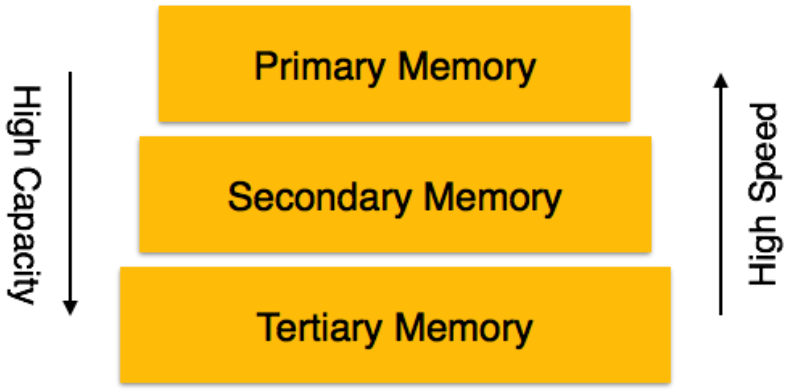
M.Sc. (Computer Science)
UNIT – V
DATE: 26-03-2020

Storage System

Databases are stored in file formats, which contain records. At physical level, the actual data is stored in electromagnetic format on some device. These storage devices can be broadly categorized into three types −



- **Primary Storage** − The memory storage that is directly accessible to the CPU comes under this category. CPU's internal memory (registers), fast memory (cache), and main memory (RAM) are directly accessible to the CPU, as they are all placed on the motherboard or CPU chipset. This storage is typically very small, ultra-fast, and volatile. Primary storage requires continuous power supply in order to maintain its state. In case of a power failure, all its data is lost.

- **Secondary Storage** − Secondary storage devices are used to store data for future use or as backup. Secondary storage includes memory devices that are not a part of the CPU chipset or motherboard, for example, magnetic disks, optical disks (DVD, CD, etc.), hard disks, flash drives, and magnetic tapes.

- **Tertiary Storage** − Tertiary storage is used to store huge volumes of data. Since such storage devices are external to the computer system, they are the slowest in speed. These storage devices are mostly used to take the back up of an entire system. Optical disks and magnetic tapes are widely used as tertiary storage.

Memory Hierarchy

A computer system has a well-defined hierarchy of memory. A CPU has direct access to it main memory as well as its inbuilt registers. The access time of the main memory is obviously less than the CPU speed. To minimize this speed mismatch, cache memory is introduced. Cache memory provides the fastest access time and it contains data that is most frequently accessed by the CPU.

The memory with the fastest access is the costliest one. Larger storage devices offer slow speed and they are less expensive, however they can store huge volumes of data as compared to CPU registers or cache memory.

Magnetic Disks

Hard disk drives are the most common secondary storage devices in present computer systems. These are called magnetic disks because they use the concept of magnetization to store information. Hard disks consist of metal disks coated with magnetizable material. These disks are placed vertically on a spindle. A read/write head moves in between the disks and is used to magnetize or de-magnetize the spot under it. A magnetized spot can be recognized as 0 (zero) or 1 (one).

Hard disks are formatted in a well-defined order to store data efficiently. A hard disk plate has many concentric circles on it, called **tracks**. Every track is further divided into **sectors**. A sector on a hard disk typically stores 512 bytes of data.

Redundant Array of Independent Disks

RAID or **R**edundant **A**rray of **I**ndependent **D**isks, is a technology to connect multiple secondary storage devices and use them as a single storage media.

RAID consists of an array of disks in which multiple disks are connected together to achieve different goals. RAID levels define the use of disk arrays.

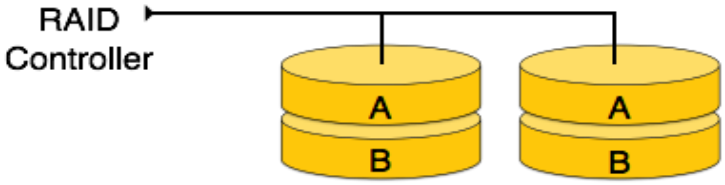
RAID 0

In this level, a striped array of disks is implemented. The data is broken down into blocks and the blocks are distributed among disks. Each disk receives a block of data to write/read in

parallel. It enhances the speed and performance of the storage device. There is no parity and backup in Level 0.

RAID 1

RAID 1 uses mirroring techniques. When data is sent to a RAID controller, it sends a copy of data to all the disks in the array. RAID level 1 is also called **mirroring** and provides 100% redundancy in case of a failure.

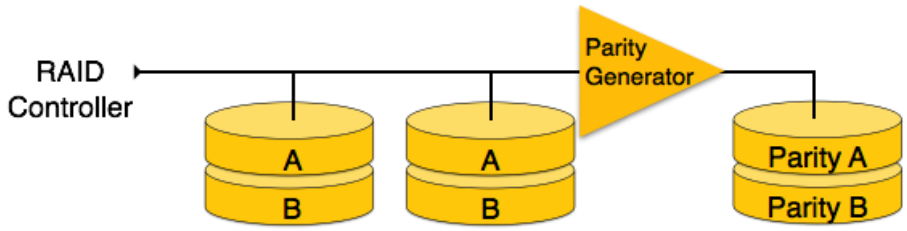


RAID 2

RAID 2 records Error Correction Code using Hamming distance for its data, striped on different disks. Like level 0, each data bit in a word is recorded on a separate disk and ECC codes of the data words are stored on a different set disks. Due to its complex structure and high cost, RAID 2 is not commercially available.

