

UNIT - VI

(1)

Data On External storage:

- The Storage System in DBMS refers to the hierarchical arrangement of storage devices and media to store, Manage and Retrieve data efficiently.
- The system is designed to handle different storage capacities, access speeds, volatility, and costs.

Storage system hierarchy in DBMS:

The storage hierarchy typically has multiple levels each with its specific characteristics. Here's typical Hierarchy from fastest (and usually most expensive type) to slowest (and usually least expensive type)

1. Registers:

- located with the CPU
- smallest and fastest type of storage
- used to hold data currently being processed.

2. Cache Memory (L1, L2, L3 cache)

- on (or) very close to CPU
- extremely fast but small in size.
- Acts as buffer for frequently used data.

3. Main Memory (RAM):

- Data that's actively being used (or) processed is loaded here.
- Faster than Secondary storage.
- Volatile in Nature (i.e. data is lost when power is turned off)

4. Flash storage (Solid State Drives - SSD):

- No moving parts.
- Faster than Traditional hard drives.
- More durable and Reliable.

5. Magnetic Disks (Hard Disk Drives HDD):

- Primary Secondary storage medium
- Non-volatile, persistent storage.
- Data is stored in tracks, sectors and cylinders
- Slower than Main Memory but offers a large storage capacity at a lower cost.

6. Optical Disks (CD, DVD, Blu-Ray):

- Data Read using Lasers
- Slower than Magnetic disks and usually have less storage capacity
- Portable and commonly used for Media and software Distribution.

(3)

7. Magnetic Tapes:

- Sequential Access storage, unlike disks which are Random Access
- Often used for backups and Archiving due to their High capacity and Low cost.
- Much slower Access times compared to Magnetic Data.

8. Remote Storage / Cloud Storage:

- Data stored in Remote servers and Accessed over the Internet.
- provide Scalability, availability, and fault-tolerance.
- Latency Depends on Network Speed and Distance to servers.

Types of Storage:

1. Primary Storage:

Includes Registers, Cache Memory and Main Memory (RAM). It's Main Memory where the operating system, application programs, and data in current use are kept for quick Access by computer's processor.

2. Secondary Storage:

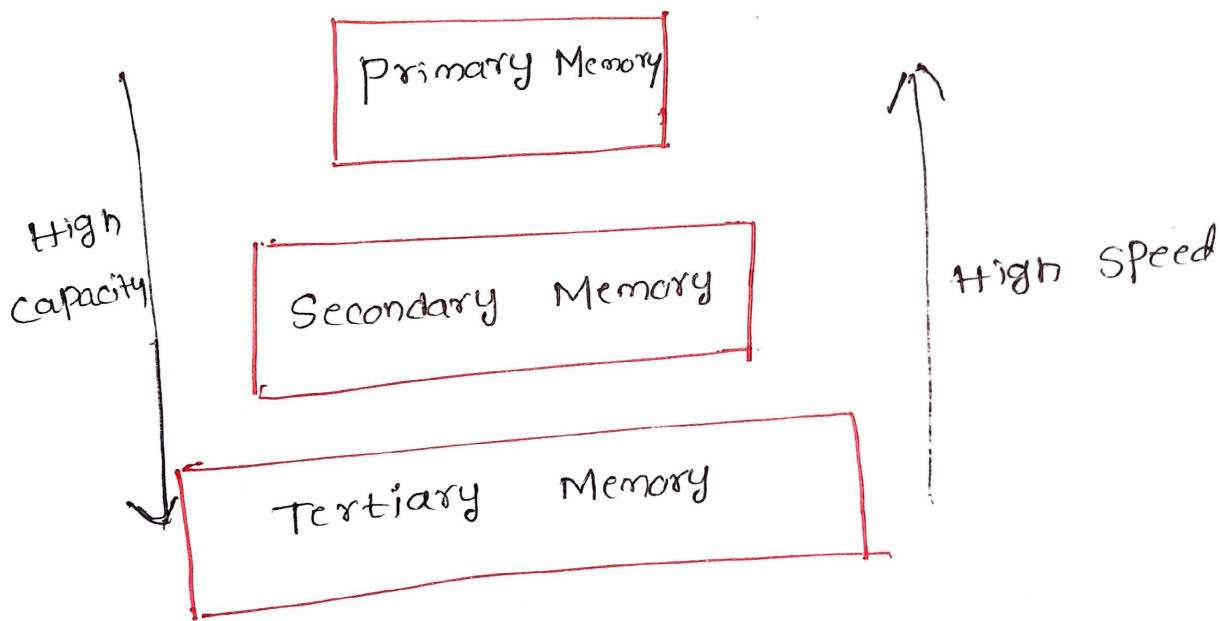
Encompasses Data Storage Devices like HDDs, SSDs, CDs and USB Drives. It is Non-Volatile and Retains data Even when the computer is turned off.

3. Tertiary storage (or) off-line storage :

often involves Magnetic tape system (or) Optical Disk archives. This is slower than Secondary storage and it is used for the data archiving and backup.

4. Quaternary Storage :

Refers to Methods like cloud storage (or) other remote storage techniques where data is stored in remote servers and is fetched over the internet (or) other networks.



File organization

(5)

→ File organization Refer to way in which data is stored in a file and consequently the Methods by which it can be Accessed

→ File Organization is a technique provided for the physical Designing of files. (Or)

→ A physical File Designing is an arrangement of data Records and its fields.

Objective of File organization:

- Records should be accessed as fast as possible.
- Insert, delete, update Transaction i.e operations on Record should be easy
- Record should be stored efficiently so that cost of storage is Minimal.

Four Basic ways (Or) Methods of Physically Organizing files on storage devices

1. Sequential File organization
2. Indexed - Sequential File organization
3. Direct-(Or) Hash File organization
4. Heap File(or) Un-ordered File organization.

Sequential File Organization:

⑥

→ In sequential File organization Records are arranged (or) store in the Ascending (or) Descending order According to Search key (or) key field.

