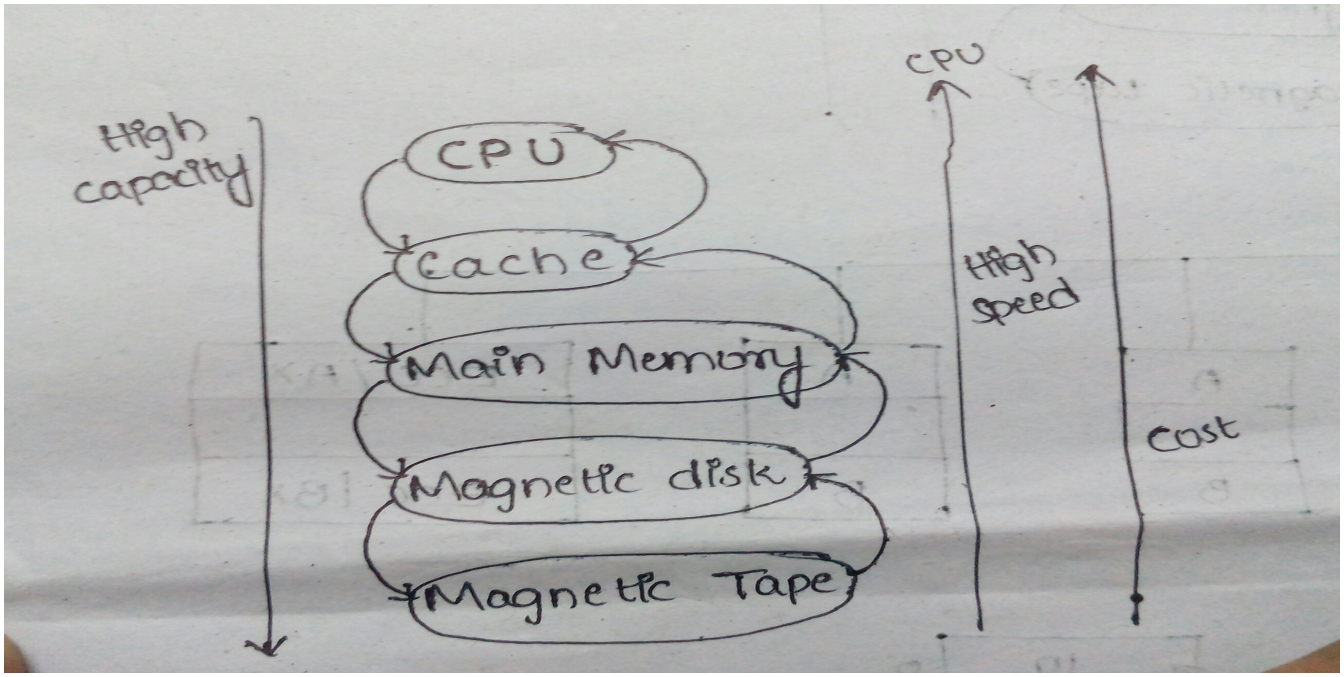
# UNIT-5

# INDEX STRUCTURE

## Memory Hierarchy:

- Memory in a computer system is arranged in hierarchial manner.




## 1.Primary Storage:

- At the top we have primary storage it consist of cache and main memory are provides very fast access of data.

**CPU:**

- It contains different types of register for temporary storage.
- It is very fast compare with cache and main memory.
- Here data stored in registers.Processed the data values,evaluate arithmetic expressions,compute the address od data values etc.

**Cache memory:**

- RAM is thousands of times slower than the CPU registers.
- It is an intermediate storage between CPU and RAM.
- It helps to fetch the data fom RAM in advance.
- It is faster than the RAM but slower than CPU.

## Maim meomry(RAM):

- It is a part of main memory which is volatile in nature.
- It stored both data and programs in increase the general speed of system.
- It is faster than hard disk but slower tan cache.
- The smallest storage unit is 1 byte.

## 2.Second storage:

- It can store the data for future use or backup purpose.
- It can consist of slower devices such as magnetic disk,optimal disk ,flash drivers etc.

## 3.Tertiary storage:

- It is also slowest devise such as optimal disk and magnetic tapes.
- Tese are store huge volume of data that's why these are very slow in speed .
- Slower storage devices such as tape and disk play on important role in dbms because the amount of data is typically very large.
- Since buying enough main memory to store all data is expensive,we must store data on tapes and disks.
- Tapes are relatively inexpensive and can store very large amount of data.
- The main drawback of tape is that they are sequential access device all data in order and can not directly access a given ocation on tape.
  For Example:
- To access the last byte on a tape,we would have a wide through entire tape,1st this tape unsuitable for storing operational data.
- Tapes are mostly used to backup operational data periodically.

## Magnetic Tape:

- Magnetic Tapes were introduced in 1928, earlier used as a secondary storage medium.
- Magnetic tape is a thin long narrow plastic strip coated with the magnetizable substance.
- The tape is wounded over a spool, and it is wounded or unwounded past a read-write head to read from or write data to the tape.
- Magnetic Tapes are **nonvolatile** in nature and hence it holds the large quantity of data **permanently**. The magnetic tapes store the data **sequentially**.
- The random access to magnetic tapes takes more time than magnetic disk because the magnetic tape has to perform **forward** and **rewind** operation to locate a correct spot.
- Magnetic tapes are also used in supercomputer centres for holding the large volume of data that is used for scientific research.


## Magnetic Disk:

- In modern computers, Magnetic Disk is used for secondary storage.
- Like Magnetic tape, the magnetic disk is also a non-volatile so, its stores the data permanently.
- The magnetic disk has several flat circular shaped platters which appear like a CD.

- ➔ The diameter of each platter ranges from 1.8 to 5.25 inches.
- ➔ Both inner and outer surfaces of the platter are covered with the magnetic material so that information can be recorded magnetically on the platters.
- ➔ There is a **read-write head** that moves over the surfaces of each platter.
- ➔ These read-write heads are attached to the disk arm that helps in moving all heads as a single unit.
- ➔ Each platter surface is divided into circular **tracks** which are further divided into **sectors**.
- ➔ The read-write head flies over the platter surface on a thin cushion of air.
- ➔ Though the disk platter is coated with a protective layer, there is always a danger that head will make contact with the disk causing **head crash**.
- ➔ The Head crash is not repairable the whole magnetic disk is to be replaced.

## Key Differences Between Magnetic Tape and Magnetic Disk

- The magnetic tapes are used for **backups** and the storage of the **data** that may be **less frequently used**. On the other hands, the magnetic disk is used as a **secondary storage** in modern computers.

- The magnetic disk has several **platters** arranged one above the other to form a cylinder, and each platter has a **read-write head** that flies over the surface of the platter.On other hands, magnetic tape is a **long thin narrow plastic strip** coated with magnetizing substance wounded over a spool.
- Magnetic tape allows **fast sequential accessing** but is **slower in random accessing**. However, the magnetic disk is **fast** in accessing data **sequentially or randomly**.
- Magnetic disk access the data faster than the magnetic tape.
- Magnetic tape can **not** be **updated** once written whereas, magnetic disk can be **updated**.
- If the magnetic tape is damaged data can be lost whereas, in the case of the magnetic disk a **head crash** can cause data loss.
- Magnetic tape has a storage capacity of **20 GB to 200 GB** whereas, the storage capacity of the magnetic disk if from several hundred **GB to Tera bytes**.
- Magnetic tape is **less expensive** as compared to the magnetic disk.

## RAID(Redundant Array Of Independent/Inexpensive Disks):

- It is a technology which is used to connect multiple secondary storage devices for to increase performance,data redundancy or both,reliability of resulting storage system.
- It contains array of disks in which multiple disks are connected to achieve different goals.

## RAID Advantages:

- Availability(Almost 99.99% data recovery of disk fails,gain connect from remaining disks).
- Reliability(performs consistency well)-if want whatever data gives that type of data.
- Capacity(Big amount of data stores).

- Performance(High performance).
- Data Loss Prevention.