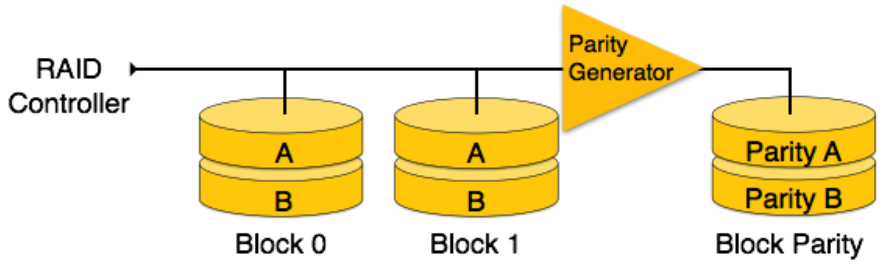
RAID 3

RAID 3 stripes the data onto multiple disks. The parity bit generated for data word is stored on a different disk. This technique makes it to overcome single disk failures.



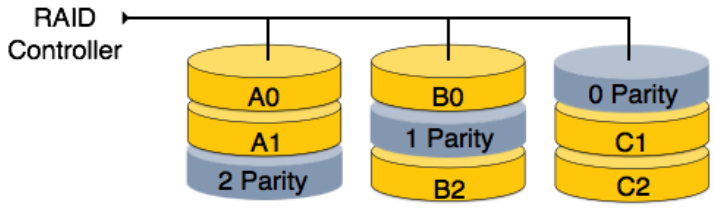
RAID 4

In this level, an entire block of data is written onto data disks and then the parity is generated and stored on a different disk. Note that level 3 uses byte-level striping, whereas level 4 uses block-level striping. Both level 3 and level 4 require at least three disks to implement RAID.



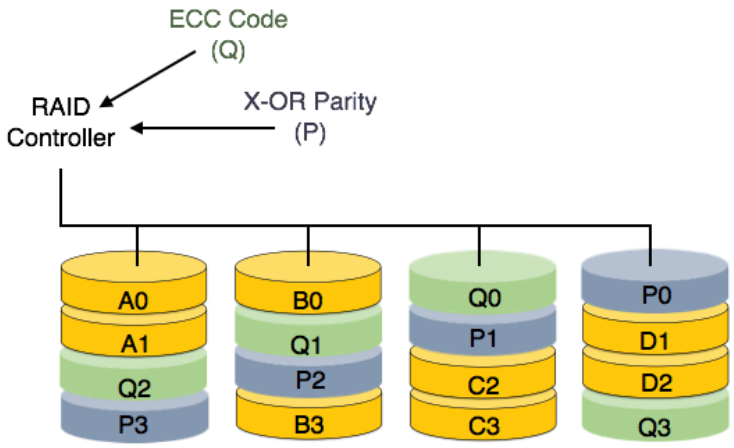
RAID 5

RAID 5 writes whole data blocks onto different disks, but the parity bits generated for data block stripe are distributed among all the data disks rather than storing them on a different dedicated disk.



RAID 6

RAID 6 is an extension of level 5. In this level, two independent parities are generated and stored in distributed fashion among multiple disks. Two parities provide additional fault tolerance. This level requires at least four disk drives to implement RAID.

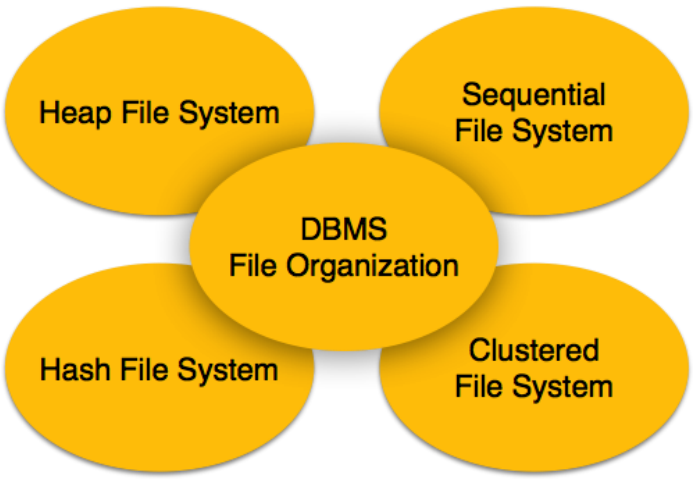


File Structure

Relative data and information is stored collectively in file formats. A file is a sequence of records stored in binary format. A disk drive is formatted into several blocks that can store records. File records are mapped onto those disk blocks.

File Organization

File Organization defines how file records are mapped onto disk blocks. We have four types of File Organization to organize file records −



Heap File Organization

When a file is created using Heap File Organization, the Operating System allocates memory area to that file without any further accounting details. File records can be placed anywhere in

that memory area. It is the responsibility of the software to manage the records. Heap File does not support any ordering, sequencing, or indexing on its own.

Sequential File Organization

Every file record contains a data field (attribute) to uniquely identify that record. In sequential file organization, records are placed in the file in some sequential order based on the unique key field or search key. Practically, it is not possible to store all the records sequentially in physical form.

Hash File Organization

Hash File Organization uses Hash function computation on some fields of the records. The output of the hash function determines the location of disk block where the records are to be placed.

Clustered File Organization

Clustered file organization is not considered good for large databases. In this mechanism, related records from one or more relations are kept in the same disk block, that is, the ordering of records is not based on primary key or search key.

File Operations

Operations on database files can be broadly classified into two categories −

- **Update Operations**
- **Retrieval Operations**

Update operations change the data values by insertion, deletion, or update. Retrieval operations, on the other hand, do not alter the data but retrieve them after optional conditional filtering. In both types of operations, selection plays a significant role. Other than creation and deletion of a file, there could be several operations, which can be done on files.