→ There are Two ways to Implement Sequential File Organization

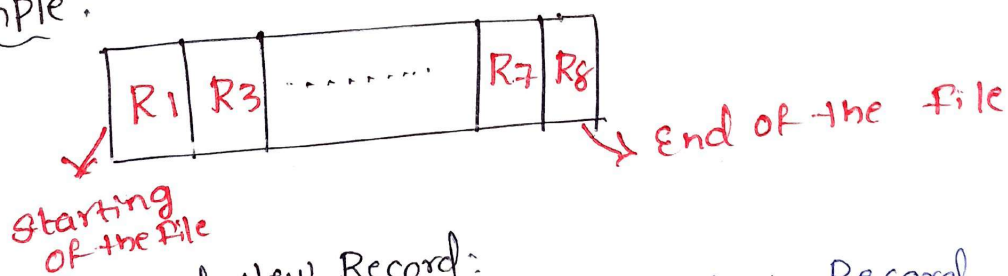
1. Pipe File Method
2. Sorted File Method

1. Pipe File Method:

→ In this Method we store the Records in a sequence i.e. one after another.

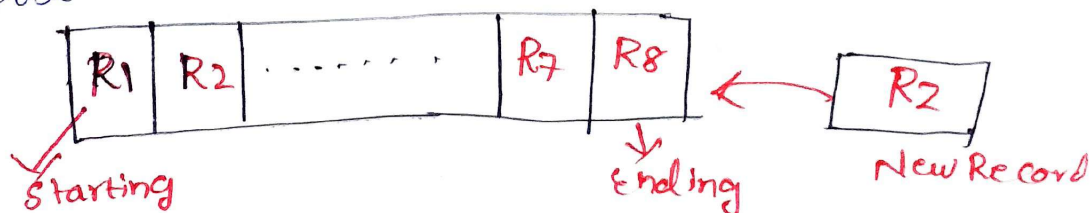
→ In case of updating (or) Deleting of any Record, The Record will be Searched in Memory Blocks. When it found Then it will be marked for Deleting and New Record is Inserted.

Example.



Inserting of New Record:

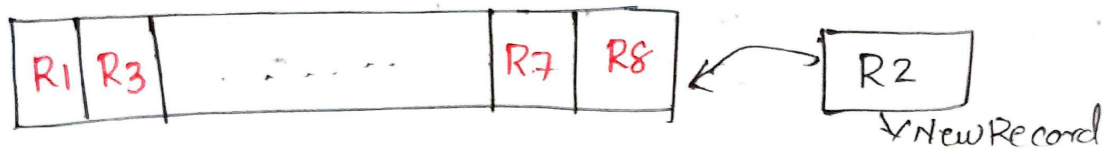
SUPPOSE we want to insert a New Record R2 in Sequence



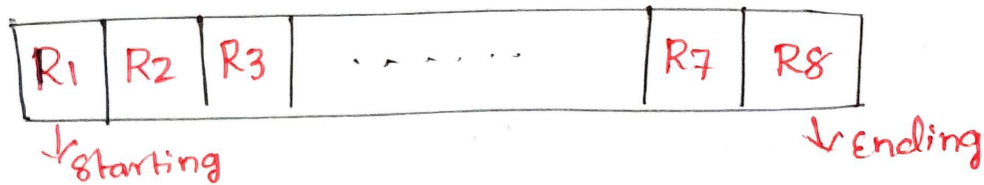
① Sort & file organization

→ In this Method The New Records is always inserted at file and then it will sorted the sequence in Ascending (or) Descending order. Sorting of Records is based on any Primary Key (or) any other key.

→ Suppose we want to insert a New Record 'R2' in the sequence



Then



② Indexed Sequential File organization

→ Indexed Sequential File organization is an advanced Sequential file organization Method, In this case Records are stored in file with the help of primary key.

→ For Each primary key an index value is created and mapped to the Record. The index contains the address of the Record in the file.

→ It provides facilities for accessing Records Both Sequential and Directly

→ A simplified version of how indexed-sequential Access operations is shown in figure

Cylinder NO	Highest Key
1	54
2	75
3	90

Cylinder Index

Track Index for Cylinder 1

Track NO	H-Key
1	18
2	38
3	54

Track Index for Cylinder 2

Track NO	H-Key
1	63
2	70
3	75

Track Index for Cylinder 3

Track No	H-Key
1	80
2	85
3	90

Hash

② Hash File Organization: (or) Direct File Organization.

→ Hash file organization is also known as ~~or~~ Direct file organization.

→ In Direct file organization, all the Records are stored in Direct Access Storage Device (DASD) Such as Hard Disk.

→ Records are Randomly Placed through out the file.

→ This file organization is useful for immediate access to Large Amount of information.

→ In this Method, Data is stored at the Data blocks where address is Generated by using Hash function.

→ The memory location where these Records are stored is called Buckets.

→ The Data ^{to} Bucket is just a "unit of storage" that can store one or more Records.

Hash function:

It is mathematical function which is used to Generate address, and used to locate Records for access, Insert & Delete.

$$h(n) = n \% m$$

↓
Input

↘ Array Size

Example

input
21
19
35

Hashing $h(m) = n \% m$
$h(21) = 21 \% 10 = 1$
$h(19) = 19 \% 10 = 9$
$h(35) = 35 \% 10 = 5$

Array [10]	
Bucket/ Index	Link
[0]	→ [21] x
[1]	
[2]	
[3]	
[4]	→ [35] x
[5]	
[6]	
[7]	
[8]	
[9]	→ [19] x

Types of Hashing:① Static Hashing:

In which the size of the Bucket is fixed.

② Dynamic Hashing:

In which the size of the Bucket is Not fixed.

④ Heap file organization:

→ Heap file organization is also known as un-ordered file organization.

→ In this method the Records are added in memory Data Blocks, in no Particular order.

→ The Records are not Sorted and not Stored in Consecutive data Blocks in memory.

