# UNIT-V
# OVERVIEW OF STORAGES AND INDEXING

Data on External Storage- File Organization and Indexing – Clustered Indexing – Primary and Secondary Indexes, Index Data Structures.

## 5.1 Data on External Storage:

- A DBMS stores vast quantities of data, and the data must persist a cross program executions. Therefore, data is stored on external storage devices such as disks and tapes, and fetched into themain memory as needed for processing.
- The unit of information read from or written to disk is a page. Disks are the most important externalstorage devices. They allow us to retrieve any page at a (more or less) fixed cost per page. However,if we read several pages in the order that they are stored physically, the cost can be much less than the cost of reading the same pages in random order.
- Tapes are sequential access devices and force us to read data one page after the other. They aremostly used to archive data that is not needed regularly.
- Each record in a file has a unique identifier called a record id, or rid for short.
- A rid has the property that we can identify the disk address of the page containing the record byusing the rid.

## Different Kind of memory in a Computer System:

**1)Primary storage:**

Primary storage that has cache, flash and main memory to provide very fast access to the data.

i. Cache memory
ii. Main memory
iii. Flash Memory

## 2)Secondary storage:

The secondary storage devices are Magnetic disks that are slower and permanent devices.

iv. Magnetic disks

v. Optical disks.

## 3)Tertiary storage:

The Tertiary Storage is a permanent and slowest device when compared with Magnetic disk.

vi. Magnetic Tape

**i. Cache memory:** The cache is the fastest but costliest memory available. It is not a concern for databases.

**ii. Main Memory:** The processor requires the data to be stored in the main memory. Although mainmemory contains a Giga byte of storage capacity it is not sufficient for databases.

**iii. Flash Memory:** Flash memory stores data even if the power fails. Data can be retrieved as fast as in main memory, however writing data to flash memory is a complex task and overwriting data cannot be donedirectly. It is used in small computers.

**iv. Magnetic Disk Storage:** Magnetic disk is the permanent data storage medium. It enables random accessof data and it is called "Direct-access" storage. Data from the disk is transferred into the main memory for processing. After modification, the data is loaded back onto the disk.

**v. Optical Disk:** Optical disks are Compact Disks (CDs), Digital Video Disks (DVDs). These are commonlyused for permanent data storage. CD s have a larger capacity that is up to 640 MB. These are relatively cheaper. To storage, a large volume of data CDs is replaced with DVDs. DVDs are in various capacities based on manufacturing.

## Advantages of Optical disk:

1. Optical disks are less expensive.
2. a large amount of data can be stored.
3. CDs and DVDs are longer durability than magnetic disk drives.
4. These provide nonvolatile storage of data.
5. These can store any type of data such as text data, music, video etc.

## vi. Tape Storage (or) Tertiary Storage media:

- Tape (or) Tertiary storage provides only sequential access to the data and the access to data is muchslower. They provide high capacity removable tapes. They can have a capacity of about 20 Gigabytes to 40 GB. These devices are also called "tertiary storage" or "off the storage".
- In a larger database system, tape (tertiary) storage devices are using for backup storage of data.
- Magnetic tapes are fragmented into vertical columns referred to as frames and horizontal rowsreferred to as tracks.

## Advantages:

1. Magnetic tapes are very less expensive and durable compared with optical disks.
2. These are reliable and a good tape drive system performs a read/write operation successfully.
3. These are a very good choice for archival storage and any number of times data can be erased and reused.

## Disadvantages:

The major disadvantage of tapes is that they are sequential access devices. They work very slow when compared to magnetic disks and optical disks.

## 5.2 File Organization and Indexing:

**File Organization:**

- A file is organization is a method of logically arranging the records in a file on the disk.
- Data is organized on secondary storage in terms of files. Each file has several records, These recordsare mapped onto disk blocks.
- Thus, a file organization can be defined as the process of arranging the records in a file, when the fileis stored on disk.

- The enormous data cannot be stored in the main memory so, it is stored on a magnetic disk. Duringthe process, the required data can be shifted to the main memory from the disk. The unit of information being transferred b/w main memory and disk is called a page.
- Each record on a file is identified by record id or rid. Whenever a page needs to be processed, thebuffer manager retrieves the page from the disk-based on its record id.
- Disk space manager is software that allocates space for records on the disk. When DBMS requires additional space, it calls disk space management to allocate the space. DBMS also informs the diskspace manager when it's not going to use the space

## Types of File Organizations:

The most widely used file organizations are:

## 1.Heap/Unordered file:

The simplest file structure is an unordered file or heap file. Records in a heap file are stored in random order across the pages of the file. A heap file organization supports retrieval of all records or retrieval of a particular record specified by its rid; the file manager must keep track of the pages allocated for the file.

