File Organizations Concepts

File organization means how data (records) is stored in a file on your computer or database. The way you organize data affects how fast you can:

- Find a record
- Add new data
- Update or delete existing data

Think of it like organizing books in a library — the method you choose affects how quickly you can find a book.

File Organization in DBMS

File organization in DBMS refers to the method of storing data records in a file so they can be accessed efficiently. It determines how data is arranged, stored, and retrieved from physical storage.

The Objective of File Organization

- It helps in the faster selection of records i.e. it makes the process faster.
- Different Operations like inserting, deleting, and updating different records are faster and easier.
- It prevents us from inserting duplicate records via various operations.
- It helps in storing the records or the data very efficiently at a minimal cost.

Types of File Organizations

Various methods have been introduced to Organize files. These particular methods have advantages and disadvantages on the basis of access or selection. Thus it is all upon the programmer to decide the best-suited file Organization method according to his requirements.

Some types of File Organizations are:

- Sequential File Organization
- Heap File Organization
- Clustered File Organization
- ISAM (Indexed Sequential Access Method)
- Hash File Organization
- B+ Tree File Organization

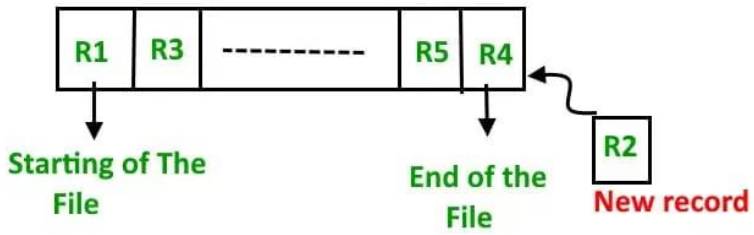
Sequential File Organization

The easiest method for file Organization is the Sequential method. In this method, the file is stored one after another in a sequential manner. There are two ways to implement this method:

1. Pile File Method

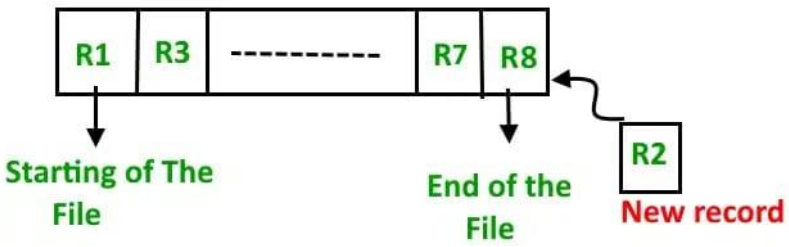
This method is quite simple, in which we store the records in a sequence i.e. one after the other in the order in which they are inserted into the tables.

Insertion of the new record: Let the R1, R3, and so on up to R5 and R4 be four records in the sequence. Here, records are 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.



2. Sorted File Method

In this method, As the name itself suggests whenever a new record has to be inserted, it is always inserted in a sorted (ascending or descending) manner. The sorting of records may be based on any primary key or any other key.



Insertion of the new record: Let us assume that there is a preexisting sorted sequence of four records R1, R3, and so on up to R7 and R8. Suppose a new record R2 has to be inserted in the sequence, then it will be inserted at the end of the file and then it will sort the sequence.

Advantages of Sequential File Organization

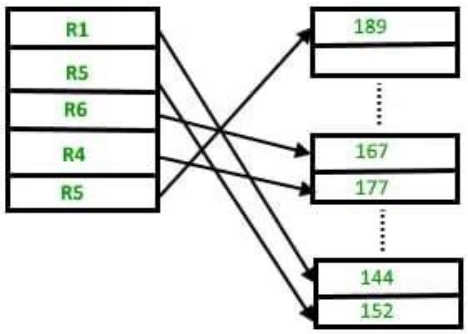
- Fast and efficient method for huge amounts of data.
- Simple design.
- Files can be easily stored in magnetic tapes i.e. cheaper storage mechanism.