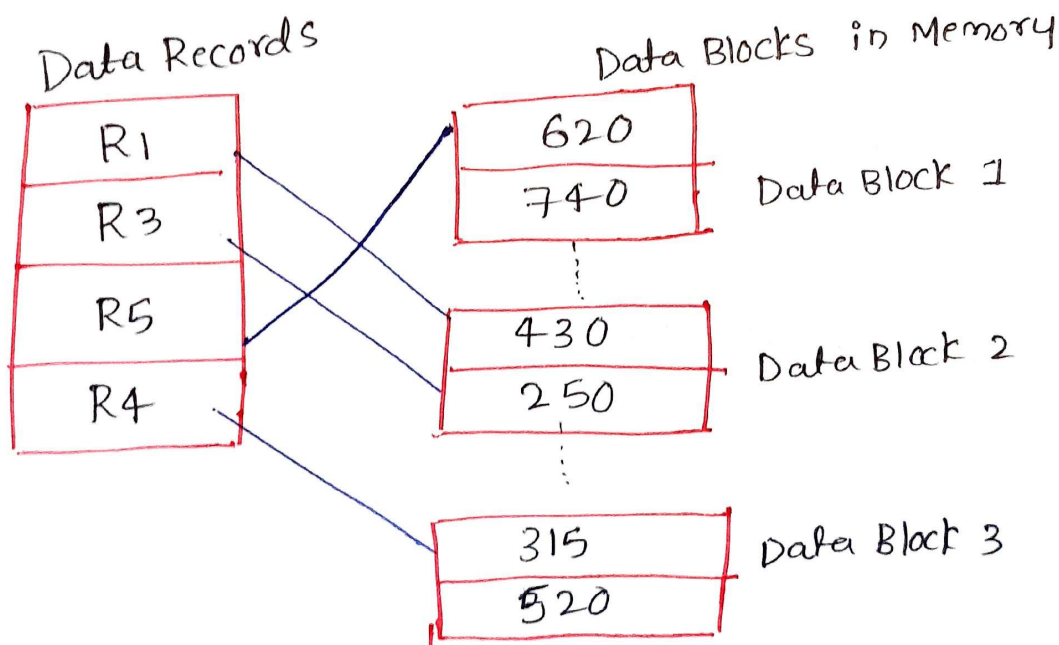
→ The Heap File organization New Records are inserted at the end of the file.

→ If the Data Block ~~full~~, is full, The New Record is stored in some other Block. Here the other ~~last~~ Data Block need ~~to~~ Not be the very next data Block but it can be any Block in the Memory.

→ It is the Responsibility of DBMS to store and manage the New Records.

→ Searching a Record is time Consuming process in this Method.

→ Update and Delete Operations also give Poor Performance as the Records need to be searched first for update and Delete.



Indexing!

→ Indexing is used to optimize the performance of Database by minimizing the number of disk accesses required when query is processed.

→ The index is a type of Data structure. It is used to locate and access the data in a Database table quickly.

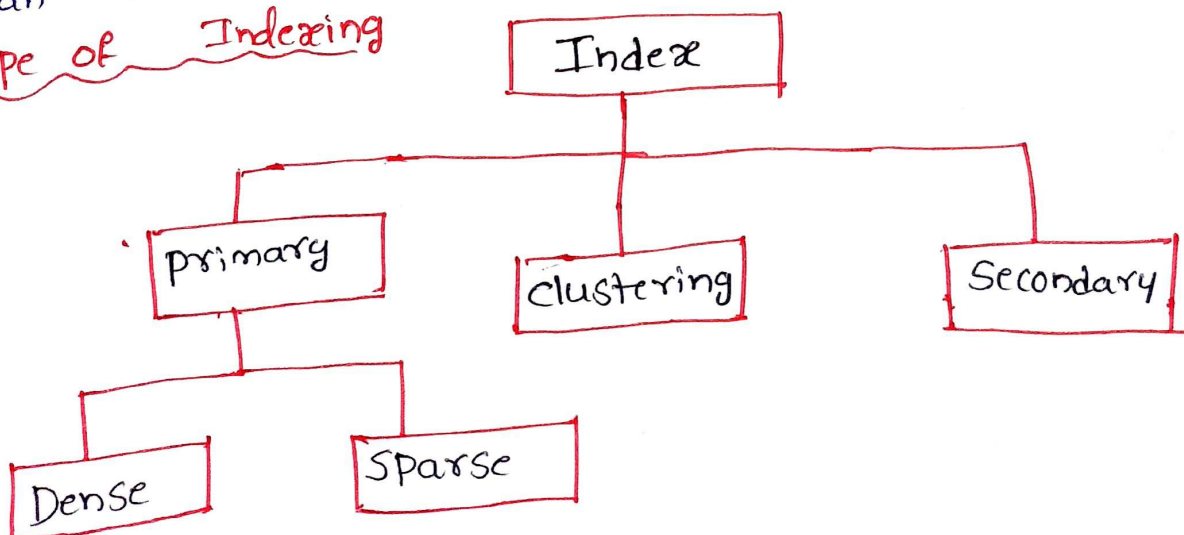
Structure of Index:

Search Key	Data Reference
------------	----------------

→ The first column of Database is the search key that contains a copy of the primary key (or) candidate key of the table. The values of primary key are stored in sorted order so that corresponding data can be accessed easily.

→ The second column of Database is the Data Reference. It contains a set of pointers holding the address of disk block where the values of particular key can be found.

