SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)
Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai
Accredited by NAAC-UGC with 'A++' Grade (Cycle III) &
Accredited by NBA (B.E - CSE, EEE, ECE, Mech & B.Tech.IT)
COIMBATORE-641 035, TAMIL NADU

UNIT V – Physical Storage and MongoDB

Data Storage and Indexes – RAID- File organization-Indexing and Hashing –Ordered Indices – B+ tree Index Files – B tree Index Files – Static Hashing – Dynamic Hashing. Query Processing Overview-Algorithms for Selection and Sorting Basics of MongoDB, Procedural Language

File Organization – Indexing and Hashing- Ordered Indices

What is File Organization?

File Organization refers to the logical relationships among various records that constitute the file, particularly with respect to the means of identification and access to any specific record. In simple terms, Storing the files in a certain order is called File Organization. File Structure refers to the format of the label and data blocks and of any logical control record.

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.

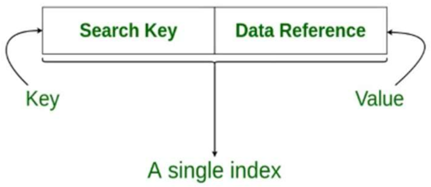
Some types of File Organizations are:

- Sequential File Organization
- Heap File Organization
- Hash File Organization
- B+ Tree File Organization

- Clustered File Organization
- ISAM (Indexed Sequential Access Method)

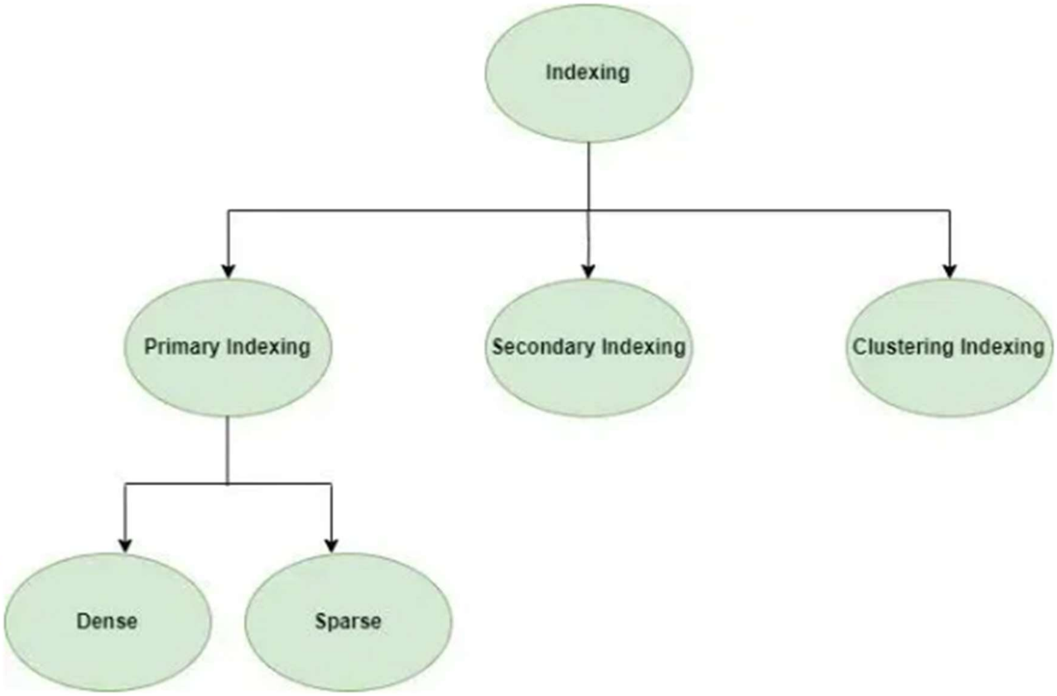
Indexing improves database performance by minimizing the number of disc visits required to fulfill a query. It is a data structure technique used to locate and quickly access data in databases. Several database fields are used to generate indexes. The main key or candidate key of the table is duplicated in the first column, which is the Search key. To speed up data retrieval, the values are also kept in sorted order. It should be highlighted that sorting the data is not required. The second column is the Data Reference or Pointer which contains a set of pointers holding the address of the disk block where that particular key value can be found.