- **Inserting a record:** Records are inserted in the same order as they arrive.
- **Deleting a record:** The record is to be deleted, first access that record and then marked as deleted.
- **Accessing a Record:** A linear search is performed on the files starting from the first record until the desired record is found.

## 2.Sequential/Ordered File:

Files are arranged in sequential order. The main advantage of this file organization is that we can now use binary search as the file are sorted.

- **Insertion of Record:** This is a difficult task. Because first we need to identify the space where the record need insert and the file is arranged in an order. If the space is available then a record can directly be inserted. If space is not sufficient, then that record moved to the next page.
- **Deletion of Files:** This task is also difficult to delete the record. In this first find deleted record andthen remove the empty space of deleted record.
- **Access of files:** This is similar as we can use binary search on files.

## 3.Hash files:

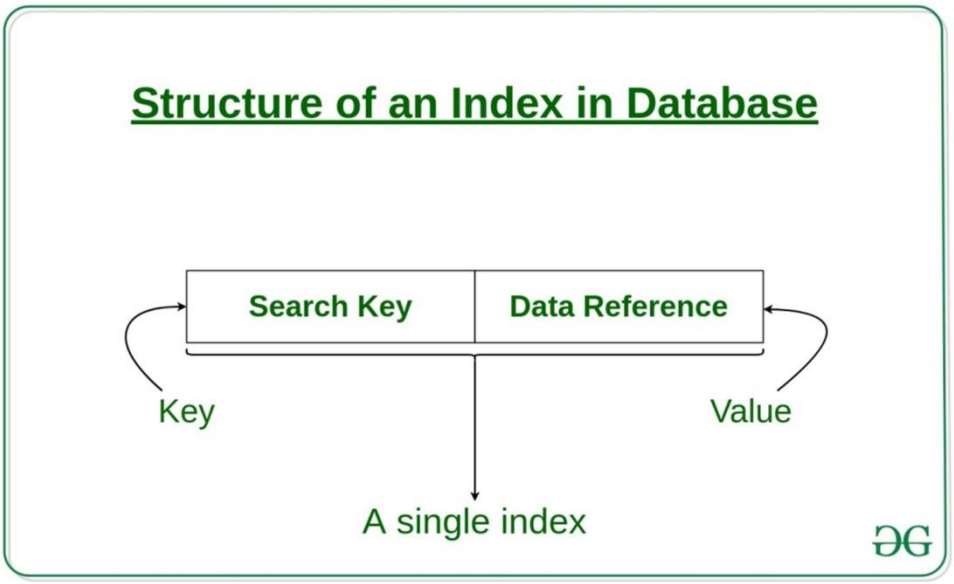
Using this file organization, files are not organized sequentially, instead they are arranged randomly. The address of the page where the record is to be stored is calculated using a 'hash function'.
All the Insertion, Deletion and searching In a hash file is based upon the Hash function itself.

## 4. Indexing:

An index is a data structure that organizes data records on disk to optimize certain kinds of retrieval operations. An index allows us to efficiently retrieve all records that satisfy search conditions on the search key fields of the index.

We can also create additional indexes on a given collection of data records, each with a different search key, to speed up search operations that are not efficiently supported by the file organization used to store the data records. we use the term data entry to refer to the records stored in an index file.

A data entry with search key value *k*, denoted as *k**, contains enough information to locate (one or more) data records with search key value *k*. We can efficiently search an index to find the desired data entries, and then use these to obtain data records.




There are three main alternatives for what to store as a data entry in an index:

1. A data entry h is an actual data record (with search key value k).
2. A data entry is a (k, rid) pair, where rid is the record id of a data record with search key value k.
3. A data entry is a (k. rid-list) pair, where rid-list is a list of record ids of data records with search key value k.

### 1. Alternative 1:

A data entry is an actual data record (with search key value *k).*

| shape | colour | holes |
|---|---|---|
| round | red | 2 |
| square | red | 4 |
| rectangle | red | 8 |
| round | blue | 2 |
| square | blue | 4 |
| rectangle | blue | 8 |

In alternative (1), each entry b is a data record with search key value *k.* We can think of such an index as a special file organization. Such an indexed file organization can be used instead of, for exarnple, a sorted file or an unordered file of records.