Type of Indexing



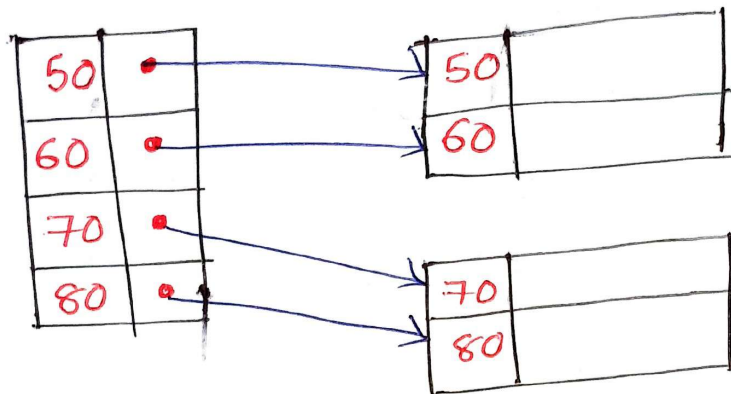
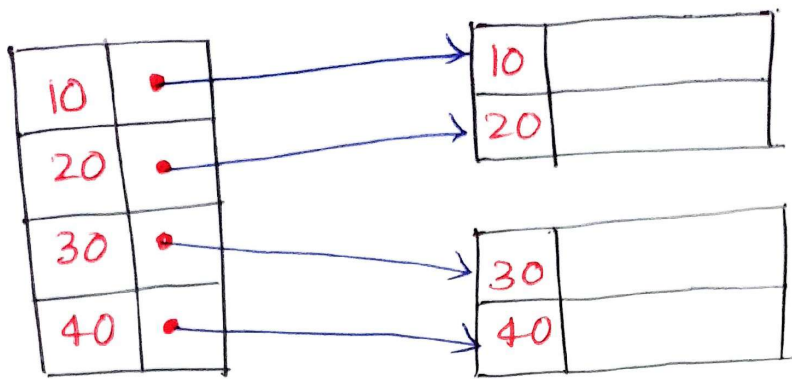
Ordered file	Primary Index	Clustered Index
Unordered file	Secondary Index	Secondary Index

Primary Indexing:

- Primary Index is an ordered file which is fixed length size with two fields
- The first field is the same primary key and second field is pointed to that specific data Block.
- In the primary key index, there is always one to one Relationship between the entries in the index table.
- The primary Indexing in DBMS is also further divided into types.
 1. Dense Index
 2. Sparse Index

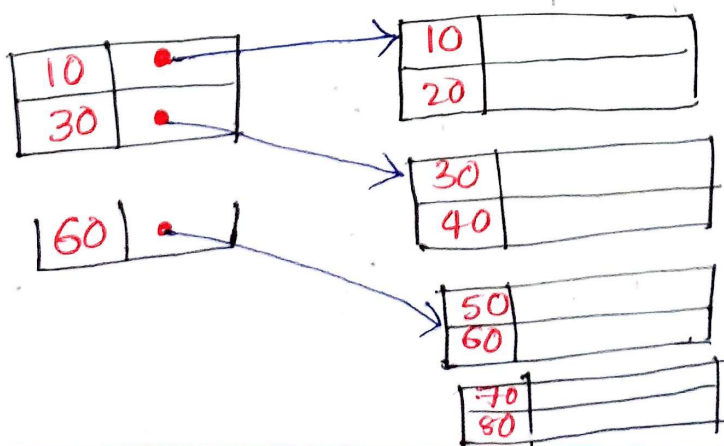
① Dense Index:

- For each and every key
- 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.



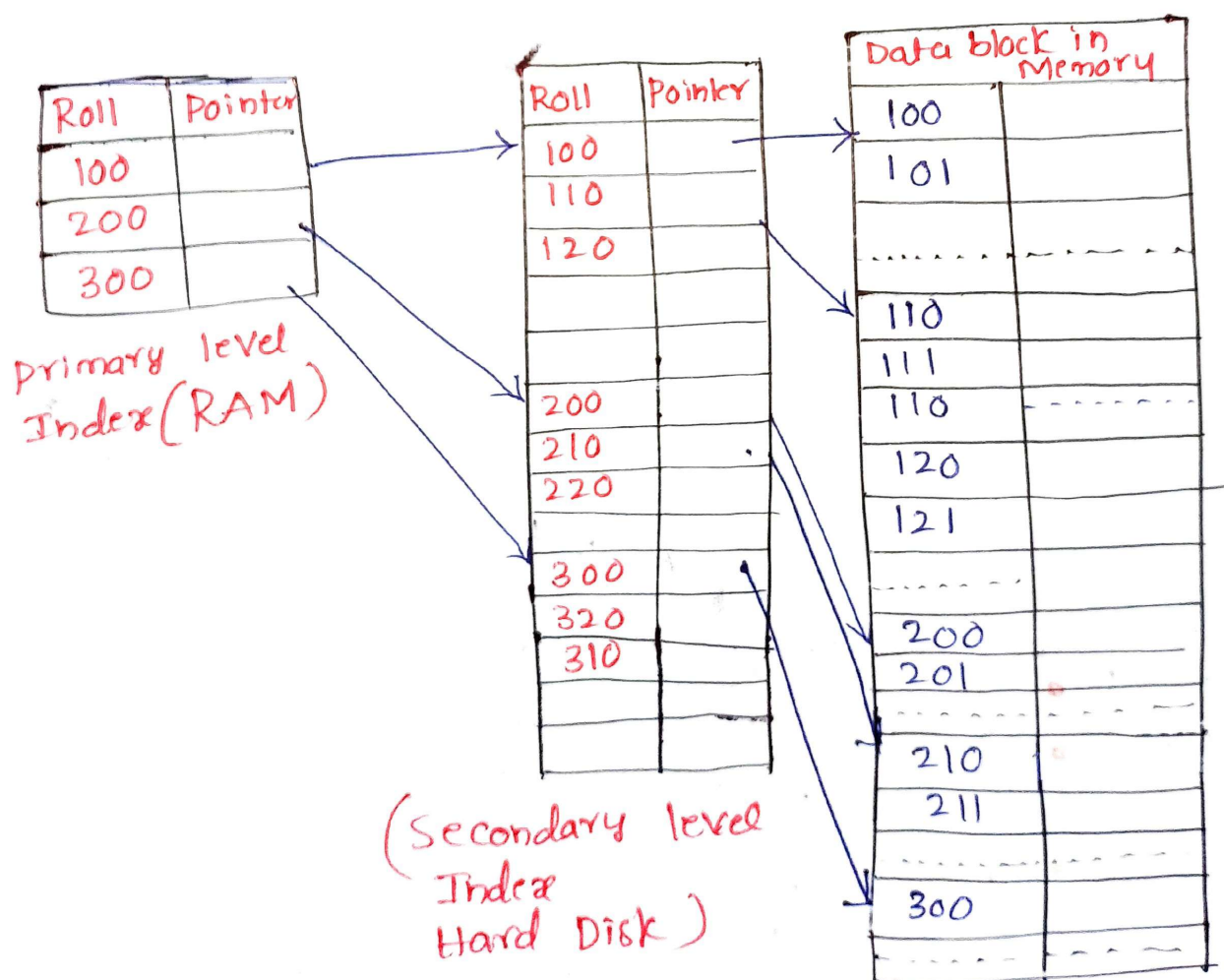
Sparse Index:

- It is an index Record that appears for only some of the values in the file.
- However, Sparse Index stores Index records for only some search-key values.
- It Needs less space, less maintenance overhead for insertion, and Deletion but It is slower compared to the Dense Index for Locating Records



Secondary Index:

- In Secondary Indexing, to Reduce the size of mapping, another level of Indexing is introduced.
- In this method, the huge range of for the columns is selected initially so that the mapping size of the first level becomes small.
- Then Each range is further divided into smaller ranges.
- The Mapping of the first level is stored in the Primary memory, so that Address fetch is faster.
- The mapping of second level and level and actual data are stored in the Secondary Memory (Hard Disk).



Clustering Index:

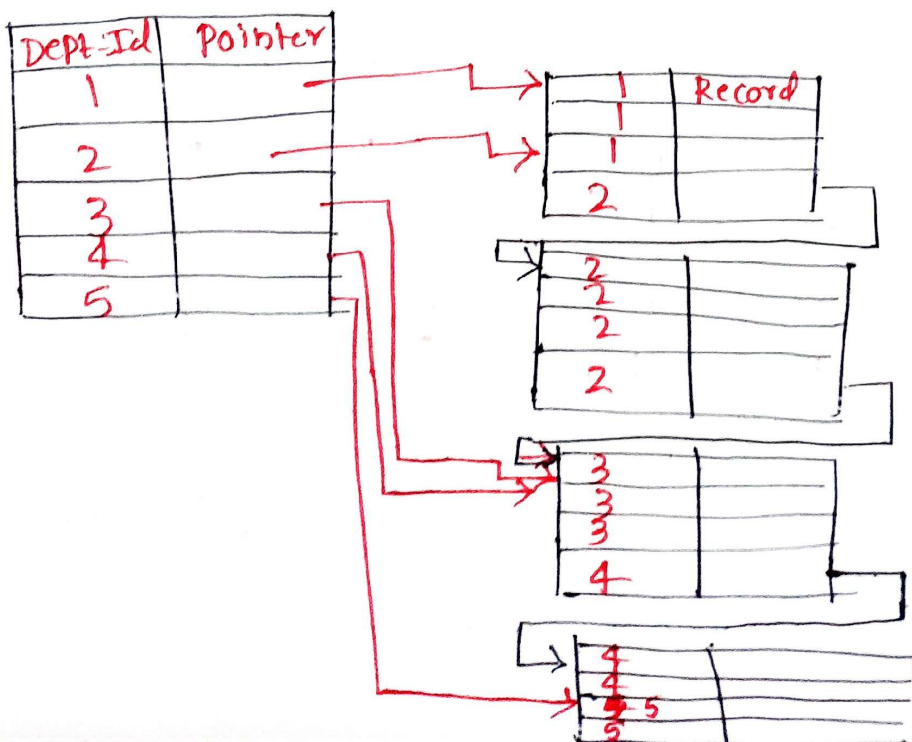
→ Suppose

→ A clustered Index can be defined as an ordered Data File, Some times the index created on Non-primary key column which may not be unique for each Record.

→ In this case, to identify the record faster, we will group two (or) more columns to get the unique value and created Index out of them. This Method is called a Clustering Index.

→ The Records which have similar characteristics are Grouped and Indexes are created for these Groups.

→ Suppose a company contains several Employee in each Department. Suppose we use clustering index, where all Employee which belong to the same Department ID are considered within a single cluster, and index pointers point to the cluster as whole. Here Dept-Id is a Non unique Key.



Index Data Structures:

(16)

→ Indexes in DBMS are Data Structures that improve the speed of data Retrieval operations on Database table at the cost of additional storage space and Maintenance Over Head.

→ Most commonly used Index data structure in DBMS are

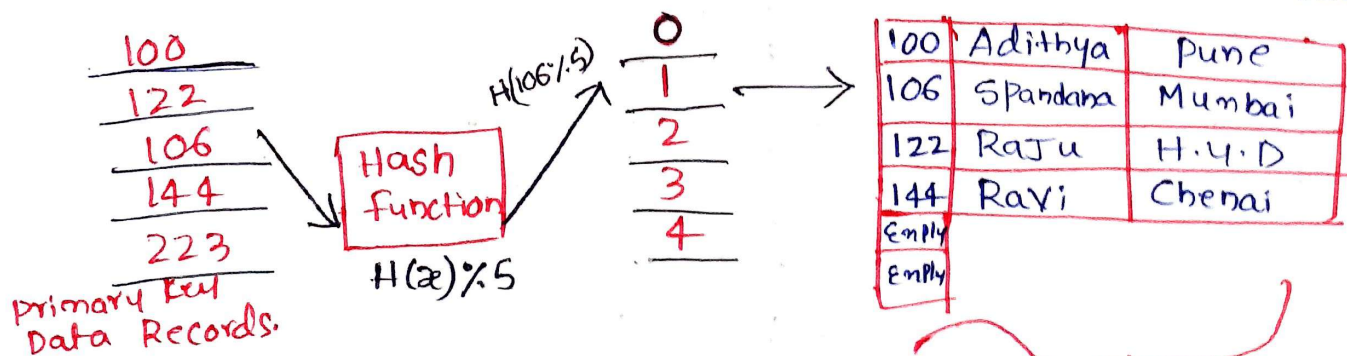
1. Hash Based Indexing
2. Tree Based Indexing

1. Hash Based Indexing:

→ we can organize Records using a Technique called Hashing to quickly find Records that have given Search Key value.

→ In this approach, the Records in a file are grouped in Buckets, where a Bucket consists of Primary Page and possibly, additional pages linked in a chain.

