File Organization and Indexes:

A **file organization** is a way of arranging the records in a file when the file is stored on disk. A file record is likely to be accessed and modified in a variety of ways, and different ways of arranging the records enable different operations over the file to be carried out efficiently. Different ways in which the files can be organized are :-

- Heap files
- Sorted files
- Clustered tree index
- Unclustered tree index
- Unclustered hash index

Cost of various operation of DBMS on different types of files

File Type	Scan	Equality Search	Range Search	Insert	Delete
Heap	PD	0.5PD	PD	2D	Search + D
Sorted	PD	$D\log_2(P)$	$D\log_2(P)$ + matching pages	Search + PD	Search + PD
Clustered Tree Index	1.5PD	$D\log_F(1.5P)$	$D\log_F(1.5P)$ + matching pages	Search + D	Search + D
Unclustered Tree Index	PDR + Read index	D + $D\log_F(0.15P)$	$D\log_F$(index size) + D*matching records	3D + $D\log_F$(index size)	Search + 2D
Unclustered Hash Index	PDR + Read index	2D	PD	4D	Search + 2D

Where P➔no. of pages in the file.
D➔amount of time required to read or write in page.
R➔no. of records in a particular page.

Heap Files

Scan: Cost is **PD** since we have to retrieve each of P pages with each page taking D time.

Equality Search: If exactly one record matches the desired equality search then on average we must scan half of the file, assuming record exists in only that part of file. Hence cost is **0.5PD**.

Range Search: In this entire file must be scanned for matching records. So cost is **PD**.

Insert: If records are inserted at the end of page the time taken is fetching the page and writing back the page. So cost is **2D**.

Delete: Here time taken is searching for relevant record and writing back the page after deleting record from it. So cost is **Search + D**.

Sorted Files

Scan: Cost is **PD** since we have to retrieve each of P pages with each page taking D time.

Equality Search: If we assume that the equality search is specified on the field by which the file is sorted, then we can search for the record by the help of binary search. Hence cost is $\mathbf{D\log_2(P)}$.

Range Search: It is equality search for all matching records. So cost is $\mathbf{D\log_2(P)}$ **+ matching pages**.

Insert: To insert the record while preserving the sorted order, first we have to search for the correct position in the file, add record and then fetch and rewrite all subsequent pages. So cost is **Search + PD**.

Delete: Here we search for record, remove the record from the page, and rewrite the subsequent pages to fill the space created by the record which is deleted. Hence cost is **Search + PD**.

Clustered Tree Index

Scan: Here effective number of pages is 1.5 times more than pages in heap files since page occupancy is 67%. So, Cost is **1.5PD** since we have to retrieve all the pages with each page taking D time.

Equality Search: If data records are ordered as data entries in some index, then we do F-ary search. So cost in $\mathbf{D\log_F(1.5P)}$.

Range Search: It is equality search for all matching records. So cost is $\mathbf{D\log_F(1.5P)}$ **+ matching pages**.

Insert: Here time required is for searching correct position for record in the page and writing back the page. So cost is **Search + D**.

Delete: Similar to insert, first search for page, delete record from it and write back the page. Cost is **Search + D**.

Unclustered Tree Index

Scan: Here each record takes D time to read from a single page. So reading R record from a page takes DR time. Hence total cost for P pages is **PDR + Read index**.

Equality Search: If we assume that data index size is one-tenth of data record, then no. leaf pages are 0.15P. So cost incurred is $\mathbf{D + D\log_F(0.15P)}$.

Range Search: It includes equality search and matching pages. So cost is $\mathbf{D\log_F(index\ size)}$ **+ D*matching records.**

Insert: Time required is for searching the page, fetching it, adding records and writing back the page. So cost is $\mathbf{3D + D\log_F(index\ size)}$.

Delete: First we search for the page where record to be deleted is located, then fetch the page, remove record and write back the page. So cost is **Search + 2D**.

Unclustered Hash Index

Scan: Here each record takes D time to read from a single page. So reading R record from a page takes DR time. Hence total cost for P pages is **PDR + Read index**.

Equality Search: If search is on the search key of hashed file, then total cost is of only getting the relevant page of data entry and record, so cost is **2D**.

Range Search: This search can be as worst as scanning the whole file. Hence cost incurred in this is of retrieving all the pages. So cost is **PD**.

Insert: Here by using search key, we can read the relevant pages, add record to it and then write back the page. So cost involved with it is **4D**.

Delete: Cost involved with it is searching for the record, reading the page, deleting the record and writing back the page. So cost is **Search + 2D**.

Comparison of I/O Costs

- A heap file has good storage efficiency and supports fast scanning and insertion of records. However, it is slow for searches and deletions.
- A sorted file also offers good storage efficiency, but insertion and deletion of records is slow. Searches are faster than in heap files.
- A clustered file offers all the advantages of a sorted file and supports inserts and deletes efficiently. Searches are even faster than in sorted files, although a sorted file can be faster when a large number of records are retrieved sequentially, because of blocked I/O efficiencies.
- Unclustered tree and hash indexes offer fast searches, insertion, and deletion, but scans and range searches with many matches are slow. Hash indexes are a little faster on equality searches, but they do not support range searches.

Primary and Secondary Indexes

An index on a set of fields that includes the *super key* is called a **primary index**.
An index that is not primary index is called a **secondary index**.