### 2. Alternative 2(k,rid):

A data entry is a *(k, rid)* pair, where *rid* is the record id of a data record with search key value *k.*

File with data records

| shape | colour | holes |
|---|---|---|
| round | red | 2 |
| square | red | 4 |
| rectangle | red | 8 |
| round | blue | 2 |
| square | blue | 4 |
| rectangle | blue | 8 |

Index File with 6 data entries

| colour | location |
|---|---|
| red | 1 |
| red | 3 |
| red | 2 |
| blue | 6 |
| blue | 4 |
| blue | 5 |

### 3. Alternative 3(k,rid-list):

A data entry is a *(k, rid-list)* pair, where *rid-list* is a list of record ids of\ data records with search key value *k.*

File with data records

| shape | colour | holes |
|---|---|---|
| round | red | 2 |
| square | red | 4 |
| rectangle | red | 8 |
| round | blue | 2 |
| square | blue | 4 |
| rectangle | blue | 8 |

Index File with 6 data entries

| colour | locations |
|---|---|
| red | 1,2,3 |
| blue | 4,5,6 |

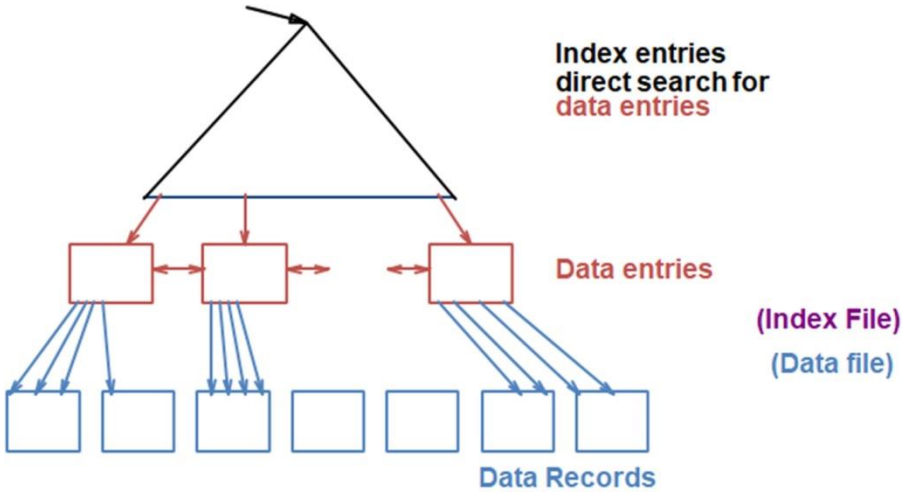
- Alternatives (2) and (3), which contain data entries that point to data records, are independent of thefile organization that is used for the indexed file (i.e., the file that contains the data records).
- Alternative (3) offers better space utilization than Alternative (2), but data entries are variable inlength, depending on the number of data records with a given search key value.

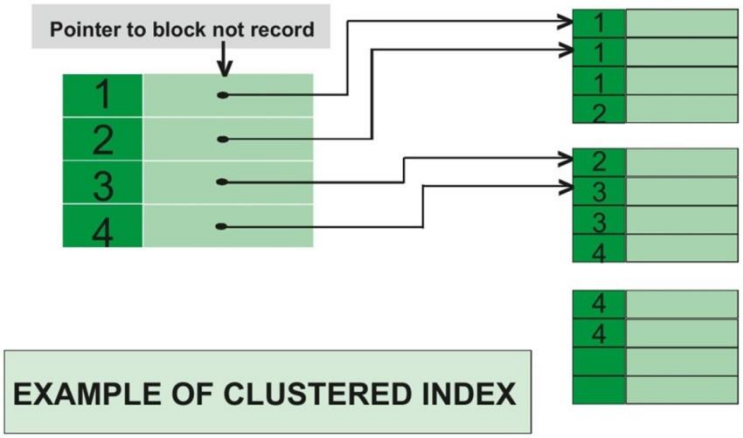
## 5.3 Clustered Indexing:

**1)Clustered Index**
**2)Un-Clustered Index**

### 1)Clustered Index:

- When a file is organized so that the ordering of data records is the same as or close to the ordering ofdata entries in some index, we say that the index is clustered.

