce quick answers to ad hoc queries
- ***Improved decision making****:* Better-managed data and improved data access make it possible to generate better quality information, on which better decisions are based.
- ***Increased end*-user productivity**: The availability of data, combined with the tools that transform data into usable information, empowers end users to make quick, informed decisions that can make the difference between success and failure in the global economy.
- ***Data administration****:* When several users share the data, centralizing the administration of data can offer significant improvements.
- ***Concurrent access and crash recovery****:* A DBMS schedules concurrent access to the data in such a manner that users can think of the data being accessed by only one user at a time. Further, DBMS protects users from the effects of system failures.
- ***Reduced application development time****:* Clearly, the DBMS supports important functions that are common to many applications accessing data in the DBMS. This, in conjunction with the high-level interface to the data, facilitates quick application development.

DBMS FUNCTIONS

- **Data Dictionary Management**-DBMS stores definitions of the data elements and their relations(metadata) in a data dictionary
- **Data Storage Management**-The DBMS creates and manages the complex structures required for data storage, thus relieving you from the difficult task of defining and programming the physical data characteristics. Data storage management is also important for database performance tuning. Performance tuning relates to the activities that make the database perform more efficiently in terms of storage and access speed.
- **Data Transformation** and Presentation- the DBMS transforms entered data to conform to required data structures. Regardless of the data presentation format, the DBMS must manage the data in the proper format
- **Security Management**- DBMS creates a security system that enforces user security and data privacy. Security rules determine which users can access the database, which data items each user can access and which data operations the user can perform. This is especially important in multi-user database systems.
- **Multi-User Access Control-** DBMS provide transaction management and concurrency control
- **Backup and Recovery Management**- Recovery management is provided by DBMS to recover the data when failures occurs
- **Data Integrity Management**-DBMS enforces integrity rules to minimize data redundancy and maximizing data consistency
- **Database Access Languages (DDL and DML) and Application Programming Interfaces-** DBMS provides data accessing language –Query language(procedural and nonprocedural)
- **Database Communication Interfaces-** Current generation DBMSs accept end-user requests via multiple, different network environments.

Disadvantages

Although the database system yields considerable advantages over previous data management approaches, database systems do carry significant disadvantages.

- Increased costs
- Managing complexity
- Maintaining currency
- Vendor dependence
- Frequent upgrades/Replacement cycles.

DATABASE SYSTEM ARCHITECTURE

A database system is divided into modules that deal with each of the responsibilities of the overall system. The functional components of database system can be broadly divided into the *STORAGE MANAGER AND QUERY PROCESSOR* components.

Storage manager: Storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system. The storage manager component includes:

- **Authorization and integrity manager-** checks integrity constraints and authority of users to access data
- **Transaction manager,** ensures consistency when system failures occurs and no conflicting with concurrent executions
- **File manager,** manages the allocation of space on the disk storage
- **Buffer manager,** responsible for fetching the data from disk storage into main memory, and deciding what data to cache in main memory.

The storage manager implements several data structures as part of the physical system implementation: Data files, Data dictionary, Indices.