## RAID Disadvantages:

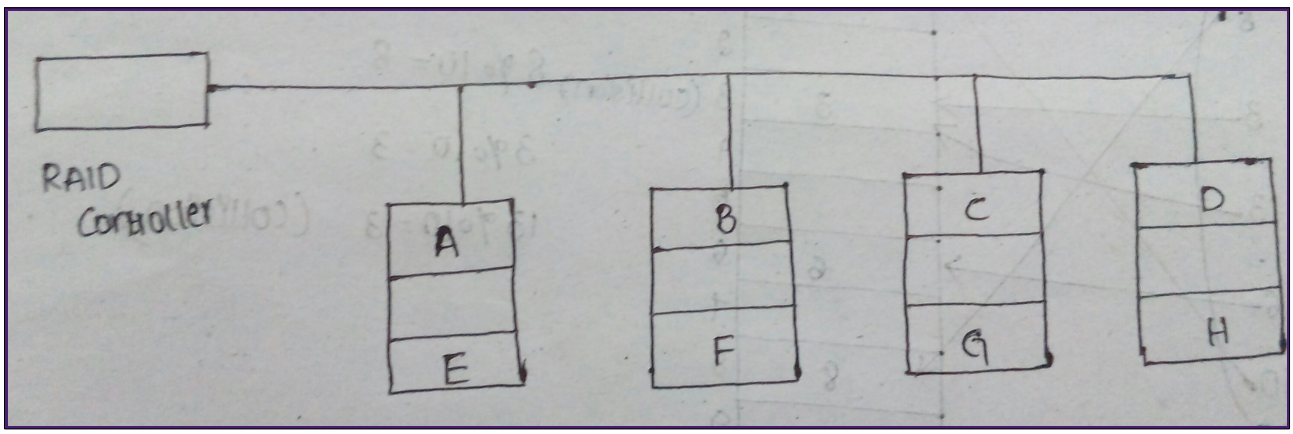
- Expensive Comparatively(controller cost)
- Data recovery becomes time consuming in case of multiple disk failure.

## RAID Levels :

Without RAID technology our storage system would consists of four disks.Depending on the RAID levels choosen the no.of additional disks varies from zero to four..

### (i)RAID Level 0:(Data Striping)

- In this level uses data striping to increase performance.No redundent information is maintained.
- Data Striping distributes data over several disks.
- The data is broken into blocks and blocks are distributes among disks.each disks receive a block of data to read/write in parallel.
- It enhance the speed and performance of storage device.
- There is no parity and backup in level 0.
- It consists of four data disks,independent of data disks,the effective space utilization for RAID level 0 system is 100%.
- There is no duplication of data,hence a block once lost cannot be retrieved..



## Example:

| Disk 0 | Disk 1 | Disk 2 | Disk 3 |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |

### Pro's of RAID 0:

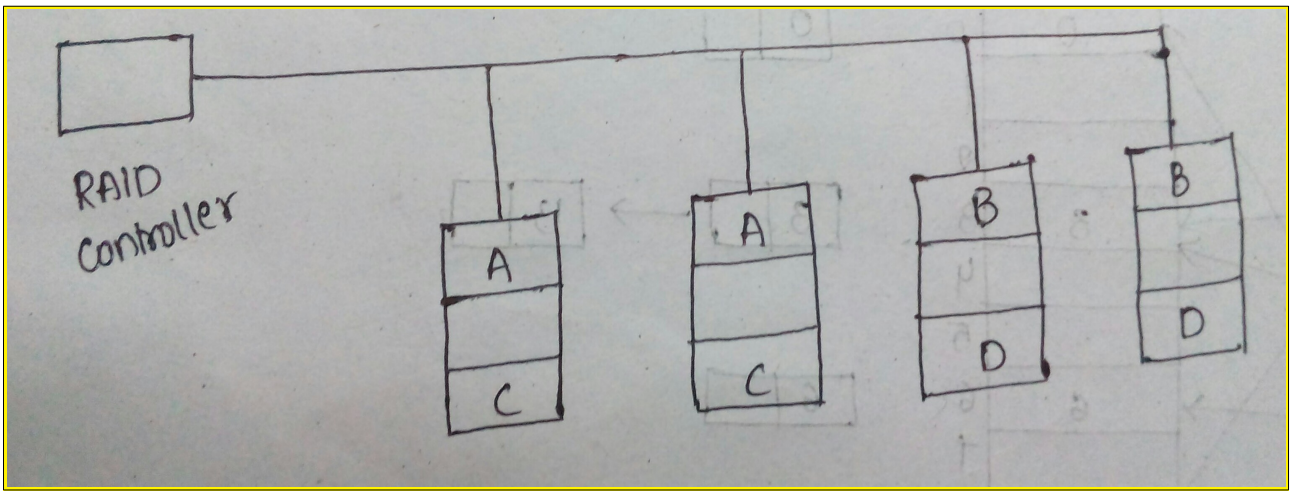
- Throughput increases because multiple data request probably not on the same disk.
- It fully utilize the disk spaces and provides high performance.
- It requires minimum "2" disks.

### Con's of RAID 0:

- It doesn't contain error detection mechanism.
- It is not true RAID because it is not fault tolerance.
- Failure of one disks result in controls loss of respective disk.

### (ii)RAID Level 1:(Mirroring)

- It is most expensive solution.Instead of having one copy of the data,two identical copies of the data on two idifferent disks are maintained.This type of redundancy is known as "Mirroring"
- Every write of a disk block involves write on both disks.It is best OS disk.
- This level uses mirrory techniques,when data sent to RAID Controller, it sends to a copy of data to all the disks in array.
- It provides 100% redundancy incase of failure.50% of space utilization.
- Here only half space of drive is used to store the data.The other half of drive is just mirror to the already stored data.




## Example:

| Disk 0 | Disk 1 | Disk 2 | Disk 3 |
|---|---|---|---|
| A | A | B | B |
| C | C | D | D |
| E | E | F | F |

## Pro's of Level 1:

- Fault Tolerance,in this level if one disks fails,then the other automatically takes over.
- In this level,the array will function even if any one of the drive fails.

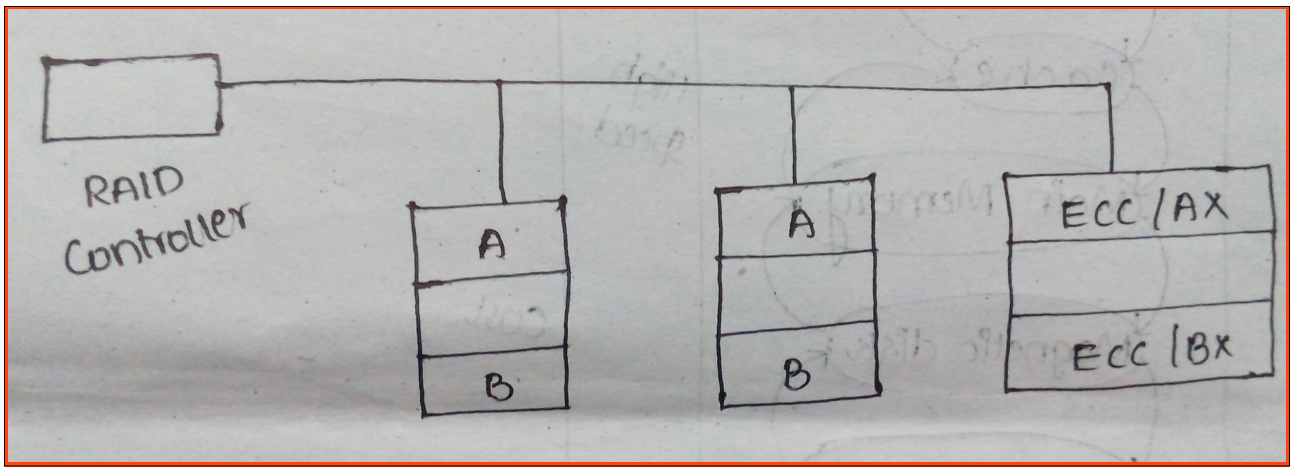
## Con's of Level 1:

- In this level one extra disk is required for mirroring,so expensive is higher.
- Sometimes it is also called as "RAID level 10" combines striping and mirroring.

## RAID Level 2:(Error-Correction Code(ECC))

- The redundancy scheme used Hamming code.
- It consists of bit level striping.
- In this level,each data bit in a word is related on seperate disk and ECC of dataword is stored on different set disks.

- Due to high cost and complex structure this level is not commercial used.




- In above example,only one checkdisk is needed,in general no.of checkdisks increases logarithmatically with no.of data disks.
- No checkdisk which disk fails.

**Pro's of Level 2:**

- Hamming code for error detection.
- This level uses one partial drive to store parity.

**Con's of Level 2:**

Additional drives for error detection.

## RAID Level 3:(Bit -interleaved parity)

- It overcomes the drawbacks of level 2.
- It keeps redundancy information that is necessary.
- Hamming code is used in RAID level 2,has the advantage of being able to identify which disk has failed.In level 3, we use checkdisk with parity information and which was failed.