- **Open** − A file can be opened in one of the two modes, **read mode** or **write mode**. In read mode, the operating system does not allow anyone to alter data. In other words, data is read only. Files opened in read mode can be shared among several entities. Write

mode allows data modification. Files opened in write mode can be read but cannot be shared.

- **Locate** − Every file has a file pointer, which tells the current position where the data is to be read or written. This pointer can be adjusted accordingly. Using find (seek) operation, it can be moved forward or backward.

- **Read** − By default, when files are opened in read mode, the file pointer points to the beginning of the file. There are options where the user can tell the operating system where to locate the file pointer at the time of opening a file. The very next data to the file pointer is read.

- **Write** − User can select to open a file in write mode, which enables them to edit its contents. It can be deletion, insertion, or modification. The file pointer can be located at the time of opening or can be dynamically changed if the operating system allows to do so.

- **Close** − This is the most important operation from the operating system's point of view. When a request to close a file is generated, the operating system

    - removes all the locks (if in shared mode),
    - saves the data (if altered) to the secondary storage media, and
    - releases all the buffers and file handlers associated with the file.

The organization of data inside a file plays a major role here. The process to locate the file pointer to a desired record inside a file various based on whether the records are arranged sequentially or clustered.

Database Model

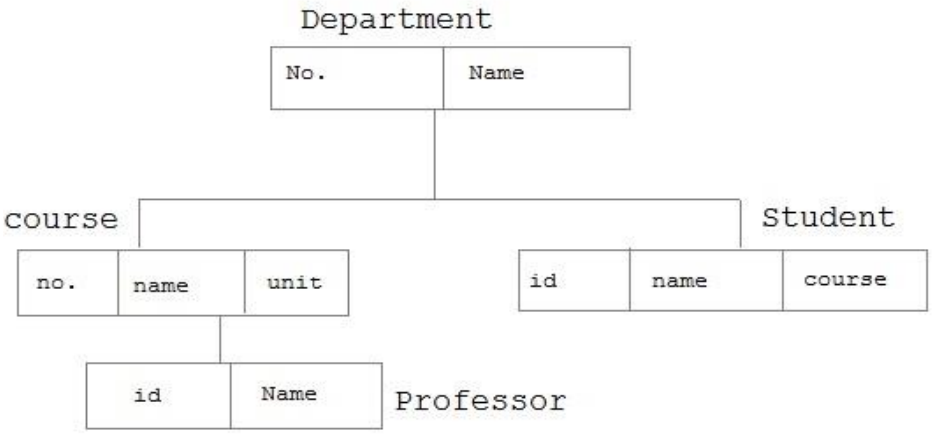
A Database model defines the logical design of data. The model describes the relationships between different parts of the data. Historically, in database design, three models are commonly used. They are,

- Hierarchical Model

- Network Model
- Relational Model

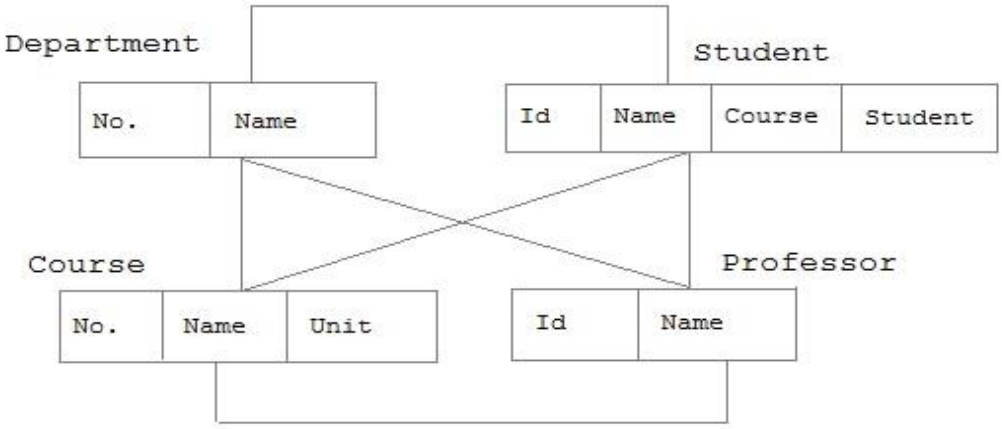
Hierarchical Model

In this model each entity has only one parent but can have several children . At the top of hierarchy there is only one entity which is called **Root**.



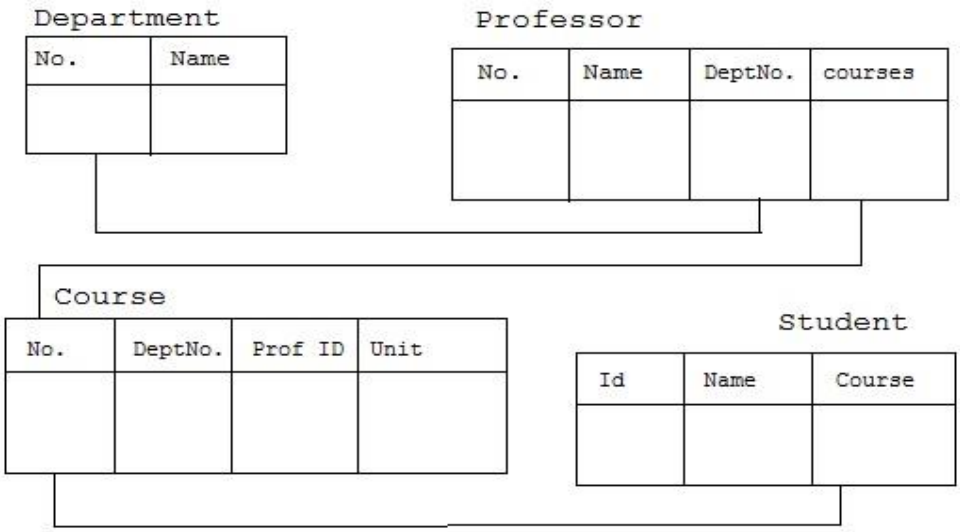
Network Model

In the network model, entities are organised in a graph,in which some entities can be accessed through several path



Relational Model

In this model, data is organised in two-dimesional tables called **relations**. The tables or relation are related to each other.