Structure of an Index in Database



Attributes of Indexing

- Access Types: This refers to the type of access such as value-based search, range access, etc.
- Access Time: It refers to the time needed to find a particular data element or set of elements.
- Insertion Time: It refers to the time taken to find the appropriate space and insert new data.
- Deletion Time: Time taken to find an item and delete it as well as update the index structure.
- Space Overhead: It refers to the additional space required by the index.

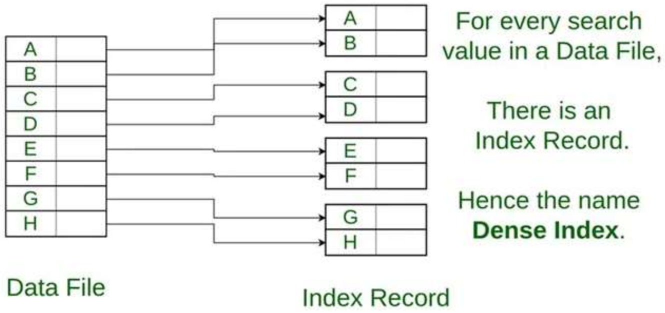


In general, there are two types of file organization mechanisms that are followed by the indexing methods to store the data:

Sequential File Organization or Ordered Index File

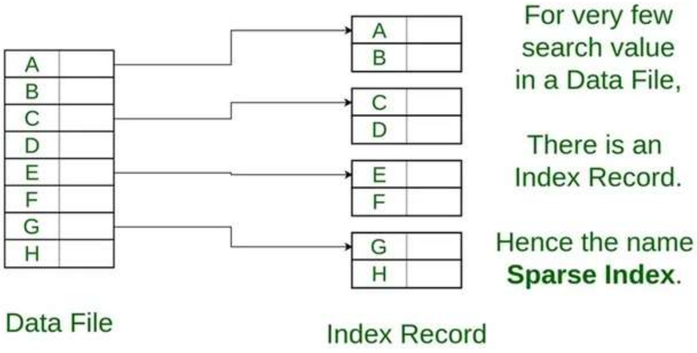
In this, the indices are based on a sorted ordering of the values. These are generally fast and a more traditional type of storing mechanism. These Ordered or Sequential file organizations might store the data in a dense or sparse format.

- **Dense Index**
  - For every search key value in the data file, there is an index record.
  - This record contains the search key and also a reference to the first data record with that search key value.


A
B
C
D
E
F
G
H
Data File
A
B
C
D
E
F
G
H
Index Record
For every search value in a Data File,
There is an Index Record.
Hence the name **Dense Index**.

- **Sparse Index**
  - The index record appears only for a few items in the data file. Each item points to a block as shown.
  - To locate a record, we find the index record with the largest search key value less than or equal to the search key value we are looking for.
  - We start at that record pointed to by the index record, and proceed along with the pointers in the file (that is, sequentially) until we find the desired record.
  - Number of Accesses required=$\log_2(n)+1$, (here n=number of blocks

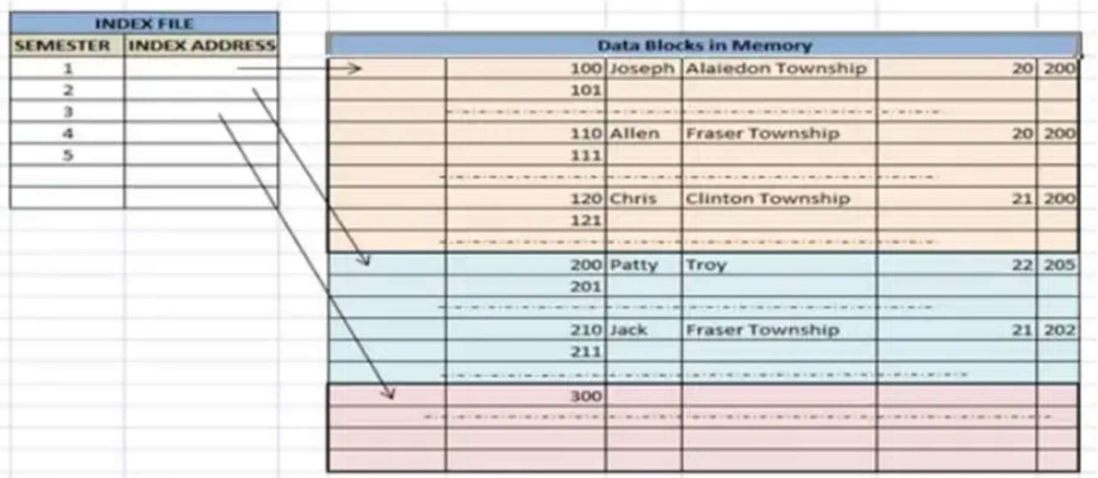
acquired by index file)


A
B
C
D
E
F
G
H
Data File
A
B
C
D
E
F
G
H
Index Record
For very few search value in a Data File,
There is an Index Record.
Hence the name **Sparse Index**.