→ The Bucket to which a Record belongs to can be determined by applying a special function, called a Hash Function to the Search Key.



$$\rightarrow H(106 \% 5) = 1$$

This indicates that data Record should be **Actual Data Record** Placed (or) Searched in 1st Bucket (or 1st Hash Index)

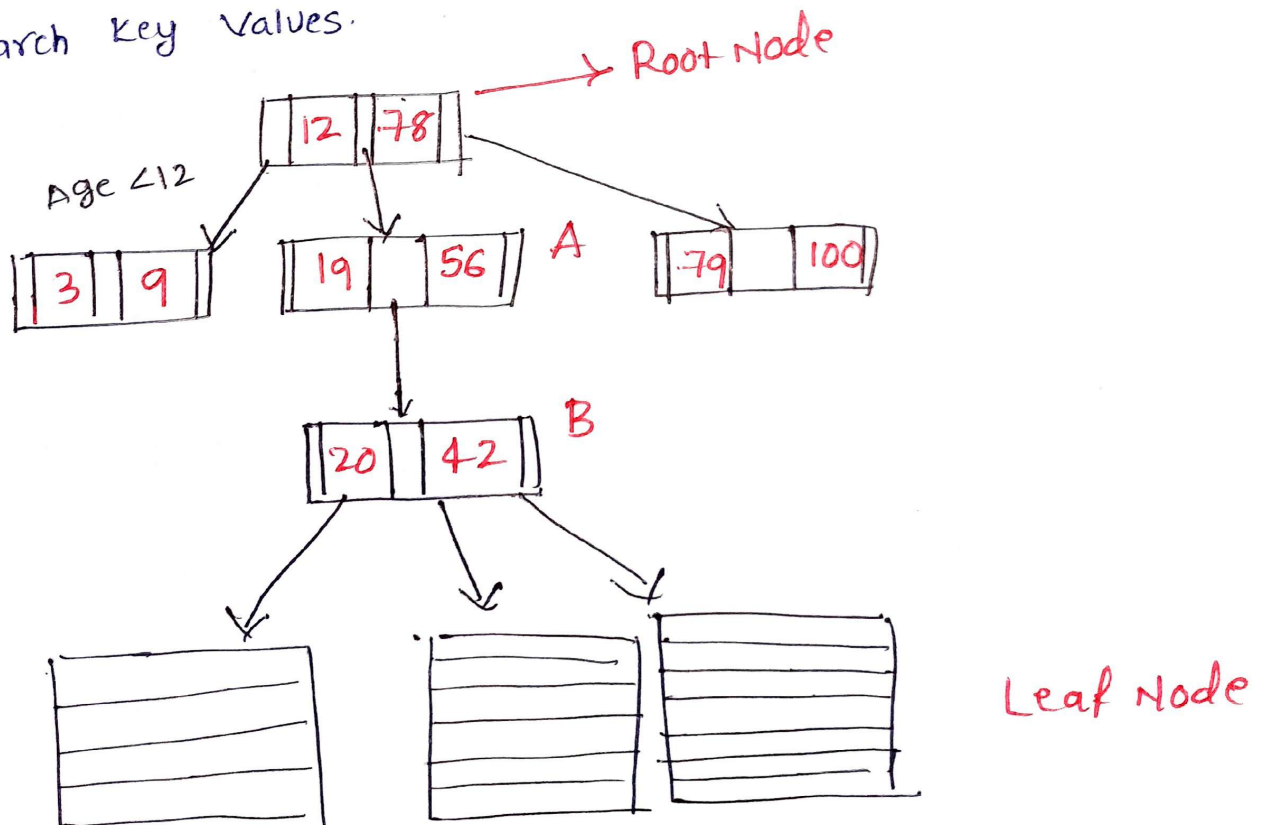
2. Tree Based Indexing :

→ An Alternative to hash-based indexing is to organize Records using a tree like data structure.

→ The Data Entries are arranged in sorted order by Search Key Value, and hierarchical Search Data structure is maintained that directs Searches to the correct Page of Data Entries.

→ The lowest level of the tree, called the Leaf Level contains the Data Entries.

→ Non-leaf pages contain Node pointers separated by Search Key Values.



Comparison of file organizations :

(18)

File Organization	Description	Advantages	Disadvantages	Use cases
Heap (unordered)	Records are stored in the order of they arrive, with no specific sequence	Simple to Implement - Efficient for bulk loading of data	Slow for searching specific records Not efficient for updates	Suitable for small datasets (or) Temporary tables
Sequential	Records are stored sequentially by the value of key field	- Fast sequential access - Recognition Required Suitable for Range Queries	- Costly to insert (or) Delete Records Recognition Required	Used when sequential access is needed
Hashing	Records are stored on a Hash Function applied to key field	- Fast for exact match searches	- Inefficient for range queries - collisions need handling	Used for unique key lookups like user Id's
clustered	Related Records are stored together based on a clustering key	- Improves performance of Related queries - Reduces I/O operations	- Complex to maintain - Requires careful choice of clustering key	- Suitable for Related data queries eg: by region
Index Sequential (ISAM)	Records are sequentially organized, with an index for fast searches	- combine fast sequential and Random Access - Good for Range Queries	- Index Maintenance Over Head - Insertion, Deletions may require splitting.	Common in OLAP systems for Reporting and analytics
B ⁺ Tree	A balanced tree structure where all leaf nodes are at the same level.	- Efficient for search, - Insert, Delete operations - support range queries	- Requires additional storage for tree maintenance	Suitable for Scenarios with frequent updates and lookups.

Indexed Sequential Access Methods (ISAM)

Creation of ISAM Structure:

→ It follows the Bottom up approach

— Steps to be followed for creating ISAM Tree structure

1. Sort the data file on the Search Key field.
2. Create the index pages for Leaf level.
3. If there are more than one index pages, Repeat until the top most index level (root) contains only one page.