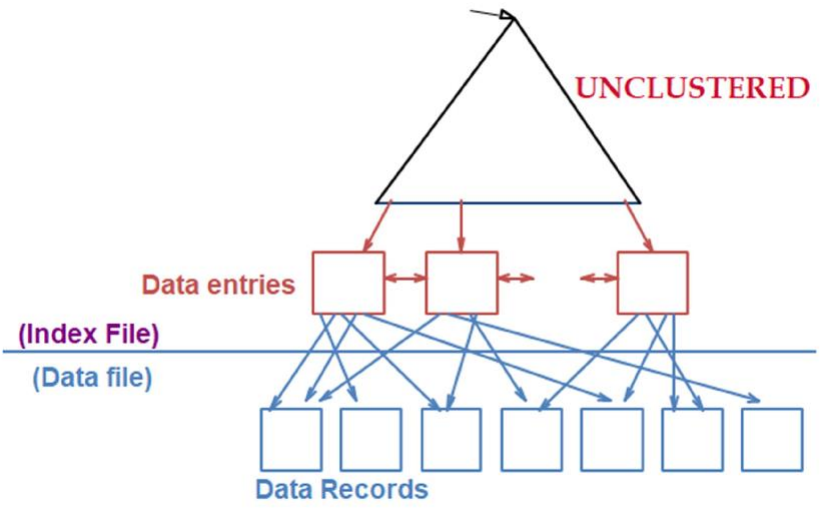


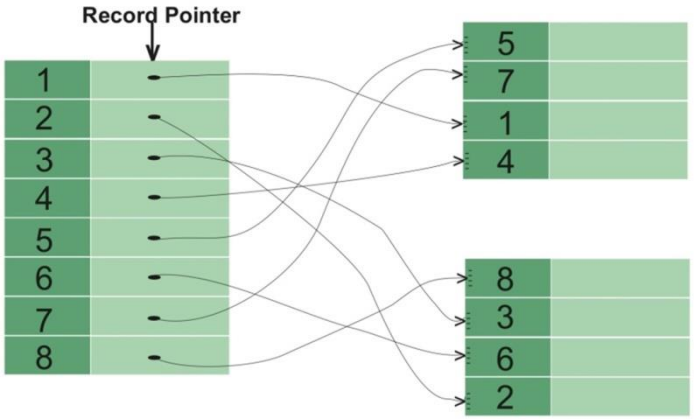




EXAMPLE OF CLUSTERED INDEX

## 2)Un-Clustered Index:

- A file organization when the records are stored and referred in a different way as data entries inindex is called “un-clustered”.
- A Non-clustered index stores the data at one location and indices at another location. The indexcontains pointers to the location of that data.






EXAMPLE OF NON-CLUSTERED INDEX

## Differences between clustered and un-clustered:

- A file organization when the records are stored and referred in same way as data entries in index iscalled clustered. Whereas un-clustered referred refer in a different way as data entries in index is called un-clustered.
- A clustered index is an index which uses alternative (1) whereas un-clustered index uses alternative (2) and (3).
- Clustered index refer few pages when we require retrieving the records. Whereas un-clustered indexrefer several pages when we require retrieving the records.
- If a file contains the records in sequential order, the index search key specifies the same order toretrieve the records from the sequential order of the file is called clustered index.
- Whereas the index search key specifies the different order to retrieve the records from the sequentialorder of the file is called un-clustered index.

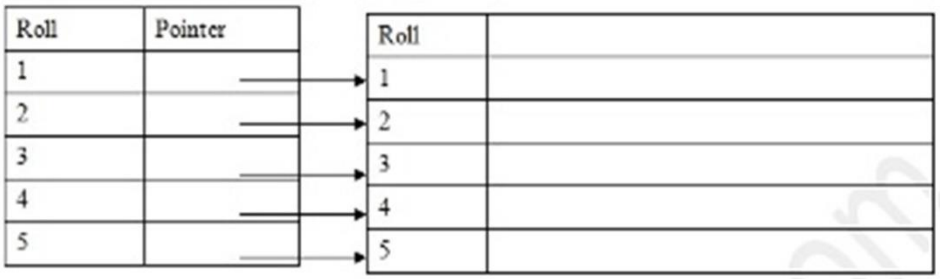
## 5.4 primary index & Secondary index:

### Primary Index:

- An index on a set of fields that includes the primary key is called a primary index.
- An index that uses Alternative (1) is called a primary index.
- Primary indexes are further divided into dense index and sparse index.