**Example:**

| Disk 0 | Disk 1 | Disk 2 | Disk 3 |
|---|---|---|---|
| A | B | C | P(A,B,C) |
| D | E | F | P(D,E,F) |
| G | H | I | P(G,H,I) |

- Incase of drive failure,the parity drive is accessed and data is reconstructed from remaining device once the failed drive is replaced ,the missing data can be restored on new drive.
- Data tranfer in bulk then high-speed performance data transfer.

**Pro's of Level 3:**

- Data is regenerated using parity drive.
- It contains high data transfer rates.
- Data is accessed in parallel.

**Con's of Level 3:**

- Additional drive for parity.
- Slow performance.

**RAID Level 4:(Block -interleaved parity)**

- In level 3 we divide the data in bitwise but in level 4 we divide the data in blockwise.
- RAID level 3 and level 4 require atleast three disks for implementation.
- In level 4,one disk dedicated to parity.
- In this level,prity can be calculated using an XOR function.If data bits are 0,0,0,1 then the parity bits is XOR(0,0,0,1)=1.

**New Parity=(Old data XOR New data) XOR old parity**

- If the parity bits are 0,0,1,1 then the parity bit is XOR(0,0,1,1)=0.

That means,even no.of one results in parity 0 and odd no.of one results in parity 1.

| Disk 0 | Disk 1 | Disk 2 | Disk 3 |
|---|---|---|---|
| $A_1$ | $A_2$ | $A_3$ | $P_A$ |
| $B_1$ | $B_2$ | $B_3$ | $P_B$ |

**Example:**

| C1 | C2 | C3 | C4 | parity |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |

- ➔ Effective space utilization is 80%.
- ➔ check disk is required.
- ➔ Recovery of disk failure.
- ➔ The read-modify-write cycle involves reading of old data block and the old parity block,modifying the two blocks,and writing them back to disk,resulting in four disk accessed per write.

## RAID Level 5 :(Block -interleaved distributed parity)

- ➔ This level improves upon level 4 by distributing the parity blocks uniformly over all disks,instead of storing them as a single check disk.
- ➔ Same as level 4,this level allow recovery of atmost one disk failure.If morethan one disk fails,then there is no way for data recovery.

**Example:**

| Disk 0 | Disk 1 | Disk 2 | Disk 3 | Disk 4 |
|---|---|---|---|---|
| P0 | 1 | 2 | 3 | 4 |
| 4 | P1 | 5 | 6 | 7 |
| 8 | 9 | P2 | 10 | 11 |
| 12 | 13 | 14 | P3 | 15 |
| 16 | 17 | 18 | 19 | P4 |

## Pro's of Level 5:

- Cost effective and high performance.
- In this level,parity distributed across all disks in an array.

## Con's of Level 5:

- Disks failure recovery take longer time as parity has to be calculated from all available drives.

## RAID Level 6 : (P+Q)Redundency

- ➔ This level is extension of level 5.It consists of block-level-strips with 2 parity bits.
- ➔ In RAID 6,You can service 2 concurrent disk failures,suppose you are using RAID 4,RAID 5.
- ➔ When your disk fail, you need to replace the failed disk bacause if simultaneously another disk fails then won't be able torecovery any data,so in this case RAID 6 plays its part where you can survive two concurrent disk failure.

**Example:**

| Disk 0 | Disk 1 | Disk 2 | Disk 3 |
|---|---|---|---|
| A0 | B0 | Q0 | P0 |
| A1 | Q1 | P1 | D1 |
| Q2 | P2 | C2 | D2 |
| P3 | B3 | C3 | Q3 |

## Pro's of Level 6:

- This level performs RAID 0 to strip data and RAID 1 to mirror.In this level striping is performed before mirroring.
- Drive required should be multiple of 2.

## Con's of Level 6:

- It is not utilised 100% disk capacity as halt is used for mirroy.
- It contains of very limited Scalability.

## Disk Space Management:

- ➔ The Lowest level of s/w in the DBMS architecture is called "Disk Space Mangement",who manages space on disk.

- The Disk Space manager supports the paging technique as unit of data and provides commands to allocate or deallocate a page and read (or) write a page.
- The size of page is chosen to be the size of disk and pages are stored as disk blocks so that reading and writing a page can be done in one disk I/O.
- It is often useful to allocate a sequence of pages as a contiguous sequence of blocks to hold data frequently accessed in Sequential Order.
- A Database grows and Shrinks as records are inserted and deleted over time.
- The Disk Space Manager Keeps track of which disk blocks are in use,in addition to keeping track of which pages are on which disk blocks.Although it is likely that blocks are initially allocated sequentially on disk.
- One way to keep track of block usage to maintain a list of free blocks.As blocks are deallocated,we can add them to the free list for future use.
- A pointer to the first block on the free block list is stored is known as “Location on disk”.
- A secondary way is to maintain a bitmap with one bit for each disk block,which indicates whether a block is in use (or)not.
- A bitmap also allows very fast identification and allocation of contiguous areas on disk.

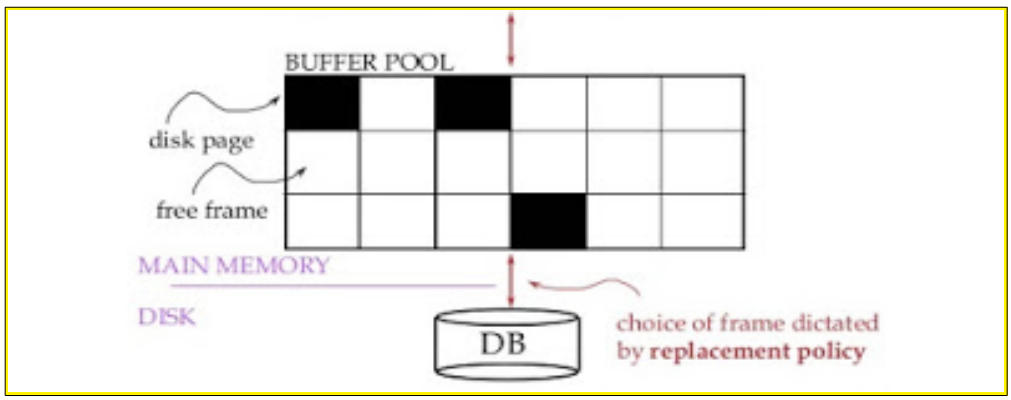
## Using OS FileSystem to Manage Disk Space:

- OS also manage space on disk.Typically,an OS Supports the abstraction of a file as Sequence of bytes.
- The OS manages space on disk and translates requests such as “read byte i of file f” into low-level instructions: “read block m” of track t of cylinder c of disk d.
- A Database disk space manager could be build using OS files.For example,the entire database could reside in one (or) more OS files for a no.of blocks are allocated and initialized.

- The Disk Space Manager is then responsible for managing space in OS files.
- Many DBS rely(Depends) on the OS FileSystem and instead of their own disk management,A technical reason is that on a 32-bit File System,the Largest file size is 4GB,whereas a DBMS may want to access a single file larger than that..

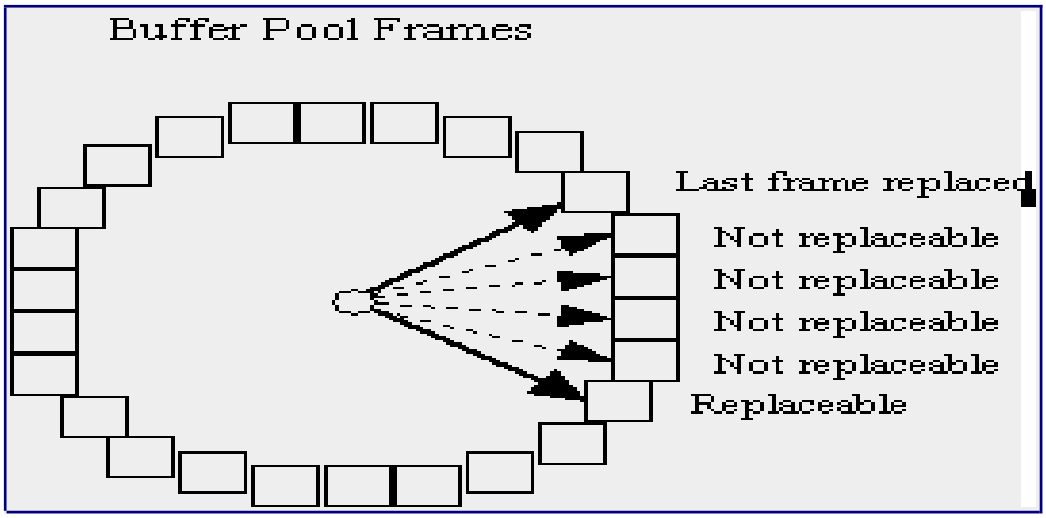
## BUFFER MANAGER:

- The **buffer manager** is the software layer that is responsible for bringing pages from physical disk to main memory as needed.
- The buffer manages the available main memory by dividing the main memory into a collection of pages, which we called as **buffer pool.**
- The main memory pages in the buffer pool are called **frames.**




- The goal of the buffer manager is to ensure that the data requests made by programs are satisfied by copying data from secondary storage devices into buffer.
- Infact, if a program performs reading from existing buffers. similarly, if a program performs an output statement: it calls the buffer manager for output operation - to satisfy the requests by writing to the buffers.

