**Why Did we Shift from File System to DBMS:**

File System manages data using files in hard disk. Users are allowed to create, delete, and update the files according to their requirement. Let us consider the example of file based University Management System. Data of students is available to their respective Departments, Academics Section, Result Section, Accounts Section, Hostel Office etc. Some of the data is common for all sections like Roll No, Name, Father Name, Address and Phone number of students but some data is available to a particular section only like Hostel allotment number which is a part of hostel office. Let us discuss the issues with this system:

* **Redundancy of data:** Data is said to be redundant if same data is copied at many places. If a student wants to change Phone number, he has to get it updated at various sections. Similarly, old records must be deleted from all sections representing that student.
* **Inconsistency of Data:**Data is said to be inconsistent if multiple copies of same data does not match with each other. If Phone number is different in Accounts Section and Academics Section, it will be inconsistent. Inconsistency may be because of typing errors or not updating all copies of same data.
* **Difficult Data Access:** A user should know the exact location of file to access data, so the process is very cumbersome and tedious. If user wants to search student hostel allotment number of a student from 10000 unsorted students’ records, how difficult it can be.
* **Unauthorized Access:** File System may lead to unauthorized access to data. If a student gets access to file having his marks, he can change it in unauthorized way.
* **No Concurrent Access:**The access of same data by multiple users at same time is known as concurrency. File system does not allow concurrency as data can be accessed by only one user at a time.
* **No Backup and Recovery:** File system does not incorporate any backup and recovery of data if a file is lost or corrupted.

These are the main reasons which made a shift from file system to DBMS.

Why We Use DBMS:

* To develop software applications In less time.
* Data independence and efficient use of data.
* For data integrity and security.
* For concurrent access to data, and data recovery from crashes.
* To use user-friendly declarative query language.

Advantages of DBMS

* **Controls database redundancy:** It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.
* **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.
* **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.
* **Reduce time:** It reduces development time and maintenance need.
* **Backup:** It provides backup and recovery subsystems which create automatic backup of data from [hardware](https://www.javatpoint.com/hardware) and [software](https://www.javatpoint.com/software) failures and restores the data if required.
* **multiple user interface:** It provides different types of user interfaces like graphical user interfaces, application program interfaces
* **-**----------------------------------------------------------------------

### Hash File Organization

It is a file organization technique where a hash function is used to compute the address of a record. It uses the value of an attribute or set of attributes as input and gives the location (page/block/bucket) where the record can be stored.

For example, let us consider the following table Student;

|  |  |  |  |
| --- | --- | --- | --- |
| ***RegNo*** | ***SName*** | ***Gen*** | ***Phone*** |
| 1 | Sundar | M | 9898786756 |
| 3 | Karthik | M | 8798987867 |
| 4 | John | M | 7898886756 |
| 2 | Ram | M | 9897786772 |
| 5 | Martin | M | 9765430231 |
| 6 | Rashmi | F | 8976543990 |

A hash function is a function which maps the large set of values into smaller set of files/locations/values. Let us organize the above table using the ***phone*** attribute value as input for the hash function.

***h(phone mod 10)***

In the above hash function, phone is the phone attribute’s value of each record. 10 is the number of buckets/pages where we want to store our table. [10 buckets means bucket0, bucket1, …, bucket9].

For our example,

*For 1st record, h(9898786756 mod 10)  = 6 ie., the first record has to be stored in 6th bucket.*

*For 2nd record, h(8798987867 mod 10) = 7 ie., the second record has be stored in 7th bucket.*

*…*

*For 4th record, h(7898886756 mod 10) = 6 ie., the fourth record has be stored in 6th bucket [like 1st]*

*For 5th record, h(9765430231 mod 10) = 1 ie., the 5th record has to be stored in 1st bucket.*

*For last record, h(8976543990 mod 10) = 0 ie., the last record has to be stored in 0th bucket.*

--------------------------------------------------------------------------------------------------------------------------

## Indexed Sequential File Organization

An indexed sequential file consists of records that can be accessed sequentially. Direct access is also possible. It consists of two parts −

* **Data File** contains records in sequential scheme.
* **Index File** contains the primary key and its address in the data file.

Following are the key attributes of sequential file organization −

* Records can be read in sequential order just like in sequential file organization.
* Records can be accessed randomly if the primary key is known. Index file is used to get the address of a record and then the record is fetched from the data file.
* Sorted index is maintained in this file system which relates the key value to the position of the record in the file.
* Alternate index can also be created to fetch the records.