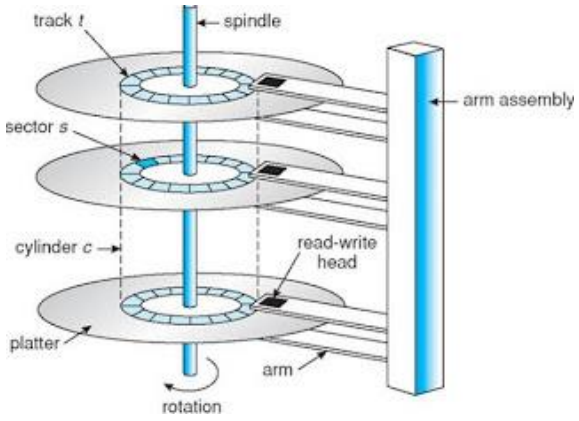


MAGNETIC DISKS

Magnetic Disk are used for storing large amounts of data. The storage capacity of a single disk ranges from 10 MB to 20 GB. Magnetic disks support direct access to a desired location and are widely used for the database applications. A DBMS provides access to data on disk ; applications of DBMS should not worry about, whether data is in main memory are disk.

The most basic unit of data on the disk is a single bit of information. By magnetizing an area on disk in certain ways, one can make a bit value to represent either in 1 or 0. To code information, bits are grouped into bytes (bytes or characters). Byte sizes are equal to 4 to 8 bytes, depending on the computer and the device. We assume that one character is stored in a single byte, and we used the terms byte and character interchangeably. The capacity of a disk is the number of bytes it can store, which is very large. small floppy disk used with micro computers holds from 400 KB to 1.5 MB; hard disks for micro - typically hold from several hundreds MB to few GB: and large disk packs used with minicomputers and mainframes have capacities that range up to a few tens or hundred of GB. Disk capacities continue to grow as technology

improves. Data is stored on disk in units called **disk blocks.** A disk block is a contiguous sequence of bytes and is the unit in which data is written to a disk and read from a disk.



Blocks are arranged in concentric rings called tracks, on one or more platters. Tracks can be recorded on one or both surfaces of platter.

We refer to platters as single - sided or double - sided accordingly.

The set of all tracks with the same diameter is called a cylinder, because the space occupied by these tracks is shaped like a cylinder; a cylinder contains one track per platter surface.

File Organization

In a database we have lots of data. Each data is grouped into related groups called tables. Each table will have lots of related records. Any user will see these records in the form of tables in the screen. But these records are stored as files in the memory. Usually one file will contain all the records of a table.

As we saw above, in order to access the contents of the files – records in the physical memory, it is not that easy. They are not stored as tables there and our SQL queries will not work. We need some accessing methods. To access these files, we need to store them in certain order so that it will be easy to fetch the records. It is same as indexes in the books, or catalogues in the library, which helps us to find required topics or books respectively.

Storing the files in certain order is called file organization. The main objective of file organization is

- Optimal selection of records i.e.; records should be accessed as fast as possible.
- Any insert, update or delete transaction on records should be easy, quick and should not harm other records.
- No duplicate records should be induced as a result of insert, update or delete
- Records should be stored efficiently so that cost of storage is minimal.

There are various methods of file organizations. These methods may be efficient for certain types of access/selection meanwhile it will turn inefficient for other selections. Hence it is up to the programmer to decide the best suited file organization method depending on his requirement.

Some of the file organizations are

1. Sequential File Organization
2. Heap File Organization
3. Hash/Direct File Organization
4. Indexed Sequential Access Method
5. B+ Tree File Organization
6. Cluster File Organization

Indexing

We know that data is stored in the form of records. Every record has a key field, which helps it to be recognized uniquely.

Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done. Indexing in database systems is similar to what we see in books.

Indexing is defined based on its indexing attributes. Indexing can be of the following types −

- **Primary Index** − Primary index is defined on an ordered data file. The data file is ordered on a **key field**. The key field is generally the primary key of the relation.
- **Secondary Index** − Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.
- **Clustering Index** − Clustering index is defined on an ordered data file. The data file is ordered on a non-key field.

Ordered Indexing is of two types −

- Dense Index
- Sparse Index

Dense Index

In dense index, there is an index record for every search key value in the database. This makes searching faster but requires more space to store index records itself. Index records contain search key value and a pointer to the actual record on the disk.