- Therefore, we can say that input and output operation occurs between the program and the buffer area only.
- If an input operation does not find the requested page in the buffer pool, then the buffer manager (software) will have to do a physical transfer the page from the secondary memory (disk) to a free block in buffer pool and then make the requested page placed in the buffer pool, that is, available to the program requesting the original input operation.
- A similar scenario will take place in the revers order for an output operation. That is, the buffer manager (software) makes a new empty buffer a available to the program for outputting the page.
- If there is no space in the buffer pool then the buffer manager (software) physically transfer one of the page from buffer pool to the disk (secondary memory) to provide the empty space in the buffer pool for the output operation of the program to display the page in the buffer pool.
- In addition to the buffer pool itself, the buffer manager maintains same block keeping information and two variable for each frame in the pool ; **pin - count** and **dirty.**
- The number of times the page is requested in the frame - each time the pin - count variable is incremented for that frame (because that page is in this frame).
- For satisfying each request of the user ; the pin - counter variable is decremented each time for that frame.
- Thus, if a page is requested the pin - count is incremented ; if it fulfills the request the pin - count is decremented.
- In addition to this, if the page has been modified the Boolean variable; dirty is set as 'on'. Otherwise 'off '.




## FILE ORGANISATION:

- File is a Sequence of records stored in binary format.
- A Disk drive formatted into several blocks that can store records.
- Fileorganization defines how file records are mapped on the disk blocks.It is logical relationships among various records that means identification and access to any specific record,
- In simple terms,Storing the files in certain order is called "File Oraganization"

## Types of File Organization:

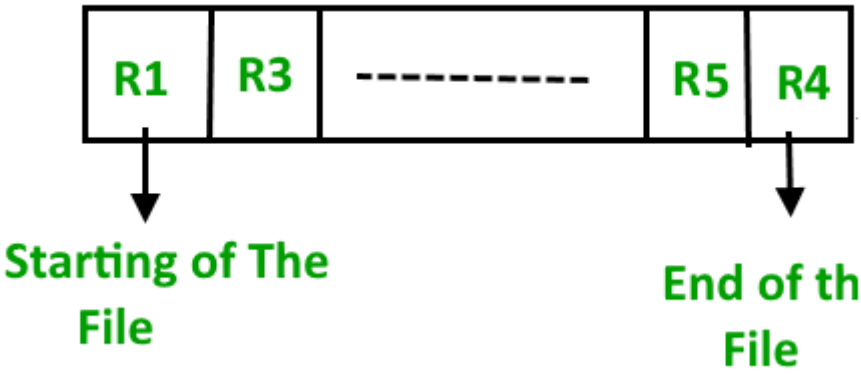
(1)Sequential File Organisation
(2)Heap File Organisation
(3)Hash File Organisation
(4)B+ tree File Organisation
(5)Clustered File Organisation

## (1)Sequential File Organisation:

The easiest method for file organization is sequential method.In this method the files are stored one after another in sequential manner.There are two ways to implement this method.

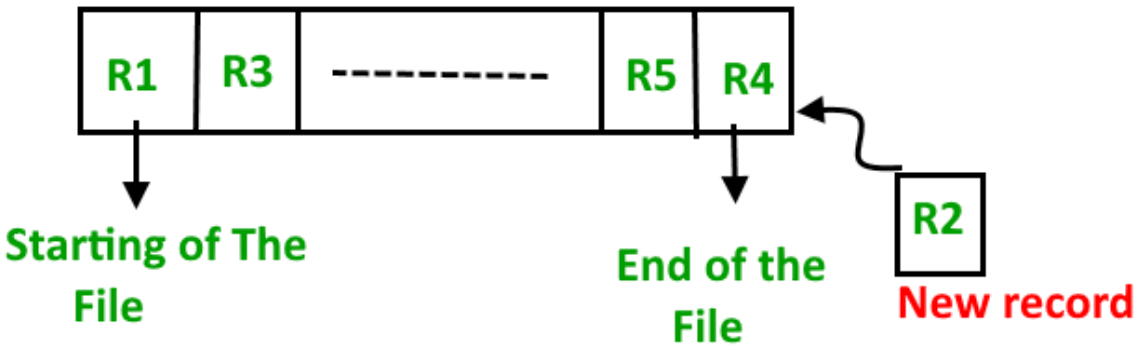
## (a)Pile File method:

In this method records are stored in sequence i.e., one after another in the order in which they are inserted into the tables.




## Insertion of new record:

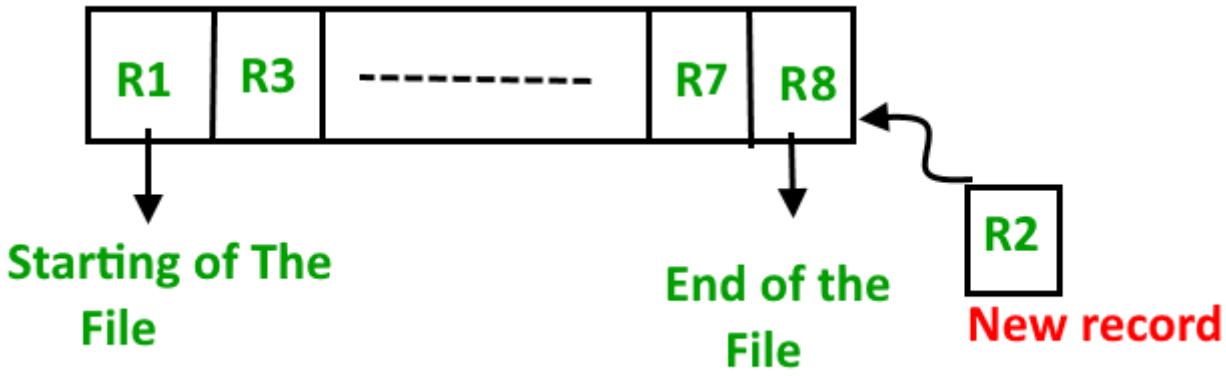
Let the R1,R3 and so on upto R5 and R4 be four records in the sequence,here records re nothing but a row in any table,suppose a new record R2 has to be inserted in the sequence,then it is simply placed at the end of the file.




## (b)Sorted File method:

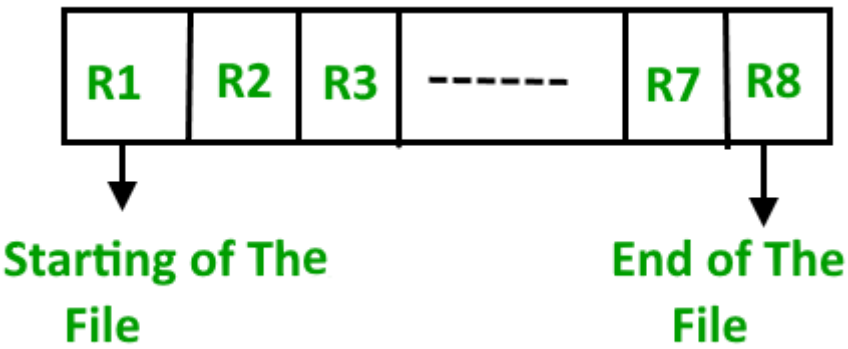
In this method as the new name itself suggest whenever a new record has to be inserted in sorted order like ascending or

descending order,sorting of records may be based on primary key or another key.




**Insert a new record:**

Let us assume that there is pre-existing sorted sequence R1,R3 and so on upto R7,R8.Suppose a new record R2 has to be inserted in this sequence,then it will be inserted in at the end of the file and then it will sort the sequence.




## Pro's of Sequential FO:

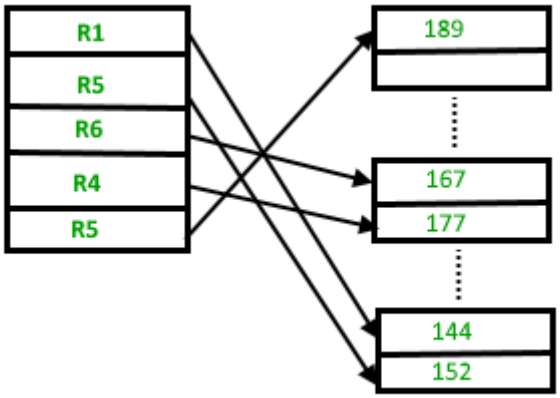
- Fast and efficient method for bulk amount of data.
- Simple to Design.