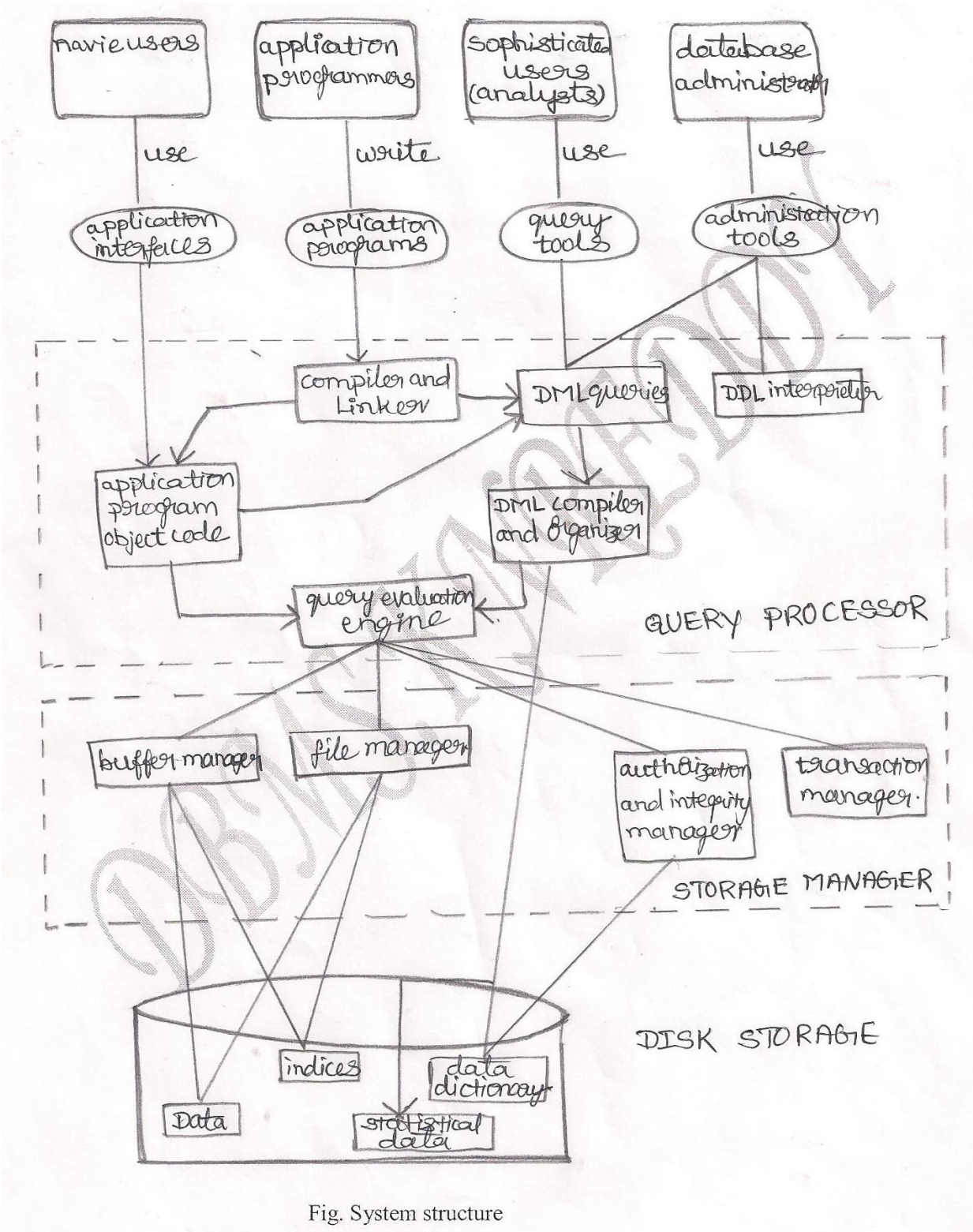


Fig. System structure

Query processor:

The **Query processor** component includes:

- **DDL interpreter,** which interprets DDL statements and records the definitions in the data dictionary
- **DML compiler,** which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands. The DML compiler also performs **query optimization.**
- **Query evaluation engine,** which executes low-level instructions generated by the DML compiler.

Transaction Management:

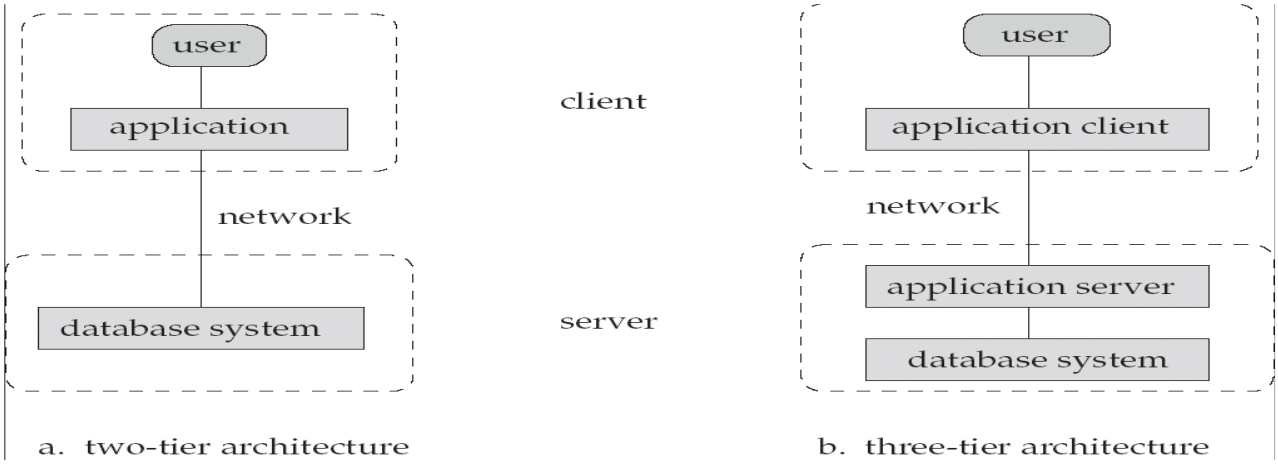
- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database

The architecture of a database system is greatly influenced by the underlying computer system on which the database system runs

Database system can be centralized or client-server, where one server machine executes work on behalf of multiple client machines.

Most users of a database system today are not present at the site of the database system, but connect to it through a network. We can therefore differentiate between client machines, on which remote database user work, and server machines, on which the database system runs.