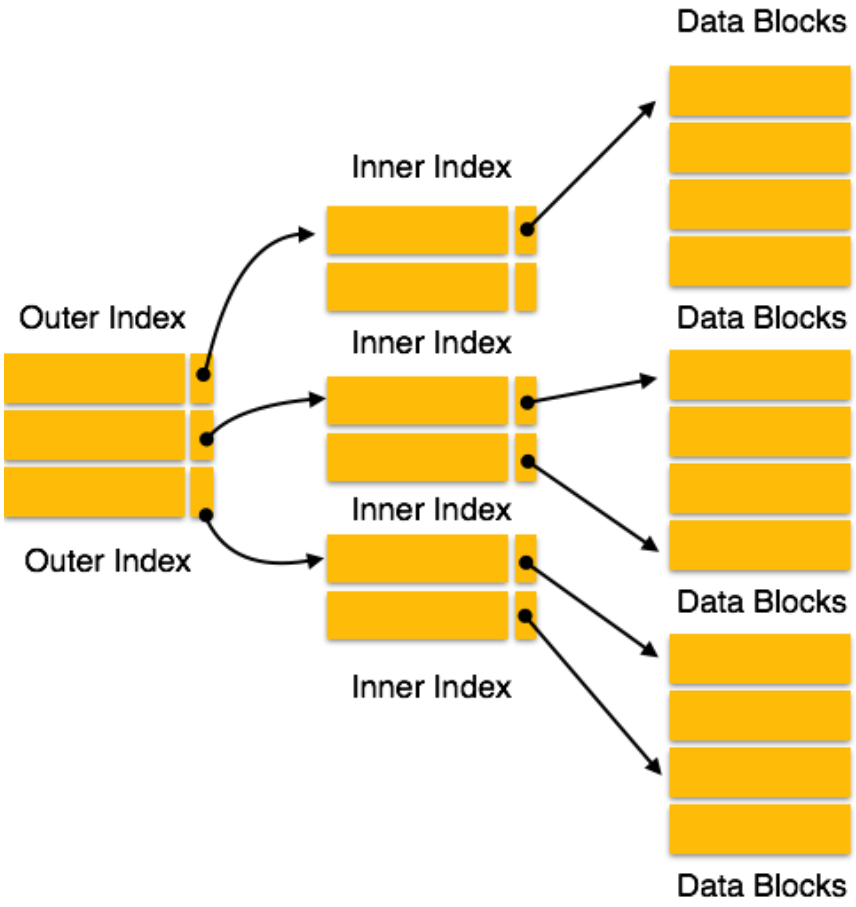
Sparse Index

In sparse index, index records are not created for every search key. An index record here contains a search key and an actual pointer to the data on the disk. To search a record, we first proceed by index record and reach at the actual location of the data. If the data we are looking

for is not where we directly reach by following the index, then the system starts sequential search until the desired data is found.

Multilevel Index

Index records comprise search-key values and data pointers. Multilevel index is stored on the disk along with the actual database files. As the size of the database grows, so does the size of the indices. There is an immense need to keep the index records in the main memory so as to speed up the search operations. If single-level index is used, then a large size index cannot be kept in memory which leads to multiple disk accesses.



Multi-level Index helps in breaking down the index into several smaller indices in order to make the outermost level so small that it can be saved in a single disk block, which can easily be accommodated anywhere in the main memory.

DBTG Model

The acronym **DBTG** refers to the **Data Base Task Group** of the Conference on Data Systems Languages (CODASYL), the group responsible for standardization of the programming language COBOL. The DBTG final report appeared in Apri1971, it introduced a new distinct and self-contained language. The DBTG is intended to meet the requirements of many distinct programming languages, not just COBOL, the user in a DBTG system is considered to be an ordinary application programmer and the language therefore is not biased toward any single specific programming language.

It is based on network model. In addition to proposing a formal notation for networks (the Data Definition Language or DDL), the DBTG has proposed a Subschema Data Definition Language (Subschema DDL) for defining views of conceptual scheme that was itself defined using the Data Definition Language. It also proposed a Data Manipulation Language (DML) suitable for writing applications programs that manipulate the conceptual scheme or a view.

Architecture of DBTG **Model**

The architecture of a DBTG system is illustrated in Figure.

The architecture of DBTG model can be divided in three different levels as the architecture of a' database system. These are:

• Storage Schema (corresponds to Internal View of database)

• Schema (corresponds to Conceptual View of database)

• Subschema (corresponds to External View of database)

Storage Schema

The storage structure (Internal· View) of the database is described by the storage schema, written in a Data Storage Description Language (DSDL).

Schema

In DBTG, the Conceptual View is defined by the schema. The schema consists essentially of definitions of the various type of record in the database, the data-items they contain, and the sets into which they are grouped. (Here, logical record types are· referred to as record types, the fields in a logical record format are called data items)

Subschema

The External view (not a DBTG term) is defined by a *subschema*. A subschema consists essentially of a specification of which schema record types the user is interested in, which schema data-items he or she wishes to see in those records, and which schema relationships (sets) linking those records he or she wishes to consider. By default, all other types of record, data-item, and set are excluded.

In DBTG model, the users are application programmers, writing in an ordinary programming language, such as COBOL that has been extended to include the DBTG data manipulation language. Each application program "invokes" the corresponding subschema; using the COBOL Data Base Facility, for example, the programmer simply specifies the name of the required subschema in the Data Division of the program. This invocation provides the definition of the "user work area" (UWA) for that program. The UWA contains a distinct location for each type of record (and hence for each type (data-item) defined in the subschema. The program may refer to these data-item and record locations by the names defined in the subschema.

Implementation of Network Model

The Network model replaces the hierarchical tree with a graph thus allowing more general connections among the nodes. The main difference of the network model from the hierarchical model, is its ability to handle many to many (N:N) relations. In other words, it allows a record to have more than one parent. Suppose an employee works for two departments. The strict hierarchical arrangement is not possible here and the tree becomes a more generalized graph - a network. The network model was evolved to specifically handle non-hierarchical relationships. As shown below data can belong to more than one parent. Note that there are lateral connections as well as top-down connections. A network structure thus allows 1:1 (one: one), l: M (one: many), M: M (many: many) relationships among entities.