## Con's of Sequential FO:

- Sorted File method is takes time and Space for sorted records.

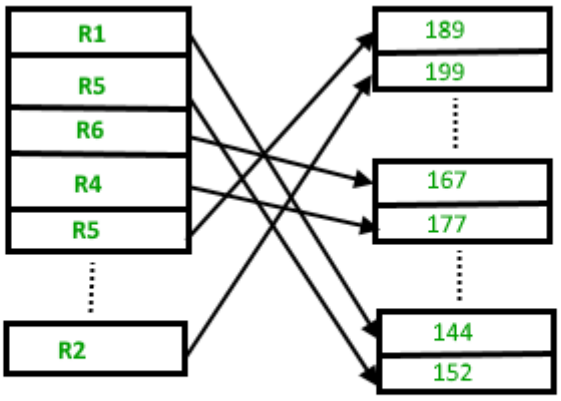
## 2.Heap file organization:

- Heap file Organization works with data blocks.
- In this method records are inserted at the end of the file,into the data blocks.No sorting or ordering is required in this method.
- If a data block is full,the new record is stored in some other block,here the other data block need not tobe very next data blocks.
- But it can be stored in any block in the memory.It is responsibility of DBMS to store and arrange the new records.




## Insertion of new record:

- suppose we have file seems R1,R5,R6,R4,R3 and suppose a new record R2 has to inserted in the heap then,since the last block i.e, data block 3 is full it will be inserted in any of the data blocks,selected by BDMS ,let say data block




- If we want to update,search,delete data in heap file organization will take lot of time because database is very large storage.

## Pros of Heap FO:

- Fetching and retrieving records is faster than sequential record only in small databases.
- When huge no.of data needs to be needed into the database at a time,then this method is suited.

## Cons of Heap FO:

- problems of unused blocks.
- Inefficient for large databases.

## 3.Hash file Organization:

- It is an efficient method to directly search location of data resided in without using index structure.
- Data is stored at datablocks where address is generated by the hash function.
- The memory location where these records are stored is called as “datablock” or “data bucket”.

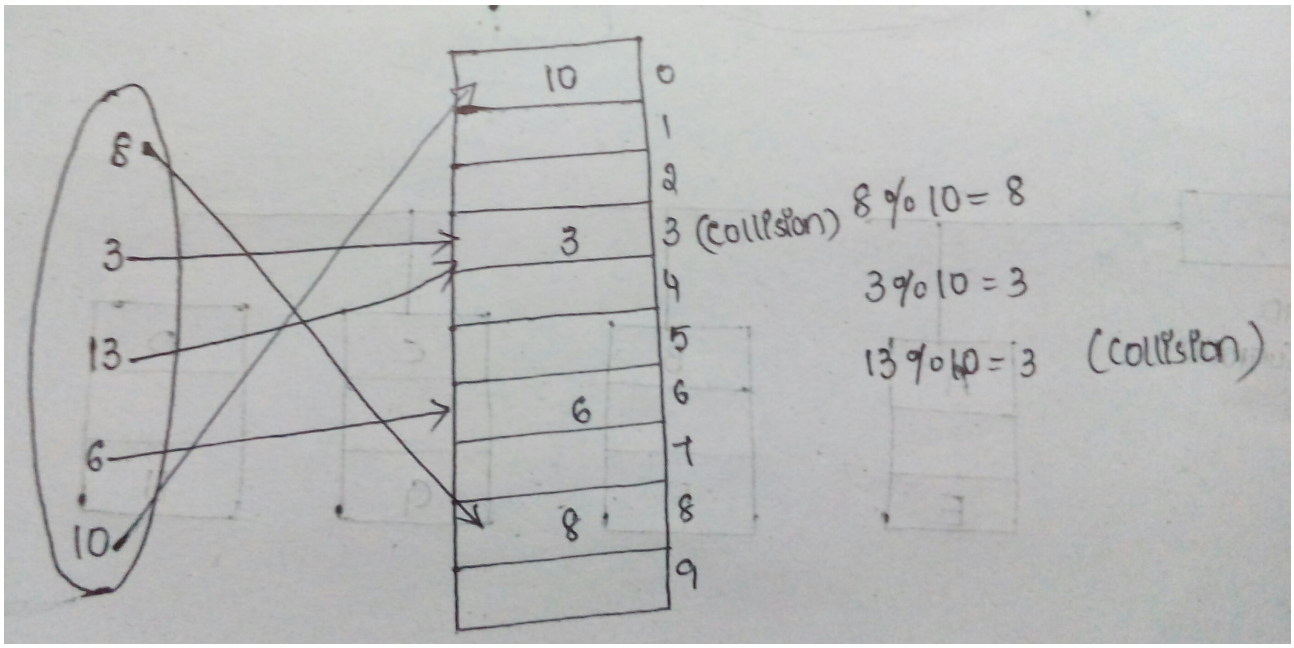
## Hash function:

- It is a mapping function that maps the set of search keys to actual record address generally hash function use primary key to generate hash values.

➔ It is a simple mathematical function.

**h(x)=x%size**

where size is the size of hash table




➔ We Can resolve collision using two methods:
a)Open hashing
b)Closed hashing

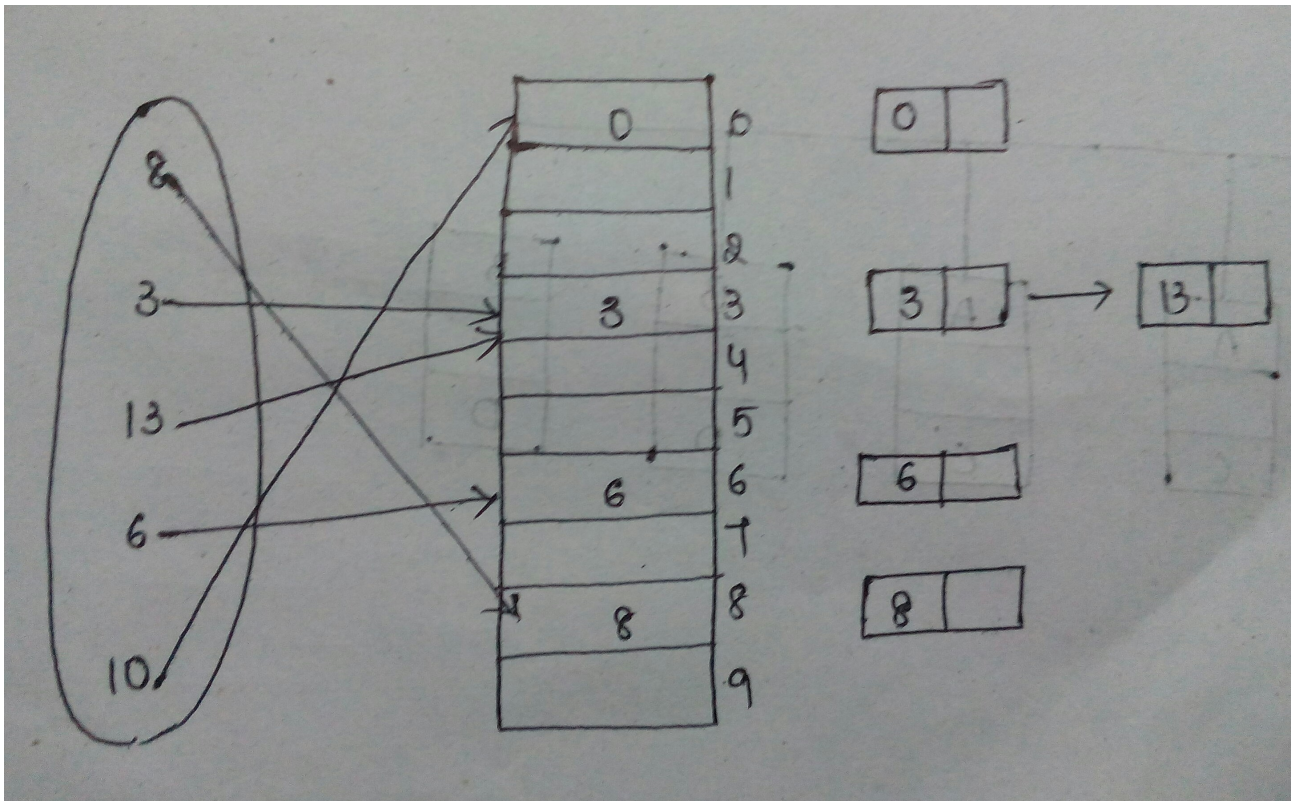
## a)open hashing:

➔ In open hashing,search keys are stored in next available blocks is used to enter the new record instead of old record.