Database applications are usually partitioned into two or three parts



In two-tier architecture, the application is partitioned into a component that resides at the client machine, which invokes database system functionality at the server machine through query language statements. Application program interface standards like ODBC and JDBC are used for interaction between the client and the server

In contrast, in three-tier architecture, the client machine acts as merely a front end and does not contain any direct database calls. Instead, the client end communicates with an application server, usually through a forms interface. The application server in turn communicates with a database system to access data. The business logic of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients. Three-tier applications are more appropriate for large applications, and for applications that run on the World Wide Web.

DBMS DATA MODELS

Hierarchical Data Model

In the *hierarchical data model*, information is organized as a collection of inverted trees of records. The inverted trees may be of arbitrary depth. The record at the root of a tree has zero or more child records; the child records, in turn, serve as parent records for their immediate descendants. This parent-child relationship recursively continues down the tree.

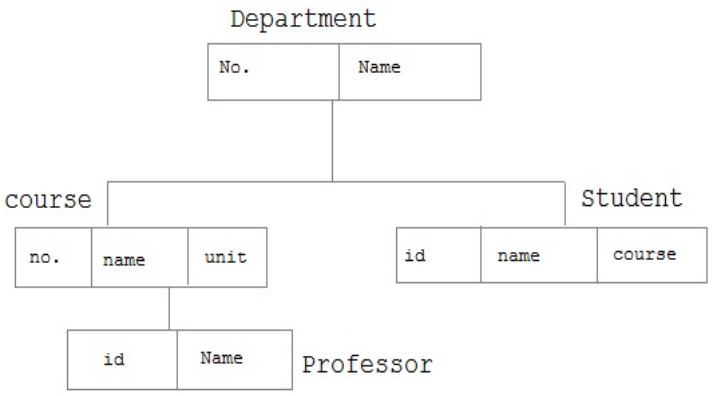
Applications can navigate a hierarchical database by starting at a root and successively navigate downward from parent to children until the desired record is found.

Advantages

–Conceptual simplicity
–Database security
–Data independence
–Database integrity
–Efficiency

Disadvantages

–Complex implementation
–Difficult to manage
–Lacks structural independence
–Complex applications programming and use
–Implementation limitations
–Lack of standards



Network Data Model

In the *network data model*, information is organized as a collection of graphs of record that are related with pointers. Network data models represent data in a symmetric manner, unlike the hierarchical data model (distinction between a parent and a child). A network data model is more flexible than a hierarchical data model and still permits efficient navigation.

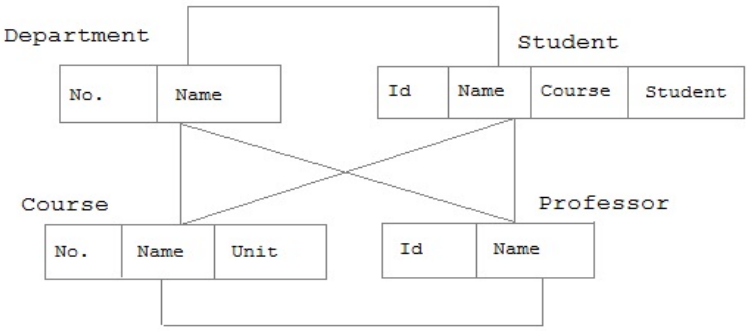
The records consists of lists of fields (fixed or variable length with maximum length), where each field contains a simple value (fixed or variable size). Some network DBMS require that the values not be null. The network data model also introduces the notion of indexes of fields and records, sets of pointers, and physical placement of records.

Advantages

–Conceptual simplicity
–Handles more relationship types
–Data access flexibility
–Promotes database integrity
–Data independence
–Conformance to standards

Disadvantages

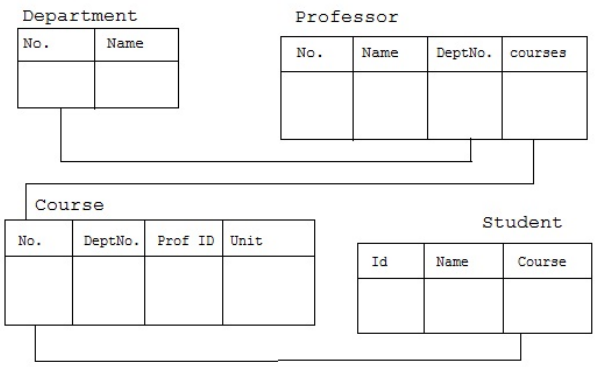
–System complexity
–Lack of structural independence



Relational Data Model

In the *relational data model*, information is organized in relations (two-dimensional tables). Each relation contain a set of tuples (records). Each tuple contain a number of fields. A field may contain a simple value (fixed or variable size) from some domain (e.g. integer, real, text, etc.).

The relational data model is based on a mathematical foundation, called *relational algebra*.