**ADVANTAGES of INDEXED FILES**

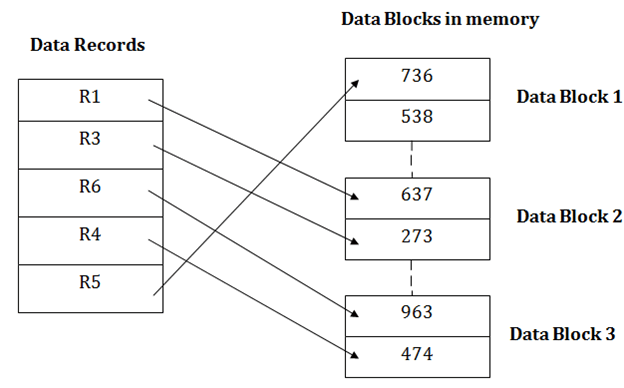
1. Quite easy to process,
2. With proper selection of a key field, records in a large file can be searched and accessed in very quickly.
3. Any field of the records can be used as the key. The key field can be numerical or alphanumerical.

**DISADVANTAGES of INDEXED FILES**

1. Extra data structures have to be maintained (the COBOL run-time modules take care of these and it is **not** the programmers' concern). These extra data structures maintained on the disk can use up much disk space, especially for long key values.
2. The indexed files have to be reorganized from time time to get rid of deleted records and improve performance that gets gradually decreased with addition of n

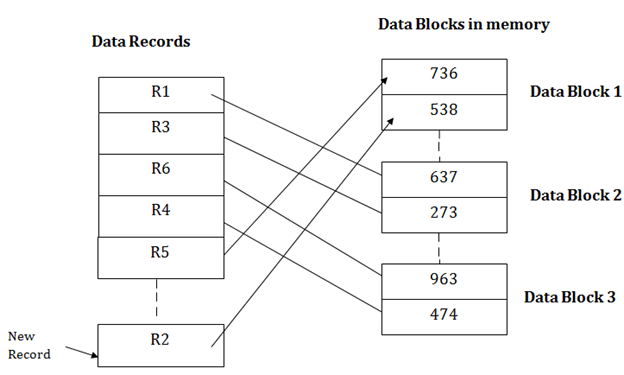
# **Heap file organization**

* It is the simplest and most basic type of organization. It works with data blocks. In heap file organization, the records are inserted at the file's end. When the records are inserted, it doesn't require the sorting and ordering of records.
* When the data block is full, the new record is stored in some other block. This new data block need not to be the very next data block, but it can select any data block in the memory to store new records. The heap file is also known as an unordered file.
* In the file, every record has a unique id, and every page in a file is of the same size. It is the DBMS responsibility to store and manage the new records.



## Insertion of a new record

Suppose we have five records R1, R3, R6, R4 and R5 in a heap and suppose we want to insert a new record R2 in a heap. If the data block 3 is full then it will be inserted in any of the database selected by the DBMS, let's say data block 1.



If we want to search, update or delete the data in heap file organization, then we need to traverse the data from staring of the file till we get the requested record.

If the database is very large then searching, updating or deleting of record will be time-consuming because there is no sorting or ordering of records. In the heap file organization, we need to check all the data until we get the requested record.

## Pros of Heap file organization

* Simple file organization.
* Insertion becomes easy.
* In case of a small database, fetching and retrieving of records is faster than the sequential record.

## Cons of Heap file organization

* This method is inefficient for the large database because it takes time to search or modify the record.
* This method is inefficient for large databases.
* **Physical files** contain the actual data that is stored on the system, and a description of how data is to be presented to or received from a program. They contain only one record format, and one or more members.
* **Logical files** do not contain data. They contain a description of records found in one or more physical files. A logical file is a view or representation of one or more physical files. Logical files that contain more than one format are referred to as **multi-format** logical files.

|  |  |
| --- | --- |
| **Physical File** | **Logical File** |
| It occupies the portion of memory. It contains the original data. | It does not occupy memory space. It does not contain data. |
| A physical file contains one record format. | It can contain upto 32 record formats. |
| It can exist without logical file. | It cannot exist without physical file. |
| If there is a logical file for physical file, the physical file cannot be deleted until and unless we delete the logical file. | If there is a logical file for a physical file, the logical file can be deleted without deleting the physical file. |