**Chaining:**

➔ This method is used for the collision is occured at same data block we use chaining method to resolve the collision.

**Example:**




## b)Closed Hashing:

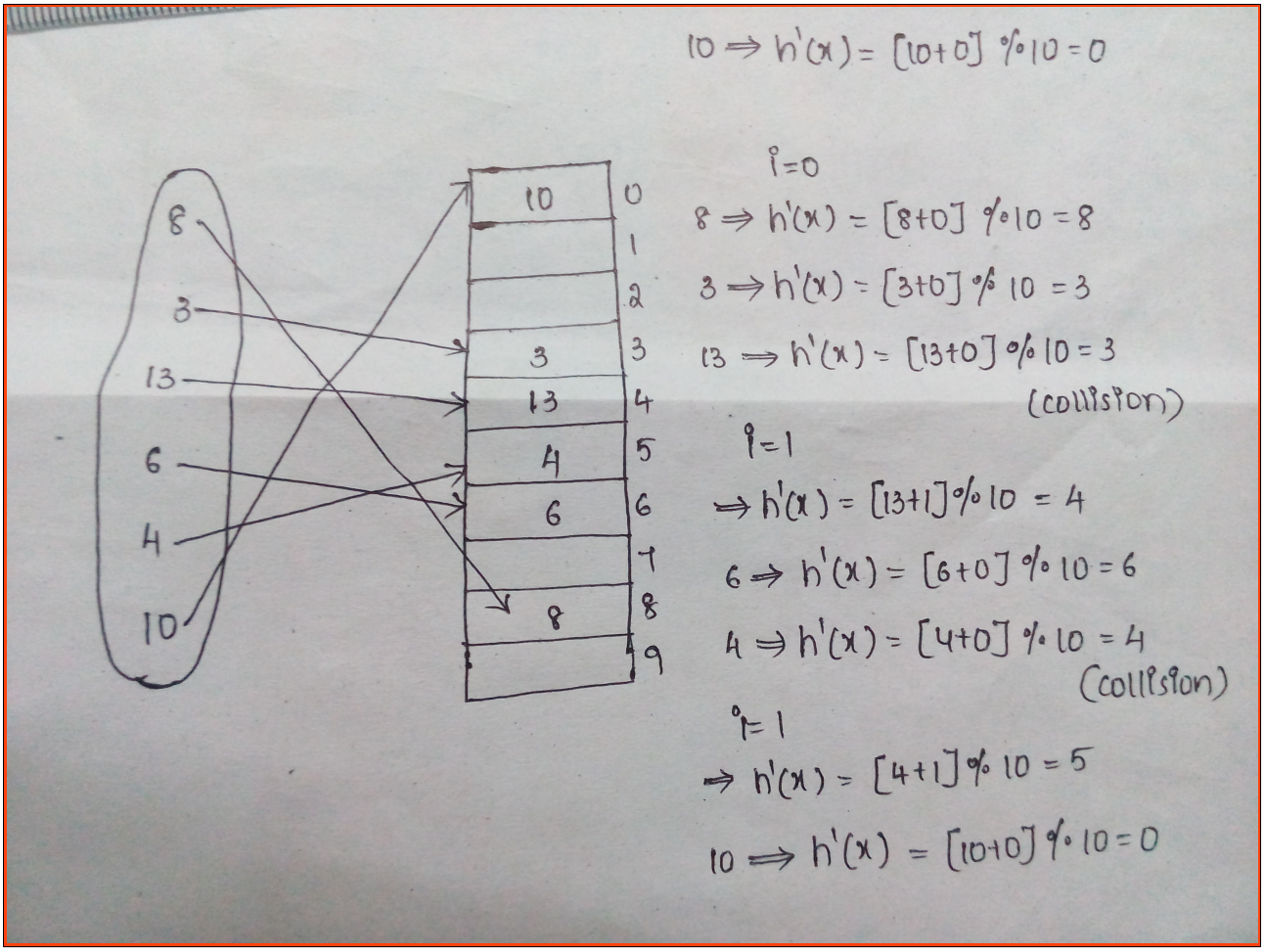
- ➔ In this method a new data block is allocated with same addresss and linked with after full data block.
- ➔ It is also called as “overflow chaining”
- ➔ It is further divided into two types.They are
  a)linear probing
  b)quadratic probing

## a)linear probing

**formula: h’(x)=[h(x)+f(i)]%size**

where f(i)=i (Linear)

**i=0,1,2,3,4.....**

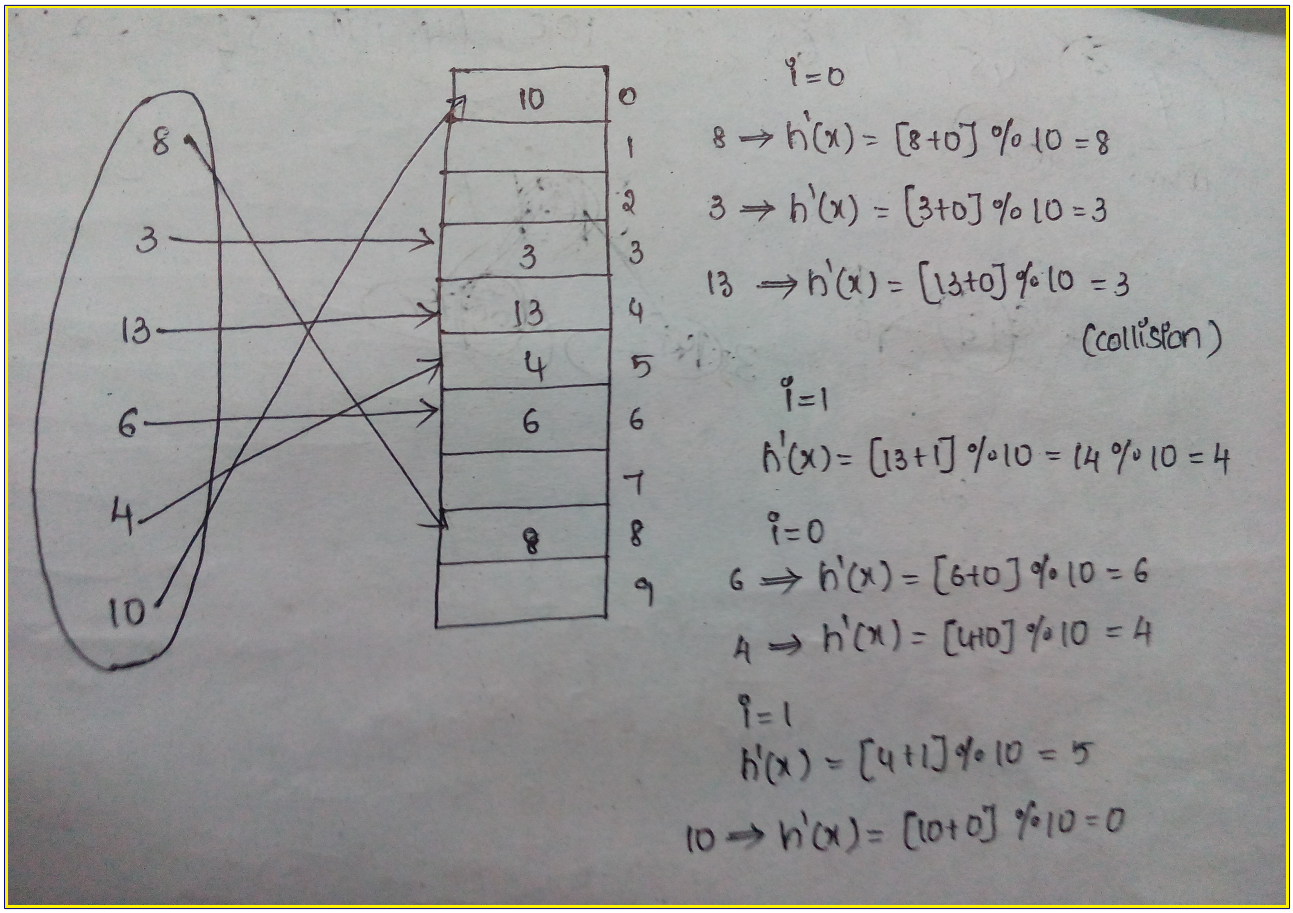



## b)Quadratic Probing:

**formula: h'(x)=[h(x)+f(i)]%size**

where f(i)=i^2 (quadratic)

i=0,1,4,9.....