Disadvantages of Sequential File Organization

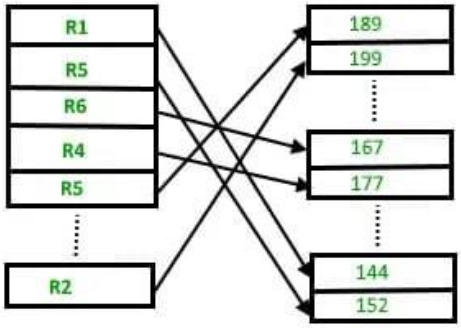
- Time wastage as we cannot jump on a particular record that is required, but we have to move in a sequential manner which takes our time.
- The sorted file method is inefficient as it takes time and space for sorting records.

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 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.



Insertion of the new record: Suppose we have four records in the heap R1, R5, R6, R4, and R3, and suppose a new record R2 has to be inserted in the heap then, since the last data block i.e data block 3 is full it will be inserted in any of the data blocks selected by the DBMS, let's say data block 1.



If we want to search, delete or update data in the heap file Organization we will traverse the data from the beginning of the file till we get the requested record. Thus, if the database is very huge, searching, deleting, or updating the record will take a lot of time.

Advantages of Heap File Organization

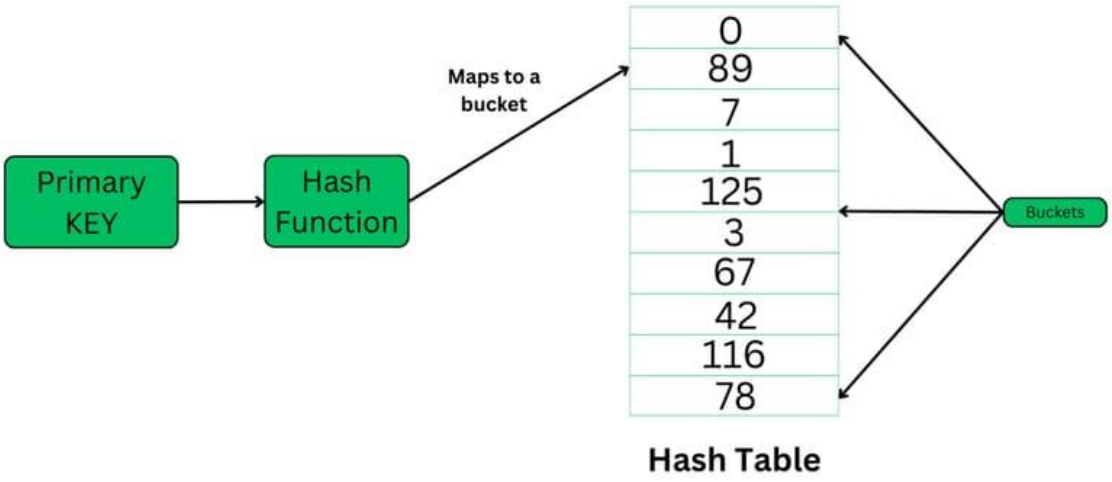
- Fetching and retrieving records is faster than sequential records but only in the case of small databases.
- When there is a huge number of data that needs to be loaded into the database at a time, then this method of file Organization is best suited.

Disadvantages of Heap File Organization

- The problem of unused memory blocks.
- Inefficient for larger databases.

In DBMS, when we want to retrieve a particular data, it becomes very inefficient to search all the index values and reach the desired data. In this situation, Hashing technique comes into the picture.

Hashing is an efficient technique to directly search the location of desired data on the disk without using an index structure. Data is stored at the data blocks whose address is generated by using a hash function.



Note: *The memory location where these records are stored is called a data block or data bucket.*