Hash File Organization

Indices are based on the values being distributed uniformly across a range of buckets. The buckets to which a value is assigned are determined by a function called a hash function. There are primarily three methods of indexing:

Clustered Indexing: When more than two records are stored in the same file this type of storing is known as cluster indexing. By using cluster indexing we can reduce the cost of searching reason being multiple records related to the same thing are stored in one place and it also gives the frequent joining of more than two tables (records). The clustering index is defined on an ordered data file. The data file is ordered on a non-key field. In some cases, the index is created on non-primary key columns which may not be unique for each record. In such cases, in order to identify the records faster, we will group two or more columns together to get the unique values and create an index out of them. This method is known as the clustering index. Essentially, records with similar properties are grouped together, and indexes for these groupings are formed. Students studying each semester, for example, are grouped together. First-semester students, second-semester students, third-semester students, and so on are categorized.



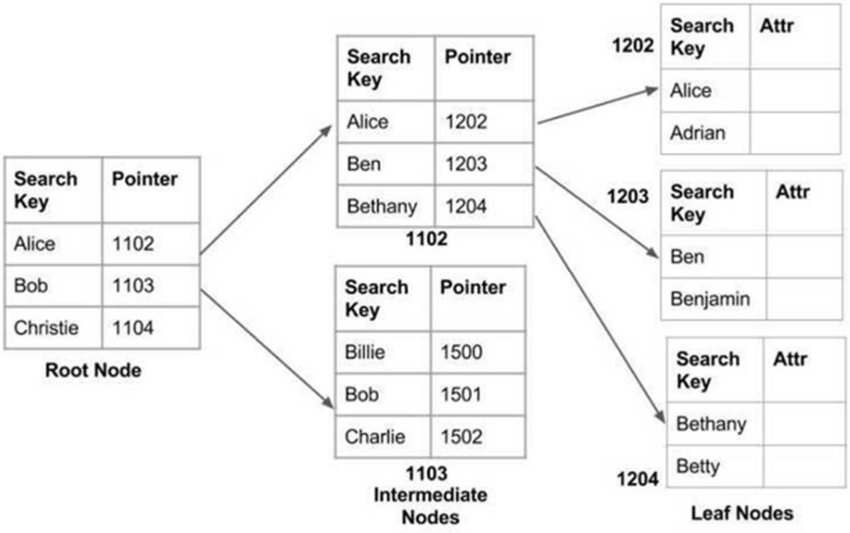
INDEX FILE	
SEMESTER	INDEX ADDRESS
1	
2	
3	
4	
5	

Data Blocks in Memory						
	100	Joseph	Alaiedon Township		20	200
	101					
	110	Allen	Fraser Township		20	200
	111					
	120	Chris	Clinton Township		21	200
	121					
	200	Patty	Troy		22	205
	201					
	210	Jack	Fraser Township		21	202
	211					
	300					

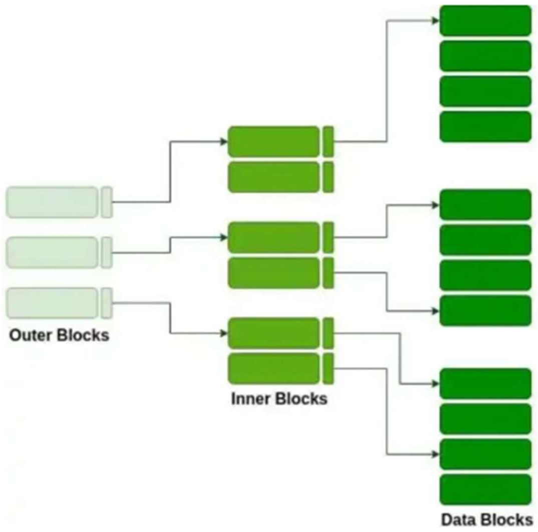
Primary Indexing: This is a type of Clustered Indexing wherein the data is sorted according to the search key and the primary key of the database table is used to create the index. It is a default format of indexing where it induces sequential file organization. As primary keys are unique and are stored in a sorted manner, the performance of the searching operation is quite efficient.