## 5.Clustered File Organization:

- In this method two or more related tables or records are stored within a same file is called “clustered file”
- These files will have two or more tables in the same data blocks and the key attributes which are used to map these tables together and stored only once.
- It is a cost of searching and retrieving the different records in same file.
- For example,we have two tables i.e emolyee and department table.These two tables are related to each other and combined using join operation can seen in clustered file.

**Example:**

**Table-1:Employee_Table**

| eid | name | address | dept_id |
|---|---|---|---|
| 1 | a | vizag | 501 |
| 2 | b | chittoor | 501 |
| 3 | c | guntur | 502 |
| 4 | d | kadapa | 503 |
| 5 | e | ongole | 504 |

**Table-2:Deptment_Table**

| dept_id | dept_name |
|---|---|
| 501 | cse |
| 502 | ece |
| 503 | eee |
| 504 | civil |

**Final Table:**

| dept_id | dept_name | eid | name | address |
|---|---|---|---|---|
| 501 | cse | 1 | a | vizag |
| 501 | cse | 3 | c | guntur |
| 502 | ece | 2 | b | chittoor |
| 503 | eee | 4 | d | kadapa |
| 504 | civil | 5 | e | ongole |

It is further divided into two types.

a)Indexed cluster

b)Hash cluster

**a)Indexed cluster:**

- In this method records are grouped based on cluster key or foreign key and stored together.

- ➔ In the above example,employee and department are examples for indexed cluster where records are based on dept_id

**b)Hash cluster:**

- ➔ This method is similar to indexed cluster with only one difference instead of storing records based on cluster key,it generate hash key values and stored the records with same hash key values.

## INDEXING:

Indexing is defined as a data structure technique which allows you to quickly retrieve records from a database file. It is based on the same attributes on which the Indices has been done.

An index

- Takes a search key as input
- Efficiently returns a collection of matching records.

An Index is a small table having only two columns. The first column comprises a copy of the primary or candidate key of a table. Its second column contains a set of pointers for holding the address of the disk block where that specific key value stored.

Database Indexing is defined based on its indexing attributes. Two main types of indexing methods are:

- Primary Indexing
- Secondary Indexing

## Primary Indexing:

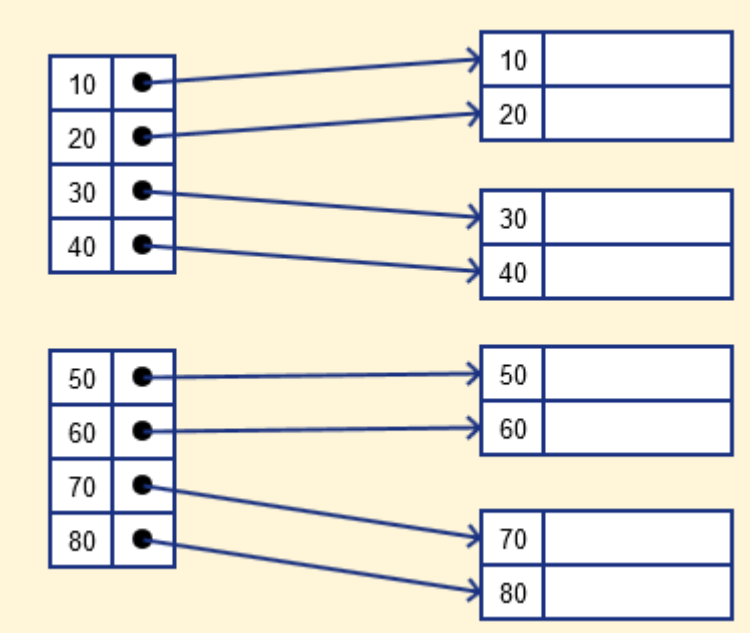
Primary Index is an ordered file which is fixed length size with two fields. The first field is the same a primary key and second, filed is pointed to that specific data block. In the primary Index, there is always one to one relationship between the entries in the index table.

The primary Indexing is also further divided into two types.

- Dense Index
- Sparse Index

## Dense Index

In a dense index, a record is created for every search key valued in the database. This helps you to search faster but needs more space to store index records. In this Indexing, method records contain search key value and points to the real record on the disk.



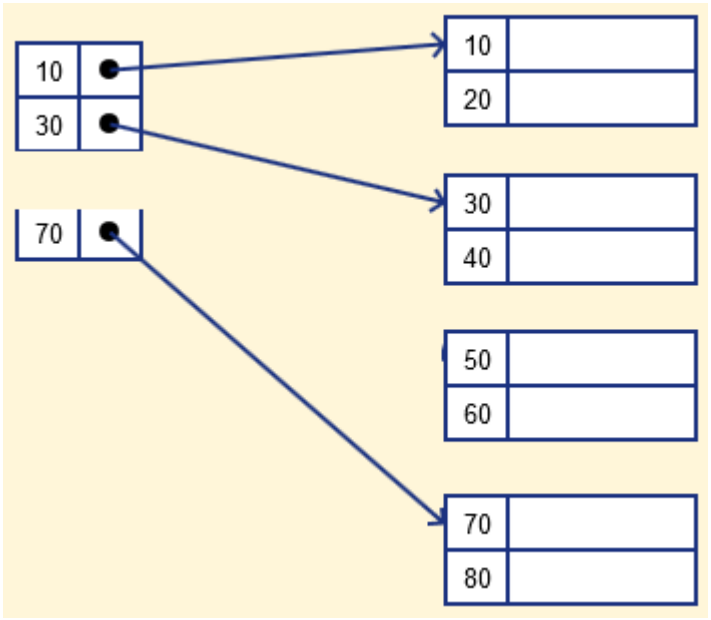

## Sparse Index:

It is an index record that appears for only some of the values in the file. Sparse Index helps you to resolve the issues of dense Indexing. In this method of indexing technique, a range of index columns stores the same data block address, and when data needs to be retrieved, the block address will be fetched.

However, sparse Index stores index records for only some search-key values. It needs less space, less maintenance overhead for

insertion, and deletions but It is slower compared to the dense Index for locating records.

### Example of Sparse Index:




## Secondary Index:

The secondary Index can be generated by a field which has a unique value for each record, and it should be a candidate key. It is also known as a non-clustering index.

This two-level database indexing technique is used to reduce the mapping size of the first level. For the first level, a large range of numbers is selected because of this; the mapping size always remains small.

## Example of secondary Indexing

In a bank account database, data is stored sequentially by acc_no; you may want to find all accounts in of a specific branch of ABC bank.

Here, you can have a secondary index for every search-key. Index record is a record point to a bucket that contains pointers to all the records with their specific search-key value.

## Clustering Index:

In a clustered index, records themselves are stored in the Index and not pointers. Sometimes the Index is created on non-primary key columns which might not be unique for each record. In such a situation, you can group two or more columns to get the unique values and create an index which is called clustered Index. This also helps you to identify the record faster.

### Example:

Let's assume that a company recruited many employees in various departments. In this case, clustering indexing should be created for all employees who belong to the same dept.

It is considered in a single cluster, and index points point to the cluster as a whole. Here, Department _no is a non-unique key.

## What is Multilevel Index?

Multilevel Indexing is created when a primary index does not fit in memory. In this type of indexing method, you can reduce the number of disk accesses to short any record and kept on a disk as a sequential file and create a sparse base on that file.