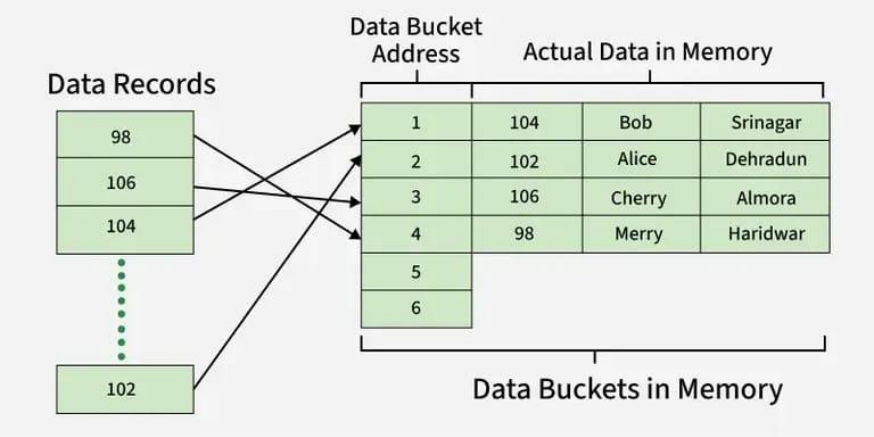
Hash File Organization

- **Data bucket:** Data buckets are the memory locations where the records are stored. These buckets are also considered Units of Storage.
- **Hash Function:** The hash function is a mapping function that maps all the sets of search keys to the actual record address. Generally, the hash function uses the primary key to generate the hash index.
- **Hash Index:** The prefix of an entire hash value is taken as a hash index. Every hash index has a depth value to signify how many bits are used for computing a hash function. These bits can address 2n buckets.

Note: *When all these bits are consumed? then the depth value is increased linearly and twice the buckets are allocated.*

Static Hashing

In static hashing, when a search-key value is provided, the hash function always computes the same address. For example, if we want to generate an address for STUDENT_ID = 104 using a mod (5) hash function, it always results in the same bucket address 4. There will not be any changes to the bucket address here. Hence a number of data buckets in the memory for this static hashing remain constant throughout.



Operations

Insertion:

- When a new record is inserted into the table, The hash function h generates a bucket address for the new record based on its hash key K.
- Bucket address = h(K)

Searching:

- When a record needs to be searched, the same hash function is used to retrieve the bucket address for the record.
- **Example:** if we want to retrieve the whole record for ID 104 and if the hash function is mod (5) on that ID, the bucket address generated would be 4.
- Then we will directly got to address 4 and retrieve the whole record for ID 104. Here ID acts as a hash key.

Deletion:

- If we want to delete a record, Using the hash function we will first fetch the record which is supposed to be deleted.
- Then we will remove the records for that address in memory.

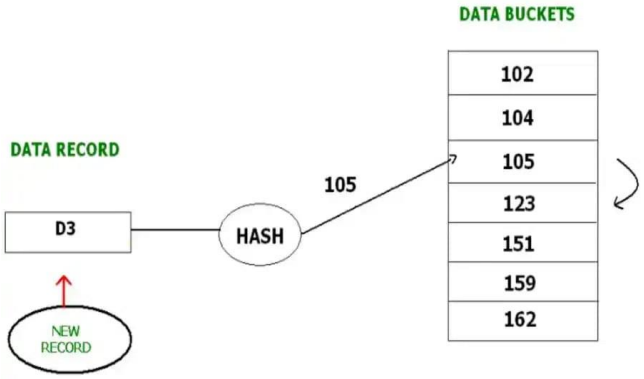
Updation: The data record that needs to be updated is first searched using the hash function and then the data record is updated.

Note: *If we want to insert some new records into the file But the data bucket address generated by the hash function is not empty or the data already exists in that address. This situation in static hashing is called* ***bucket overflow****.*

Commonly Used Methods

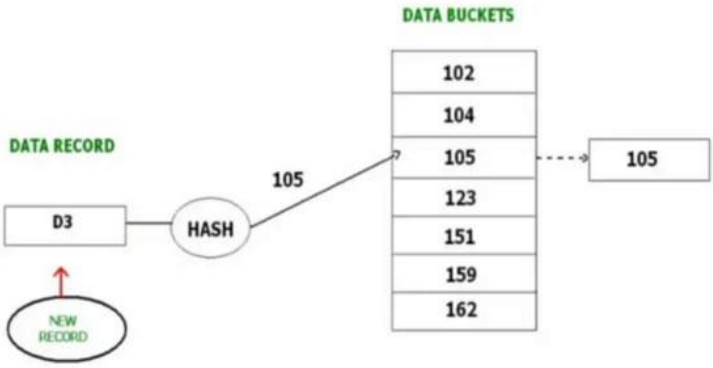
Open Hashing: In the Open hashing method, the next available data block is used to enter the new record, instead of overwriting the older one.

- This method is also called linear probing.
- **Example:** D3 is a new record that needs to be inserted, the hash function generates the address as 105. But it is already full. So, the system searches the next available data bucket, 123 and assigns D3 to it.