Non-clustered or Secondary Indexing: A non-clustered index just tells us where the data lies, i.e. it gives us a list of virtual pointers or references to the location where the data is actually stored. Data is not physically stored in the order of the index. Instead, data is present in leaf nodes. For eg. the contents page of a book. Each entry gives us the page number or location of the information stored. The actual data here(information on each page of the book) is not organized but we have an ordered reference(contents page) to where the data points actually lie. We can have only dense ordering in the non-clustered index as sparse ordering is not possible because data is not physically organized accordingly.

It requires more time as compared to the clustered index because some amount of extra work is done in order to extract the data by further following the pointer. In the case of a clustered index, data is directly present in front of the index.



Multilevel Indexing: With the growth of the size of the database, indices also grow. As the index is stored in the main memory, a single-level index might become too large a size to store with multiple disk accesses. The multilevel indexing segregates the main block into various smaller blocks so that the same can be stored in a single block. The outer blocks are divided into inner blocks which in turn are pointed to the data blocks. This can be easily stored in the main memory with fewer overheads.



Advantages of Indexing

- Improved Query Performance: Indexing enables faster data retrieval from the database. The database may rapidly discover rows that match a specific value or collection of values by generating an index on a column, minimizing the amount of time it takes to perform a query.
- Efficient Data Access: Indexing can enhance data access efficiency by lowering the amount of disk I/O required to retrieve data. The database can maintain the data pages for frequently visited columns in memory by generating an index on those columns, decreasing the requirement to read from disk.
- Optimized Data Sorting: Indexing can also improve the performance of sorting operations. By creating an index on the columns used for sorting, the database can avoid sorting the entire table and instead sort only the relevant rows.
- Consistent Data Performance: Indexing can assist ensure that the database performs consistently even as the amount of data in the database rises. Without

indexing, queries may take longer to run as the number of rows in the table grows, while indexing maintains a roughly consistent speed.

- By ensuring that only unique values are inserted into columns that have been indexed as unique, indexing can also be utilized to ensure the integrity of data. This avoids storing duplicate data in the database, which might lead to issues when performing queries or reports.

  Overall, indexing in databases provides significant benefits for improving query performance, efficient data access, optimized data sorting, consistent data performance, and enforced data integrity

Disadvantages of Indexing

- Indexing necessitates more storage space to hold the index data structure, which might increase the total size of the database.
- Increased database maintenance overhead: Indexes must be maintained as data is added, destroyed, or modified in the table, which might raise database maintenance overhead.
- Indexing can reduce insert and update performance since the index data structure must be updated each time data is modified.
- Choosing an index can be difficult: It can be challenging to choose the right indexes for a specific query or application and may call for a detailed examination of the data and access patterns.

Features of Indexing

- The development of data structures, such as B-trees or hash tables, that provide quick access to certain data items is known as indexing. The data structures themselves are built on the values of the indexed columns, which are utilized to quickly find the data objects.
- The most important columns for indexing columns are selected based on how frequently they are used and the sorts of queries they are subjected to. The cardinality, selectivity, and uniqueness of the indexing columns can be taken into account.
- There are several different index types used by databases, including primary, secondary, clustered, and non-clustered indexes. Based on the particular needs of the database system, each form of index offers benefits and drawbacks.
- For the database system to function at its best, periodic index maintenance is required. According to changes in the data and usage patterns, maintenance work

involves building, updating, and removing indexes.

- Database query optimization involves indexing, which is essential. The query optimizer utilizes the indexes to choose the best execution strategy for a particular query based on the cost of accessing the data and the selectivity of the indexing columns.
- Databases make use of a range of indexing strategies, including covering indexes, index-only scans, and partial indexes. These techniques maximize the utilization of indexes for particular types of queries and data access.
- When non-contiguous data blocks are stored in an index, it can result in index fragmentation, which makes the index less effective. Regular index maintenance, such as defragmentation and reorganization, can decrease fragmentation.