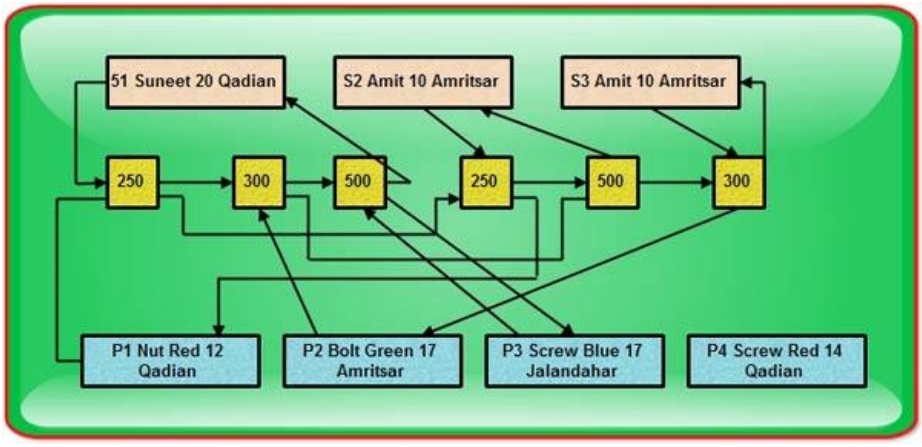
In network database terminology, a relationship is a set. Each set is made up of at least two types of records: an owner record (equivalent to parent in the hierarchical model) and a member record (similar to the child record in the hierarchical model).

The database of Customer-Loan, which we discussed earlier for hierarchical model, is now represented for Network model as shown.

In can easily depict that now the information about the joint loan L1 appears single time, but in case of hierarchical model it appears for two times. Thus, it reduces the redundancy and is better as compared to hierarchical model.

Network view of Sample Database

Considering again the sample supplier-part database, its network view is shown. In addition to the part and supplier record types, a third record type is introduced which we will call as the connector. A connector occurrence specifies the association (shipment) between one supplier and one part. It contains data (quantity of the parts supplied) describing the association between supplier and part records.



All connector occurrences for a given supplier are placed on a chain .The chain starts from a supplier and finally returns to the supplier. Similarly, all connector occurrences for a given part are placed on a chain starting from the part and finally returning to the same part.

Operations on Network Model

Detailed description of all basic operations in Network Model is as under:

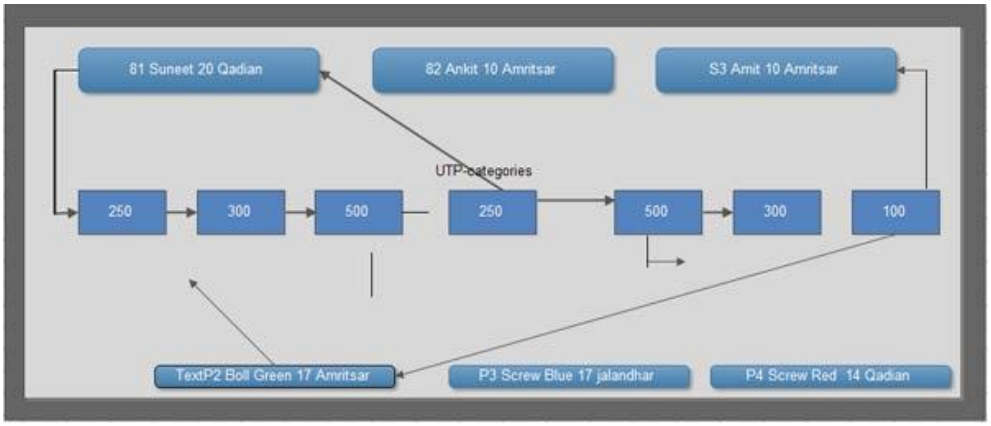
Insert Operation: To insert a new record containing the details of a new supplier, we simply create a new record occurrence. Initially, there will be no connector. The new supplier's chain will simply consist of a single pointer starting from the supplier to itself.

For example, supplier S4 can be inserted in network model that does not supply any part as a new record occurrence with a single pointer from S4 to itself. This is not possible in case of hierarchical model. Similarly a new part can be inserted who does not supplied by any supplier.

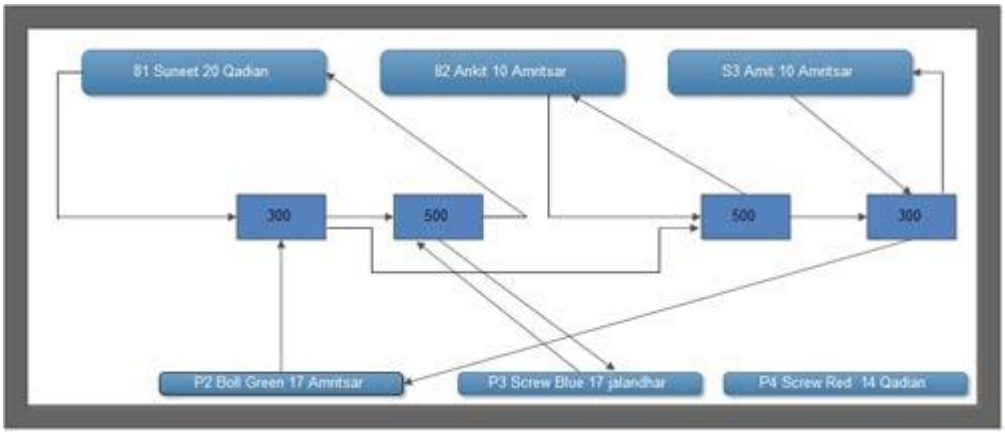
Consider another case if supplier S 1 now starts supplying P3 part with quantity 100, then a new connector containing the 100 as supplied quantity is added in to the model and the pointer of S1 and P3 are modified as shown in the below.