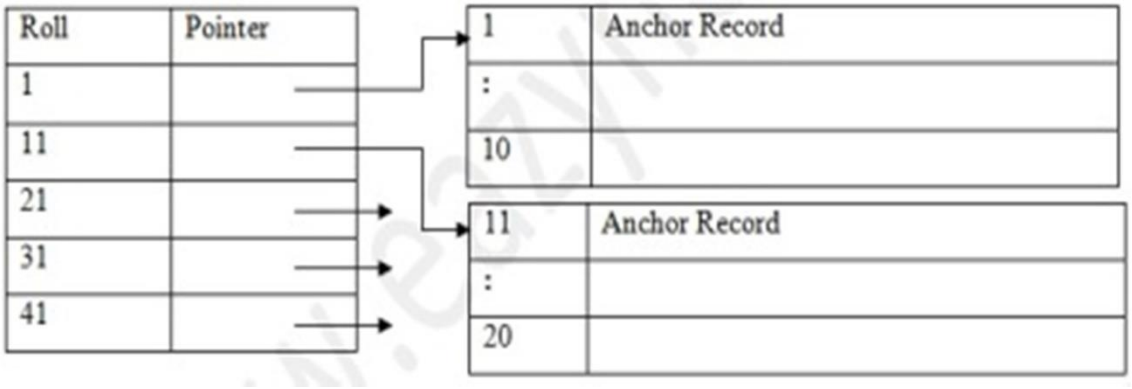
#### i) Dense Index:

- An index record appears for every search key value in the file. The index record contains thesearch-key value and a pointer to the first record with that search-key.
- In dense index, there is an index record for every search key value in the database. This makessearching 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.



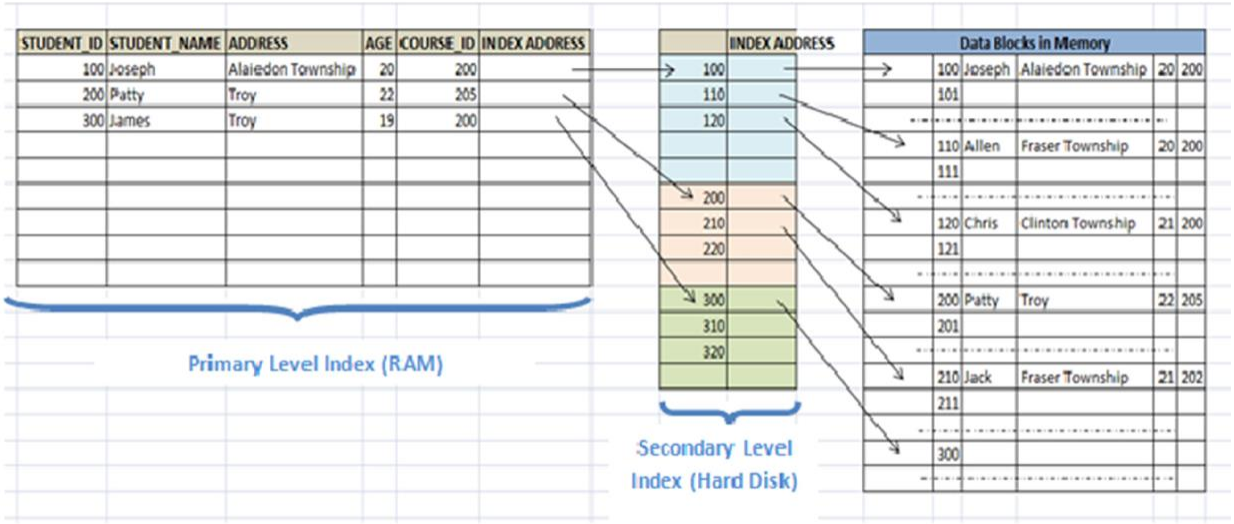

## ii) Sparse Index:

- An index record is created for only some of the values.
- As it is true in dense indexes, each index record contains a search-key value and a pointer to the firstdata record with the largest search-key value that is less than or equal to the search-key value for which we are looking.
- 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 indexrecord 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 isfound.




## 2)Secondary Index:-

- An index that is not a primary key index is called a secondary index. That is an index on a set of fields that does not include the primary key is called a secondary index. This method is the next version of sparse indexing, In this method, another level of indexing is introduced to reduce the(index, address) mapping size.
- That means initially huge range for the columns are selected so that first level of mapping size is small. Then each range is further divided into smaller ranges. First level of mapping is stored in theprimary memory so that address fetch is faster.
- Secondary level of mapping and the actual data are stored in the secondary memory – hard disk. In general, secondary indices are different from primary indices. It the search key of a primary index innot a primary key, it suffices the index pointing to the first record with a particular value for the search key, since the other records can be fetched by a sequential scan of the file.
- A secondary index must contain pointers to all the records, because if the search key of a secondaryindex is not a primary key, it is not enough to point to just the first record. Thus, the records are ordered by the search key of the primary index but same search-key value could be anywhere in thefile.
- A sequential scan in primary index is efficient because records in the file are stored physically in the same order as the index order. We cannot store a file physically ordered both by the search key of theprimary index and the search key of a secondary index.
- Because secondary-key order and physical-key order differ, but if we attempt to scan the file sequentially in secondary-key order, the reading of each record is likely to require the reading of anew block from disk.