There are two types of multilevel Index..They are:

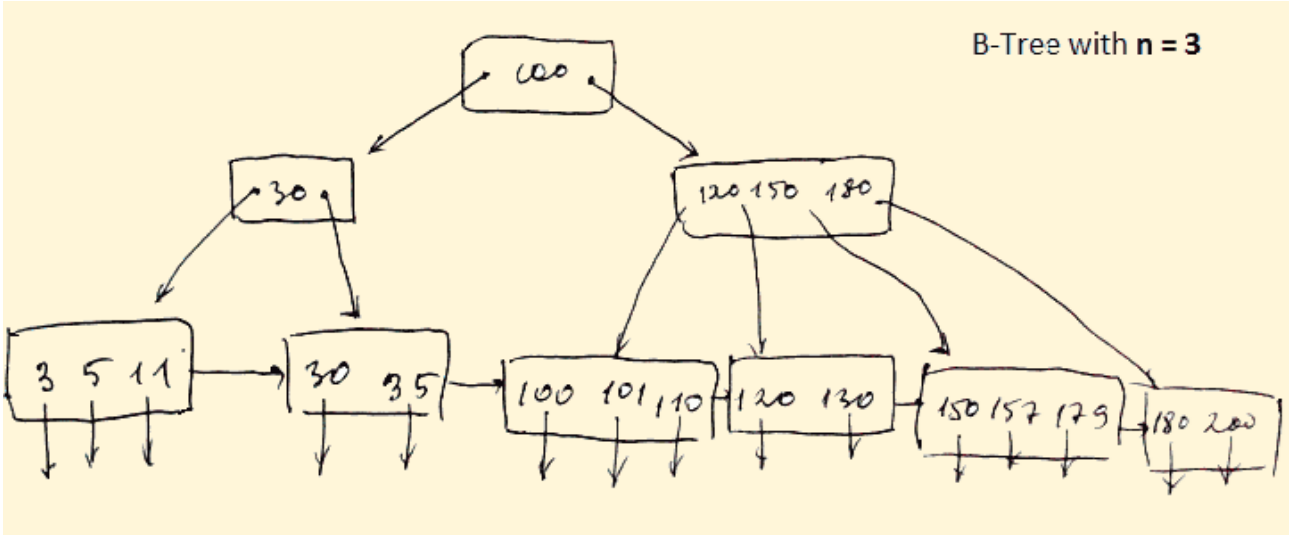
1.B Tree

2.B+ Tree

## B-Tree Index:

B-tree index is the widely used data structures for Indexing. It is a multilevel index format technique which is balanced binary search trees. All leaf nodes of the B tree signify actual data pointers.

Moreover, all leaf nodes are interlinked with a link list, which allows a B tree to support both random and sequential access.




- Lead nodes must have between 2 and 4 values.
- Every path from the root to leaf are mostly on an equal length.
- Non-leaf nodes apart from the root node have between 3 and 5 children nodes.
- Every node which is not a root or a leaf has between n/2] and n children.

## B+ Trees:

- B+ tree is a balanced binary search tree.It follows a Multi-level Index Format.

- In the B+ tree Leafnodes denote actual data pointers B+ tree ensures that all leafnodes remain at same height.
- In B+ tree leafnodes are linked using a linkedlist and support both random and sequential access.

## Structure of B+Trees:

- In B+ Tree,every leaf node is at equal distance from root.
- It contains an internal node and leafnode.

## Internal Node:

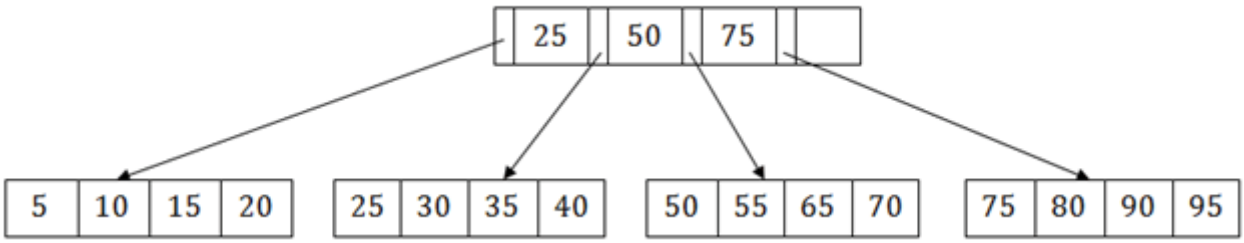
- An internal node of B+ tree can contain atleast n/2 record pointers except the root node.
- Atmost, an Internal node of the tree contains ‘n’ pointers.

## Leaf Node:

- The Leaf Node of B+ Tree can contain atleast n/2 record and n/2 key values.
- Every Leaf Node of B+ tree contain one block pointer p to next leaf Node.

## Searching a record in B+ Tree

Suppose we have to search 55 in the below B+ tree structure. First, we will fetch for the intermediary node which will direct to the leaf node that can contain a record for 55.



So, in the intermediary node, we will find a branch between 50 and 75 nodes. Then at the end, we will be redirected to the third leaf node. Here DBMS will perform a sequential search to find 55.

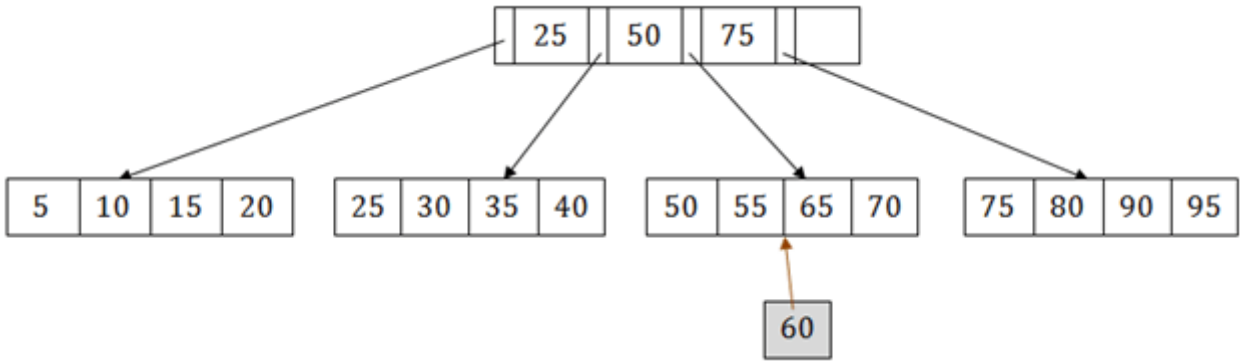
## B+ Tree Insertion:

Suppose we want to insert a record 60 in the below structure. It will go to the 3rd leaf node after 55. It is a balanced tree, and a leaf node of this tree is already full, so we cannot insert 60 there.

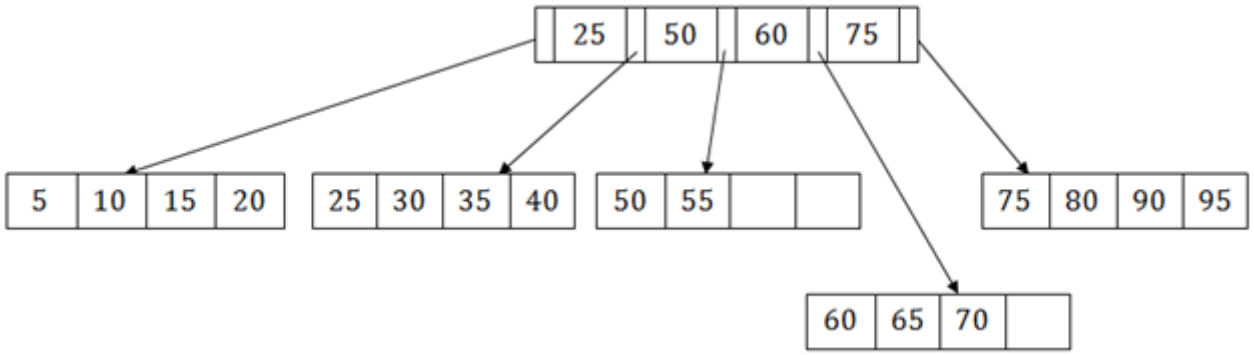
In this case, we have to split the leaf node, so that it can be inserted into tree without affecting the fill factor, balance and order.

The 3rd leaf node has the values (50, 55, 60, 65, 70) and its current root node is 50. We will split the leaf node of the tree in the middle so that its balance is not altered. So we can group (50, 55) and (60, 65, 70) into 2 leaf nodes.

If these two has to be leaf nodes, the intermediate node cannot branch from 50. It should have 60 added to it, and then we can have pointers to a new leaf node.



This is how we can insert an entry when there is overflow. In a normal scenario, it is very easy to find the node where it fits and then place it in that leaf node.




## Advantages of Indexing

Important pros/ advantage of Indexing are:

- It helps you to reduce the total number of I/O operations needed to retrieve that data, so you don't need to access a row in the database from an index structure.
- Offers Faster search and retrieval of data to users.
- Indexing also helps you to reduce tablespace as you don't need to link to a row in a table, as there is no need to store the ROWID in the Index. Thus you will able to reduce the tablespace.
- You can't sort data in the lead nodes as the value of the primary key classifies it.

## Disadvantages of Indexing:

Important drawbacks/cons of Indexing are:

- To perform the indexing database management system, you need a primary key on the table with a unique value.
- You can't perform any other indexes on the Indexed data.
- You are not allowed to partition an index-organized table.
- SQL Indexing Decrease performance in INSERT, DELETE, and UPDATE query.

~~~~~~~~~~~~~~~~~***ALL THE BEST***~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~