Closed hashing: In the Closed hashing method, a new data bucket is allocated with the same address and is linked to it after the full data bucket.

- This method is also known as overflow chaining.
- **Example:** we have to insert a new record D3 into the tables.
- The static hash function generates the data bucket address as 105. But this bucket is full to store the new data.
- In this case, a new data bucket is added at the end of the 105 data bucket and is linked to it. The new record D3 is inserted into the new bucket.



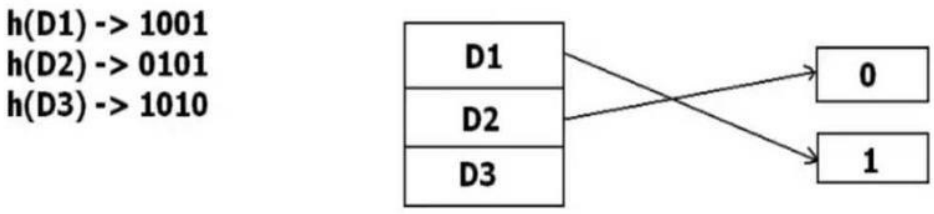
Quadratic probing: Quadratic probing is very much similar to open hashing or linear probing. Here, the only difference between old and new buckets is linear. The quadratic function is used to determine the new bucket address.

Double Hashing: Double Hashing is another method similar to linear probing. Here the difference is fixed as in linear probing, but this fixed difference is calculated by using another hash function. That's why the name is double hashing.

Dynamic Hashing

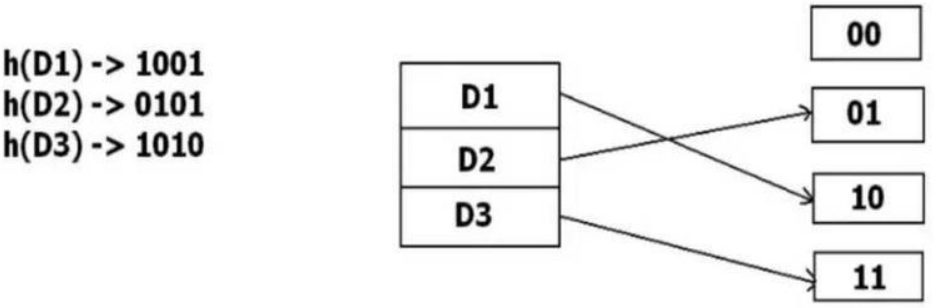
In Dynamic hashing, data buckets grow or shrink as the records increase or decrease & also known as extended hashing. Here,

- The hash function is made to produce a large number of values.
- **Example:** there are three data records D1, D2 and D3.
- The hash function generates three addresses 1001, 0101 and 1010 respectively.
- This method of storing considers only part of this address - especially only the first bit to store the data. So, it tries to load three of them at addresses 0 and 1.



But the problem is that No bucket address is remaining for D3.

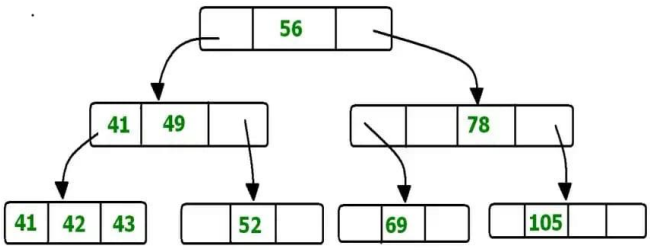
- The bucket has to grow dynamically to accommodate D3.
- So it changes the address to have 2 bits rather than 1 bit and then it updates the existing data to have a 2-bit address.
- Then it tries to accommodate D3.



B+ Tree File Organization

B+ Tree, as the name suggests, uses a tree-like structure to store records in a File. It uses the concept of Key indexing where the primary key is used to sort the records. For each primary key, an index value is generated and mapped with the record. An index of a record is the address of the record in the file.

B+ Tree is very similar to a binary search tree, with the only difference being that instead of just two children, it can have more than two. All the information is stored in a leaf node and the intermediate nodes act as a pointer to the leaf nodes. The information in leaf nodes always remains a sorted sequential linked list.



In the above diagram, 56 is the root node which is also called the main node of the tree. The intermediate nodes here, just consist of the address of leaf nodes. They do not contain any actual records. Leaf nodes consist of the actual record. All leaf nodes are balanced.