---

## 5.5 Indexed Data Structures:

### Hash based Indexing:-

- In this approach, the records in a file are grouped in buckets, where a bucket consists of a primarypage and, possibly, additional pages linked in a chain.
- The bucket to which a record belongs can be determined by applying a special function, called a hashfunction, to the search key.
- Given a bucket number, a hash-based index structure allows us to retrieve the primary page for thebucket in one or two disk I/Os.

### Record Insertion:

The record is inserted into the appropriate bucket, with 'overflow' pages allocated as necessary.

## Record Searching:

- A hash function is used to find first, the bucket containing the records and then by scanning all thepages in a bucket, the record with a given search key can be found.
- Suppose, if the record doesn't have search key value then all the pages in the file needs to bescanned.

## Record Retrieval:

- By applying a hash function to the record's search key, the page containing the needed record can beidentified and retrieved in one disk I/O.

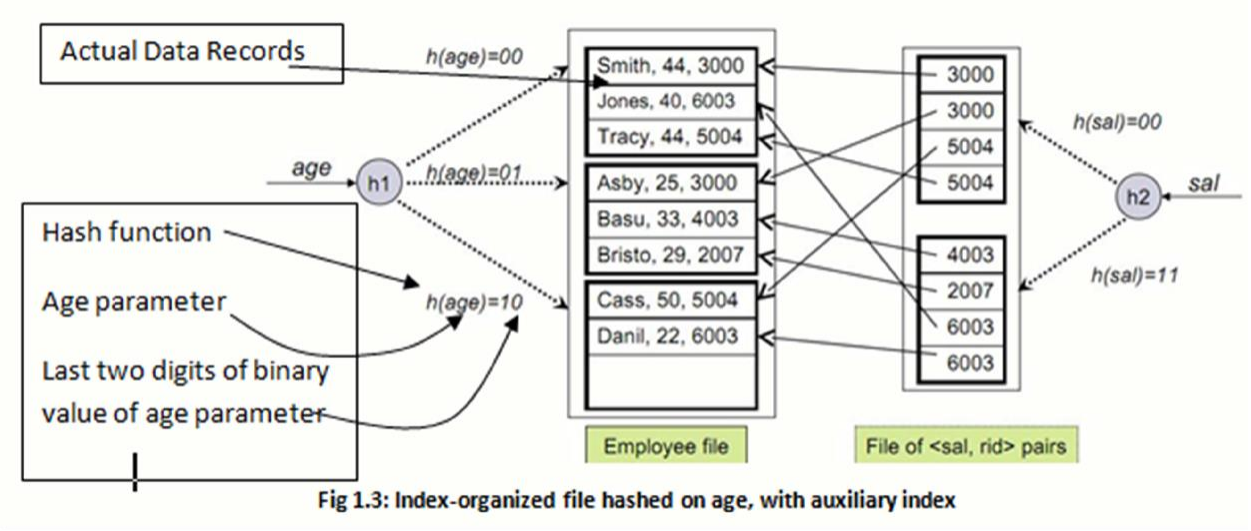
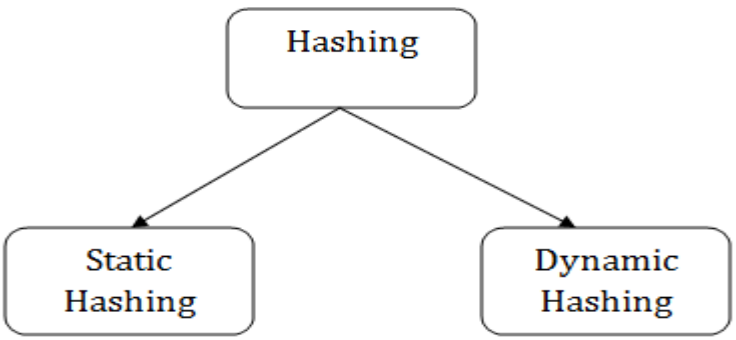



**Fig 1.3: Index-organized file hashed on age, with auxiliary index**

- Hash indexing is illustrated in Figure, where the data is stored in a file that is hashed on age; the dataentries in this first index file are the actual data records.
- Applying the hash function to the age field identifies the page that the record belongs to.
- The hash function h for this example is quite simple; it converts the search key value to its binaryrepresentation and uses the two least significant bits as the bucket identifier.
- Figure also shows an index with search key sal that contains (sal, rid) pairs as data entries.
- The rid (short for record id) component of a data entry in this second index is a pointer to a recordwith search key value sal (and is shown in the figure as an arrow pointing to the data record).