Relational modeling focuses on the information in the system. Not on the behavior. The modeling dimension consists of translations between the human knowledge and the database model.

Advantages

–Structural independence
–Improved conceptual simplicity
–Easier database design, implementation, management, and use
–Ad hoc query capability
–Powerful database management system

Disadvantages

–Substantial hardware and system software overhead
–Can facilitate poor design and implementation
–May promote "islands of information" problems

TERMS RELATED TO RELATION MODEL:

RELATION

A relation is a truth predicate. It defines what attributes are involved in the predicate and what the meaning of the predicate is.

ATTRIBUTE

An attribute identifies a name that participates in the relation and specifies the domain from which values of the attribute must come.

DOMAIN

A domain is simply a data type. It specifies a data abstraction: the possible values for the data and the operations available on the data.

TUPLE

A tuple is a truth statement in the context of a relation. A tuple has attribute values which match the required attributes in the relation and that state the condition that is known to be true.

ATTRIBUTE VALUE

An attribute value is the value for an attribute in a particular tuple. An attribute value must come from the domain that the attribute specifies.

RELATION KEYS

An important element of the relational model is the notion of keys: candidate keys, primary keys, and foreign keys. A *candidate key* consists of one or more fields whose values uniquely identifies the tuples within the record. A *primary key* is an (possibly arbitrarily) chosen candidate key which is preferred used to reference instances. Normally, a relation has only one primary key. A *foreign key* is a reference to a candidate key (usually the primary key). A foreign key is used within a relation to refer from one tuple in that relation to some tuple in another relation.

Object-oriented Data Model

In the *object-oriented data model*, information is organized in graphs of objects, where each object has a number of attributes. Attributes can be simple values, complex values (part objects), references to other objects, or methods. Objects are instances of classes, and classes are (possibly) related to each by means of inheritance. The inheritance mechanism supports generalization and specialization and offers many aspects of structured reuse of models. Inheritance also offers the mechanism for qualified polymorphism, since the resulting type system can allow for objects to be recognized as belonging to several different types, namely the types of all the classes in the inheritance hierarchy which lies on the path from the instantiating class to the root of the hierarch

Advantages

–Adds semantic content
–Visual presentation includes semantic content
–Database integrity
–Both structural and data independence

Disadvantages

–Complex navigational data access

–High system overhead slows transactions
–Lack of market penetration

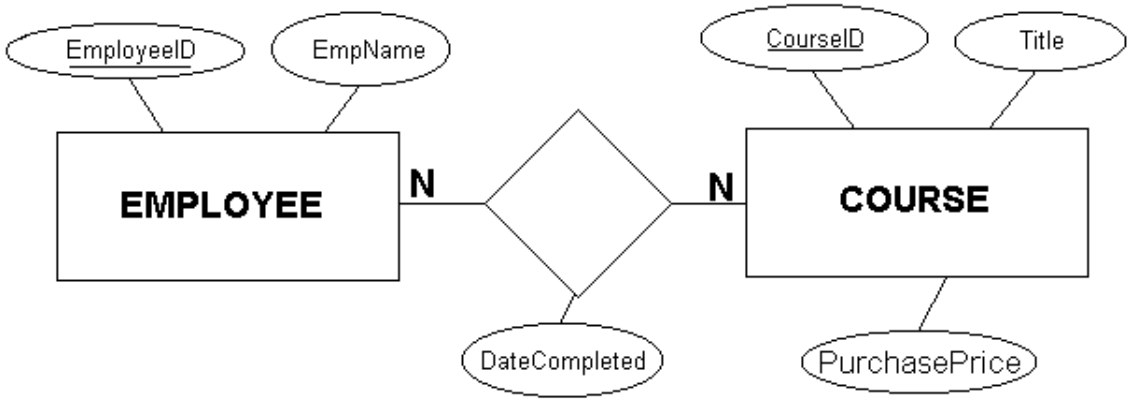
The Entity Relationship Model

Widely accepted and adapted graphical tool for data modeling. Introduced by Chen in 1976. Graphical representation of entities and their relationships in a database structure. Entity relationship diagram (ERD) Uses graphic representations to model database components. Entity is mapped to a relational table.

Entity instance (or occurrence) is row in table

Entity set is collection of like entities

Connectivity labels types of relationships. Diamond connected to related entities through a relationship line.



Advantages

–Exceptional conceptual simplicity
–Visual representation
–Effective communication tool
–Integrated with the relational data model

Disadvantages

–Limited constraint representation
–Limited relationship representation
–No data manipulation language
–Loss of information content

Entity: Real-world object distinguishable from other objects. An entity is described (in DB) using a set of *attributes*.

Entity Set: A collection of similar entities. E.g., all employees.

Relationship: Association among two or more entities.

Relationship Set: Collection of similar relationships.