Given Records

15	10	27	20	37	33	46	40	55	52	97	63
----	----	----	----	----	----	----	----	----	----	----	----

Sorted Records

10	15	20	27	33	37	40	46	52	55	63	97
----	----	----	----	----	----	----	----	----	----	----	----

Let us Assume node size of ISAM tree is "2"

Step 1

10	15	20	27	33	37	40	46	52	55	63	97
----	----	----	----	----	----	----	----	----	----	----	----

↳ Step 1
Data Pages

Step 2

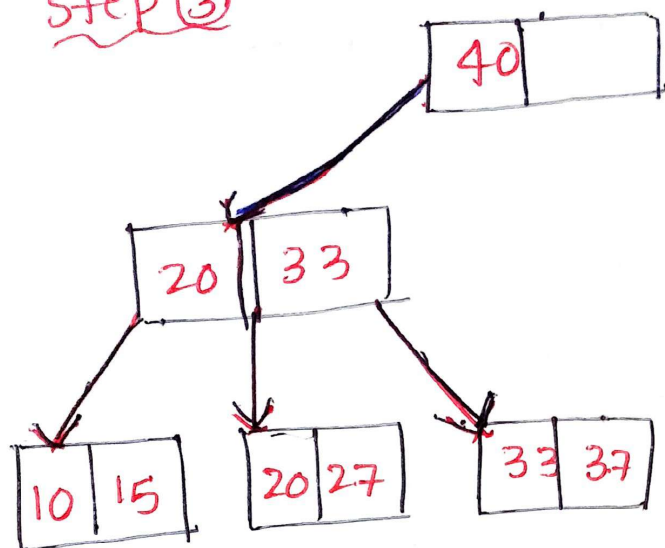
20	33	52	63
----	----	----	----

→ Step 2
Index Pages

10	15	20	27	33	37	40	46	52	55	63	97
----	----	----	----	----	----	----	----	----	----	----	----

↳ Step 1
Data Pages

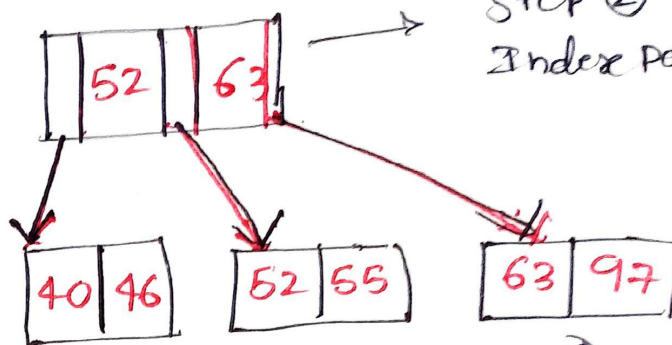
Step ③



Root

Step ③

Step ②
Index Pages



Step ①
Data Pages

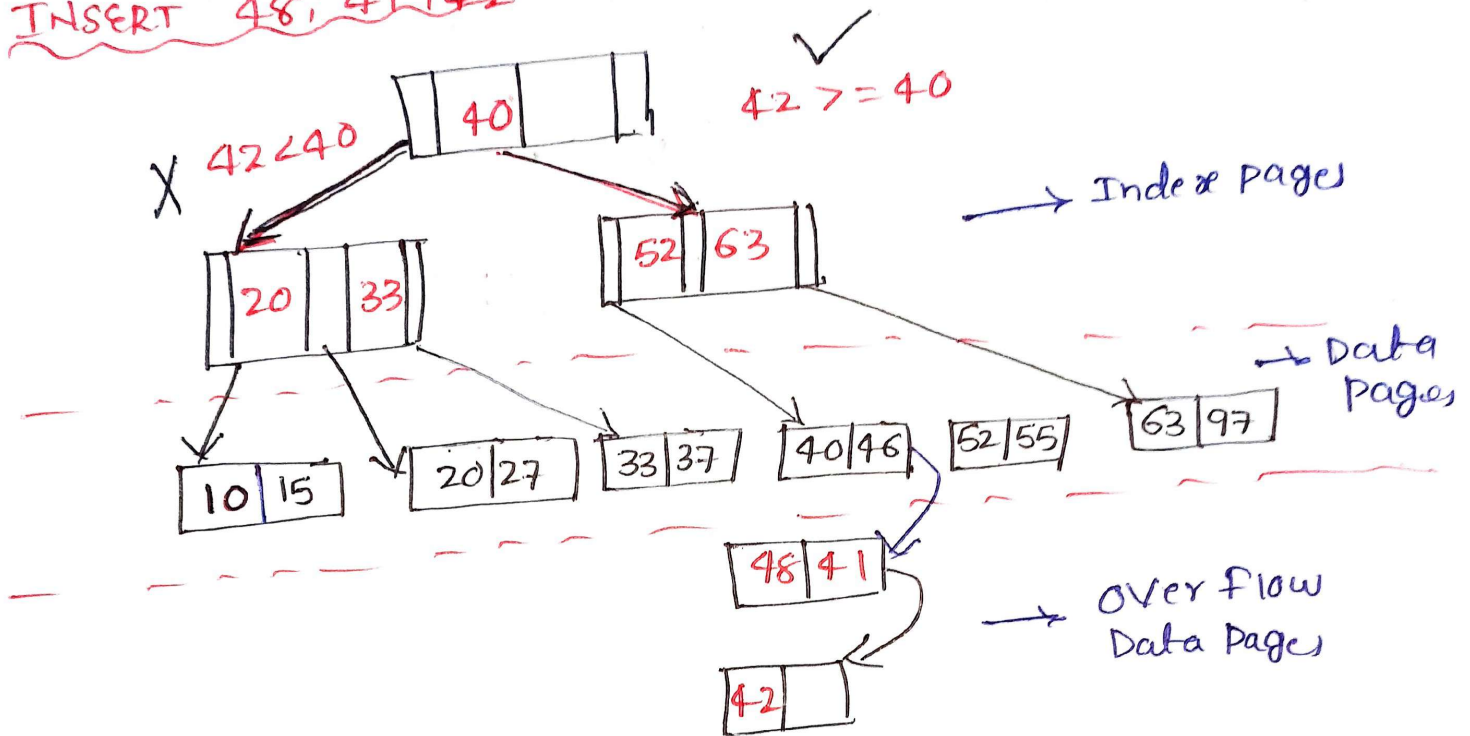
Search operation in ISAM

1. Start at the Root index Node
2. Use Key Comparisons with the Search Keys in the Root index Node.
3. If Search Key is less than index key. Follow the Left Pointer pointing to the left child. If it is Greater than (or) Equal to index key, follow the Right Pointer pointing to the right child.
4. Continue this Process at each subsequent index Node until you reach a leaf Node.
5. Search with in the corresponding data Page for the target Data.