## Types of Hashing:

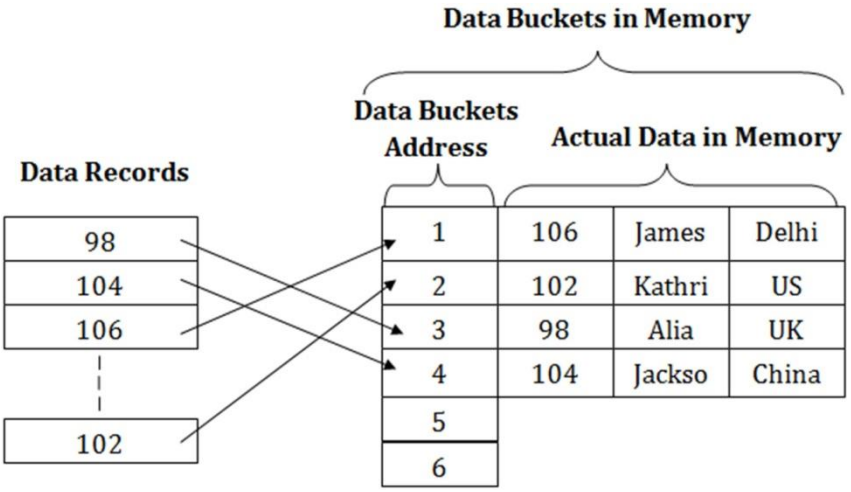


## a) Static Hashing(or) Internal Hashing:

- In static hashing, when a search-key value is provided, the hash function always computes the same address. For example,mod-5 hash function The number of buckets provided remains unchanged at alltimes.

## Operation:

- **Insertion** − When a record is required to be entered using static hash, the hash function **h** computesthe bucket address for search key **K**, where the record will be stored.
- **Search** − When a record needs to be retrieved, the same hash function can be used to retrieve the address of the bucket where the data is stored.
- **Delete** − This is simply a search followed by a deletion operation.



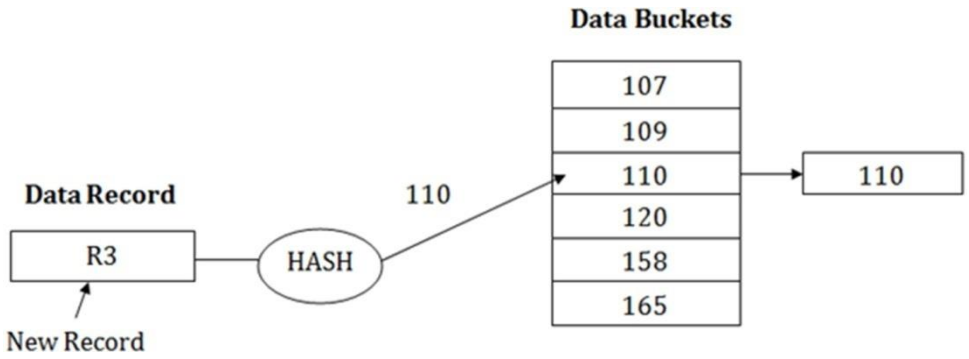

## Bucket Overflow:

If we want to insert some new record into the file but the address of a data bucket generated by the hash function is not empty, or data already exists in that address. This situation in the static hashing is known as **bucket overflow**. This is a critical situation in this method. The condition of bucket-overflow is known as **collision**.

To overcome this situation, there are various methods. Some commonly used methods are as follows:

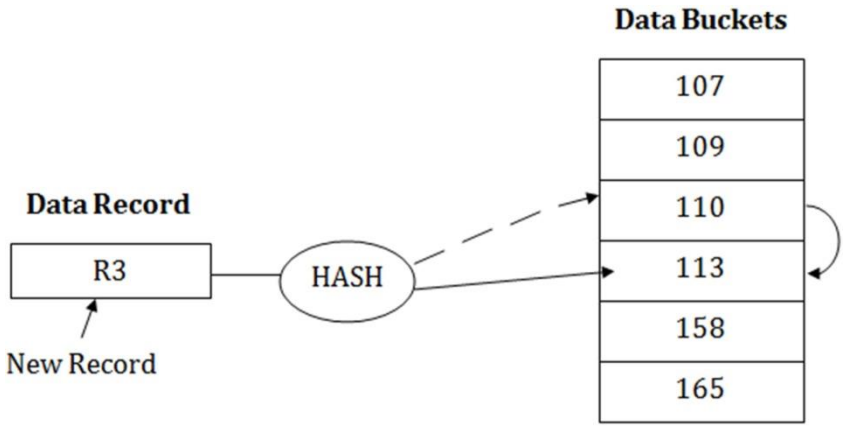
### 1.Closed Hashing:-

- When buckets are full, a new bucket is allocated for the same hash result and is linkedafter the previous one. This mechanism is called Closed Hashing.
- Suppose R3 is a new address which needs to be inserted into the table, the hash function generatesaddress as 110 for it.
- But this bucket is full to store the new data. In this case, a new bucket is inserted at the end of 110buckets and is linked to it.




## 2. Open Hashing:

- When a hash function generates an address at which data is already stored, the next free bucket isallocated to it. This mechanism is called Open Hashing.
- Suppose R3 is a new address which needs to be inserted, the hash function generates address as 110 for R3. But the generated address is already full. So the system searches next available data bucket,113 and assigns R3 to it.




## b) Dynamic Hashing(or) External Hashing (or) Extendible Hashing:-

- The problem with static hashing is that it does not expand or shrink dynamically as the size of the database grows or shrinks. Dynamic hashing provides a mechanism in which data buckets are addedand removed dynamically and on-demand. Dynamic hashing is also known as extended hashing.
- Hash function, in dynamic hashing, is made to produce a large number of values and only a few areused initially.

## Searching:

- First, calculate the hash address of the key. Check how many bits are used in the directory.
- Take the least significant bits of the hash address. This gives an index of the directory.
- Now using the index, go to the directory and find bucket address where the record might be.

## Insertion:

- Firstly, you have to follow the same procedure for retrieval, ending up in some bucket.
- If there is still space in that bucket, then place the record in it.
- If the bucket is full, then we will split the bucket and redistribute the records.

## Deletion:

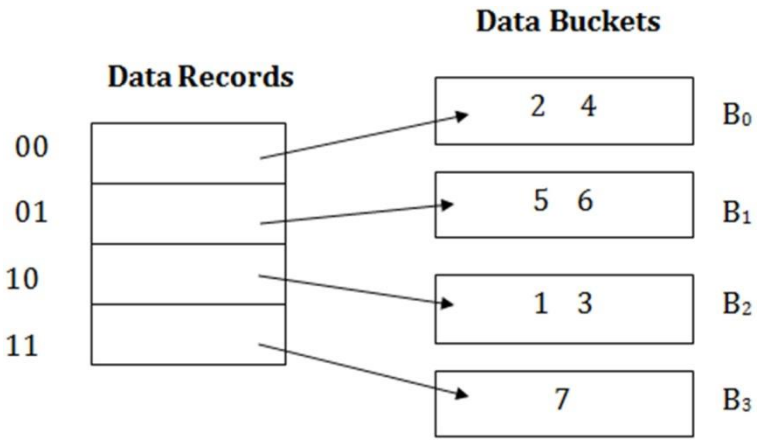
- Deletion is nothing but a simple search operation followed by delete operation after entering into thebucket, with the help of the hash function.

**For example:-**

Consider the following grouping of keys into buckets, depending on the prefix of their hash address:

| Key | Hash address |
|---|---|
| 1 | 11010 |
| 2 | 00000 |
| 3 | 11110 |
| 4 | 00000 |
| 5 | 01001 |
| 6 | 10101 |
| 7 | 10111 |

- The last two bits of 2 and 4 are 00. So it will go into bucket B0. The last two bits of 5 and 6 are 01, so it will go into bucket B1. The last two bits of 1 and 3 are 10, so it will go into bucket B2. The last two bits of 7 are 11, so it will go into B3.




Insert key 9 with hash address 10001 into the above structure:

- Since key 9 has hash address 10001, it must go into the first bucket. But bucket B1 is full, so it will get split.
- The splitting will separate 5, 9 from 6 since last three bits of 5, 9 are 001, so it will go into bucketB1, and the last three bits of 6 are 101, so it will go into bucket B5.
- Keys 2 and 4 are still in B0. The record in B0 pointed by the 000 and 100 entry because last two bitsof both the entry are 00.

- Keys 1 and 3 are still in B2. The record in B2 pointed by the 010 and 110 entry because last two bitsof both the entry are 10.
- Key 7 are still in B3.
- The record in B3 pointed by the 111 and 011 entry because last two bits of both the entry are 11.