We can summarize that there is no insert anomalies in network model as in hierarchical model.

Update Operation: Unlike hierarchical model, where updation was carried out by search and had many inconsistency problems, in a network model updating a record is a much easier process. We can change the city of S I from Qadian to Jalandhar without search or inconsistency problems because the city for S1 appears at just one place in the network model. Similarly, same operation is performed to change the any attribute of part.

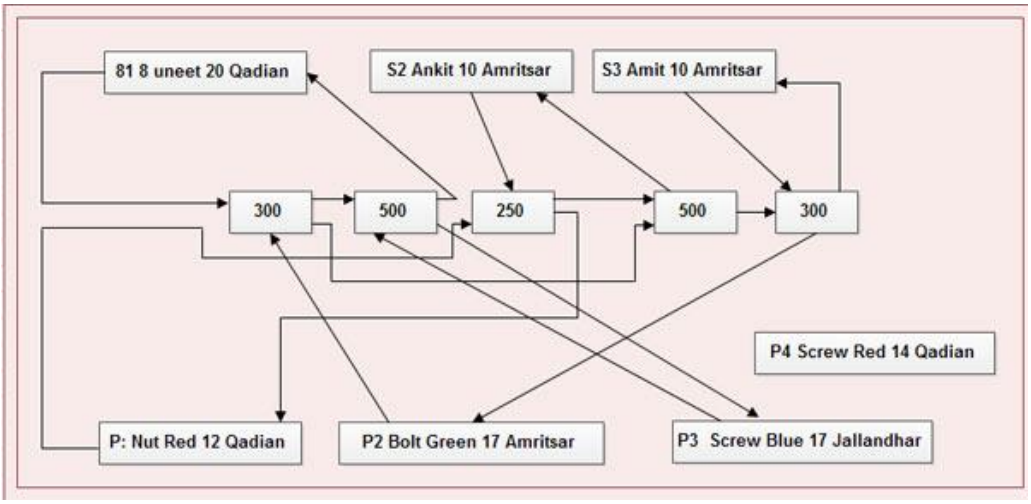


Delete operation: If we wish to delete the information of any part say PI, then that record occurrence can be deleted by removing the corresponding pointers and connectors, without affecting the supplier who supplies that part i.e. P1, the model is modified as shown. Similarly, same operation is performed to delete the information of supplier.



In order to delete the shipment information, the connector for that shipment and its corresponding pointers are removed without affecting supplier and part information.

For example, if supplier SI stops the supply of part PI with 250 quantity the model is modified as shown below without affecting P1 and S1 information.



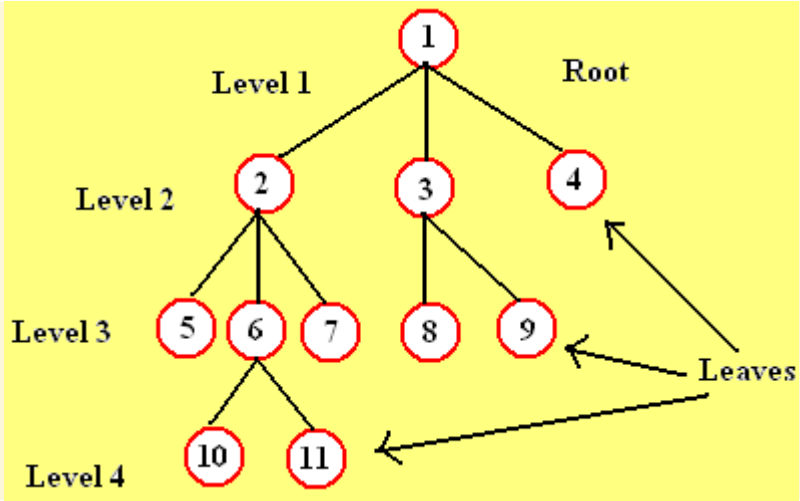
Hierarchical Model in DBMS

Hierarchical model is a data model which uses the tree as its basic structure. So, lets define the basics of the tree.

Basics of Tree :

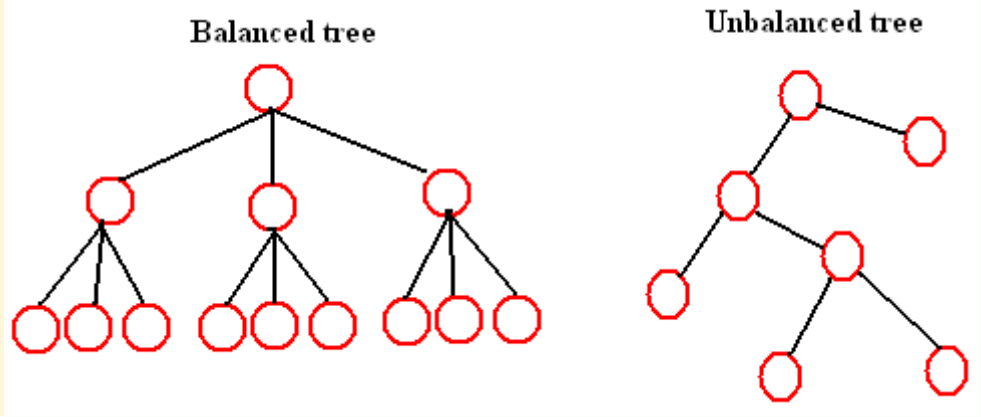
- A tree is a data structure that consists of hierarchy of nodes with a single node, called the **root at highest level**.

- A node may have any number of children, but each child node may have only one parent node on which it is dependent. Thus the parent to child relationship in a tree is one to many relationship whereas child to parent relationship in a tree is one to one.