Insertion Operation in ISAM

1. If there is space in corresponding primary data page, insert Record in that page
2. Else overflow data page needs to be added

INSERT 48, 41, 42

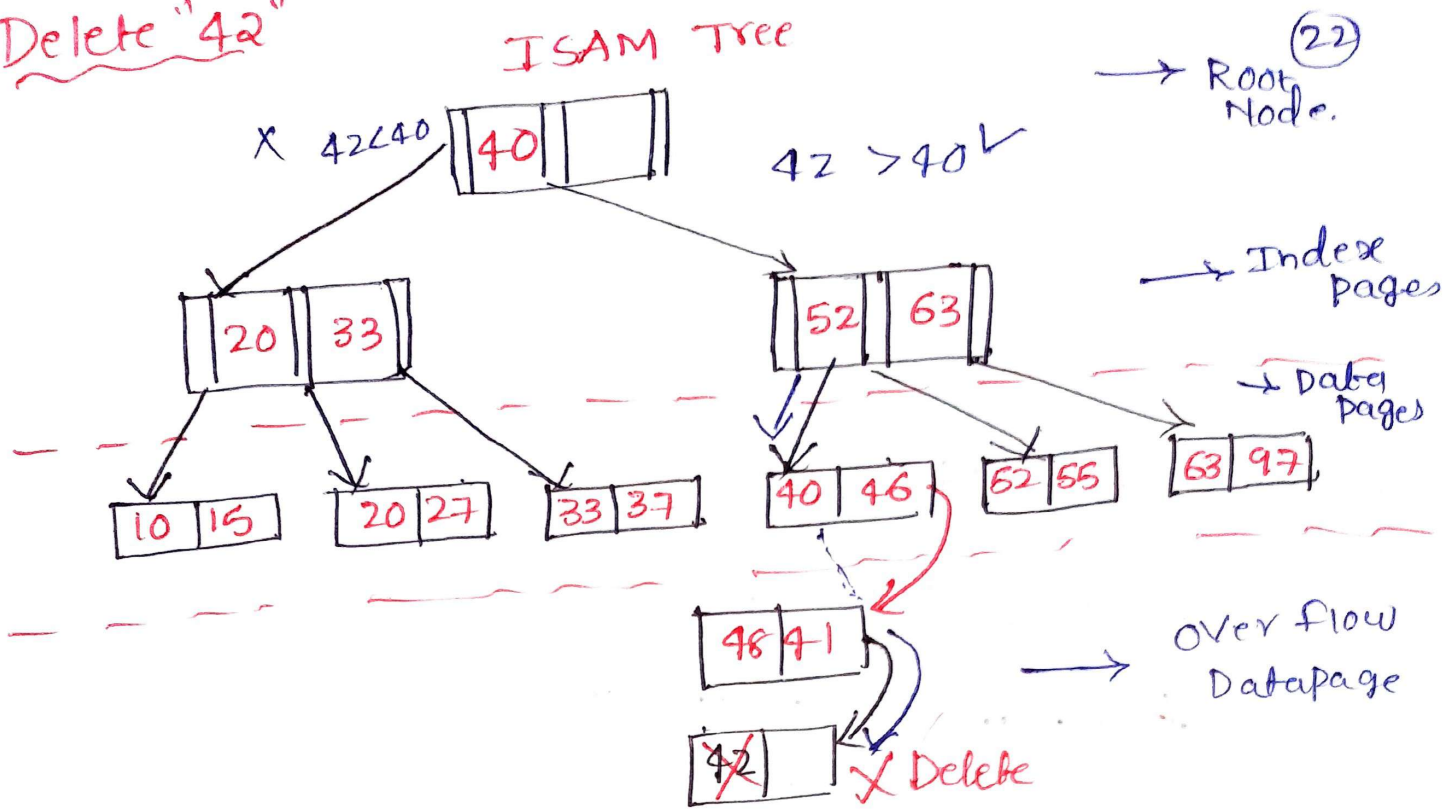


Deletion Operation in ISAM

1. Simply Remove the Record from corresponding data pages.
2. If the Removal makes an overflow data page empty, Remove that overflow data page.
3. If the Removal makes a primary data page empty, keep it as a placeholder for future insertions.
4. Don't move Records from overflow data pages to primary data pages

Delete "42"

ISAM Tree



Advantages & Dis-advantages of ISAM:

Advantages:

It performs well when data does not change frequently and there are no large overflow chains.

Dis advantages:

When more records are inserted - Non-sequential overflow chains may be created, which degrades search performance.

B+ Tree (A Dynamic Index Structure - Insertion)

B+ Tree ÷

- The B+ Tree file organization is a method to organize a file in DBMS.
- The Records are sorted using the primary key to conform with Key-Value pair rule.
- It is in a tree-like structure.
- B+ Tree file organization is similar to the BST (Binary Search Tree)
- But it can ~~be~~ have more than two children, The Leaf Node of B+ Tree file organization contain all the Records and middle nodes contain only pointers to the Leaf Node.
- It Does n't carry any Records.

Rules:

1. Data is present in leaf Nodes only.
2. Tree can be Left biased / Right biased.
3. Left sub-tree must be less than Root Node and Right Subtree will be Greater than Root Node.
4. All the Leaf Nodes connected with Linked list

Note ÷

If Order of Tree is "m" then

Max children = m

Min children = $\lceil m/2 \rceil$

Max Keys = m-1

Min Keys = $\lceil m/2 \rceil - 1$

B+Tree - Insertion

Example

Insert 1, 4, 7, 10, 17, 21, 31, 25, 19, 20, 28, 42 of order "4"

Sol:

order = 4

Maximum children = 4

Minimum children = $\lceil \frac{n}{2} \rceil = \frac{4}{2} = 2$

Max keys = $m - 1 \Rightarrow 4 - 1 = 3$

Min keys = $\lceil \frac{n}{2} \rceil - 1 \Rightarrow 2 - 1 = 1$