Advantages of B+ Tree File Organization

- Tree traversal is easier and faster.
- Searching becomes easy as all records are stored only in leaf nodes and are sorted in sequentially linked lists.
- There is no restriction on B+ tree size. It may grow/shrink as the size of the data increases/decreases.

Disadvantages of B+ Tree File Organization

- Inefficient for static tables.

Cluster File Organization

In **Cluster file organization**, two or more related tables/records are stored within the same file known as clusters. These files will have two or more tables in the same data block and the key attributes which are used to map these tables together are stored only once.

Thus, it lowers the cost of searching and retrieving various records in different files as they are now combined and kept in a single cluster. For example, we have two tables or relation Employee and Department. These tables are related to each other.

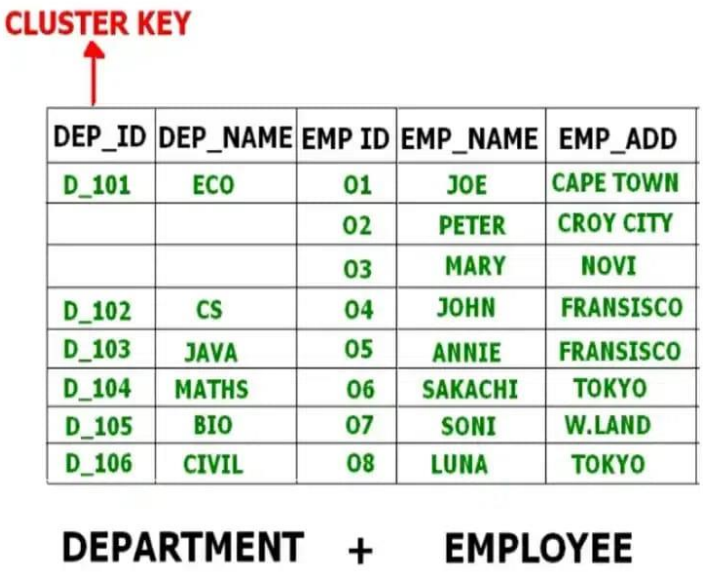
EMPLOYEE

EMP ID	EMP_NAME	EMP_ADD	DEP_ID
01	JOE	CAPE TOWN	D_101
02	ANNIE	FRANSISCO	D_103
03	PETER	CROY CITY	D_101
04	JOHN	FRANSISCO	D_102
05	LUNA	TOKYO	D_106
06	SONI	W.LAND	D_105
07	SAKACHI	TOKYO	D_104
08	MARY	NOVI	D_101

DEPARTMENT

DEP_ID	DEP_NAME
D_101	ECO
D_102	CS
D_103	JAVA
D_104	MATHS
D_105	BIO
D_106	CIVIL

Therefore, this table is allowed to combine using a join operation and can be seen in a cluster file.



If we have to insert, update or delete any record we can directly do so. Data is sorted based on the primary key or the key with which searching is done. The **cluster key** is the key with which the joining of the table is performed.

Types of Cluster File Organization

There are two ways to implement this method.

- **Indexed Clusters:** In Indexed clustering, the records are grouped based on the cluster key and stored together. The above-mentioned example of the Employee and Department relationship is an example of an Indexed Cluster where the records are based on the Department ID.
- **Hash Clusters:** This is very much similar to an indexed cluster with the only difference that instead of storing the records based on cluster key, we generate a hash key value and store the records with the same hash key value.

Advantages of Cluster File Organization

- It is basically used when multiple tables have to be joined with the same joining condition.
- It gives the best output when the cardinality is 1:m.

Disadvantages of Cluster File Organization

- It gives a low performance in the case of a large database.
- In the case of a 1:1 cardinality, it becomes ineffective.

ISAM (Indexed Sequential Access Method):

A combination of sequential and indexed methods. Data is stored sequentially, but an index is maintained for faster access. Think of it like having a bookmark in a book that guides you to specific pages.

Advantages of ISAM:

- Faster retrieval compared to pure sequential methods.
- Suitable for applications with a mix of sequential and random access.

Disadvantages of ISAM:

- Index maintenance can add overhead in terms of storage and update operations.
- Not as efficient as fully indexed methods for random access.