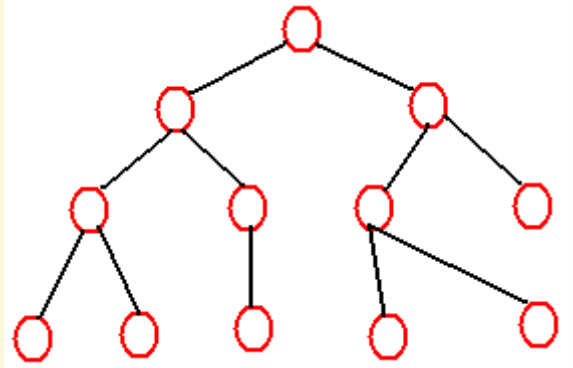


- Nodes that are children of the same parent are called **siblings**. For example, nodes 2, 3, 4 are siblings.
- For any node there is a single path called the **hierarchical path from the root node**. The nodes along this path are called that nodes **ancestors**.
- Similarly for a given node, any node along a path from that node to leaf is called its **descendent**.
- For example, suppose we have to find out the hierarchical path of node 10, then it will be 1→2→6→10 and the ancestors of node 10 are 1, 2 and 6.
- The **height of tree** is the number of levels on the longest hierarchical path from the root to a leaf. The above tree has a height= 4.
- A **tree is said to be balanced** if every path from the root node to a leaf has the same length.

Figure 2 shows a **balanced and an unbalanced tree**.



A **binary tree** is one in which each node has not more than two children. Figure 3 shows a binary tree



Example of Hierarchical Model :

- **Figure 4 shows** a data structure diagram for a tree representing the STUDENT, FACULTY and CLASS.
- The root node chosen is faculty, CLASS as a child of faculty and STUDENT as a child of class.
- The cardinality between CLASS and FACULTY is one to many cardinality as a FACULTY teaches one or more CLASS.

- The cardinality between a CLASS and a STUDENT is also one to many cardinality because a CLASS has many STUDENTS.

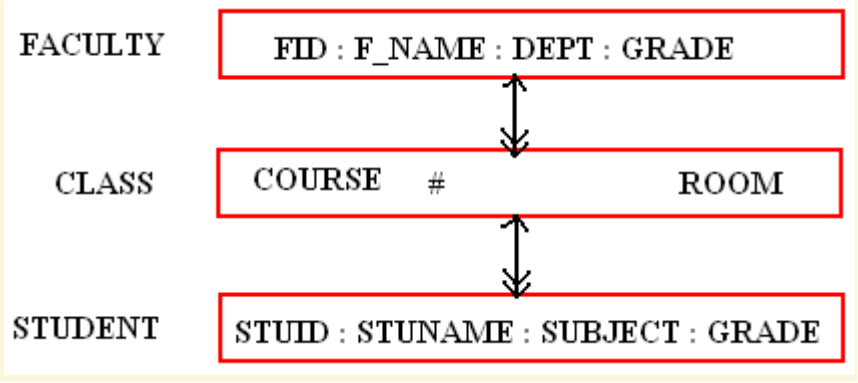
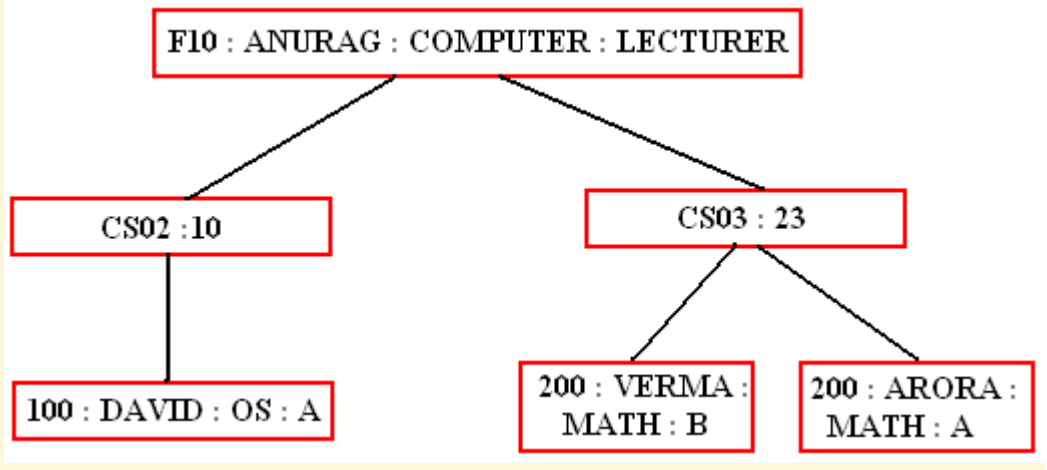


Figure 5 shows an **occurrence of the FACULTY-CLASS-STUDENT**.



Operations on Hierarchical Model

1. **Deletion-** If CS02 is deleted, then all the students in CS02 class will be deleted. So deletion is very difficult. However deletion of leaf nodes that is students does not create difficulty in deletion.
2. **Insertion-** A new class say, CS03 may not be introduced unless some faculty is available at root level. So insertion is also difficult.

3. **Updation-** Suppose a student has changed his subject from Hindi to Sanskrit, then firstly a search is performed to find out Hindi subject and then an update is made. A search is a time consuming process here.

So these problem occurs in all the three operations.

Advantages of Hierarchical Model

- Easy to understand
- Performance is better than relational data model

Disadvantages of Hierarchical Model

- Difficult to access values at lower level
- This model may not be flexible to accomodate the dynamic needs of an organisation
- Deletion of parent node result in deletion of child node forcefully
- Extra space is required for the storage of pointers