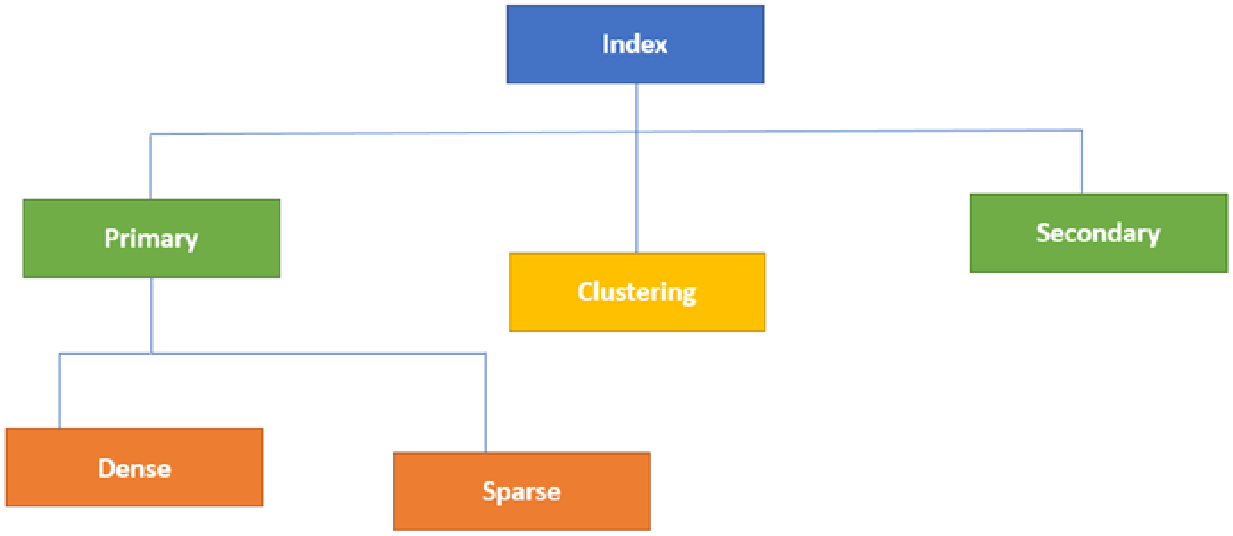
Indexing

Indexing is a data structure technique which allows you to quickly retrieve records from a database file. An Index is a small table having only two columns. The first column comprises a copy of the primary or candidate key of a table. Its second column contains a set of pointers for holding the address of the disk block where that specific key value stored.
An index -

- Takes a search key as input
- Efficiently returns a collection of matching records.

Types of Indexing



Indexing in Database is defined based on its indexing attributes. Two main types of indexing methods are:

- Primary Indexing
- Secondary Indexing

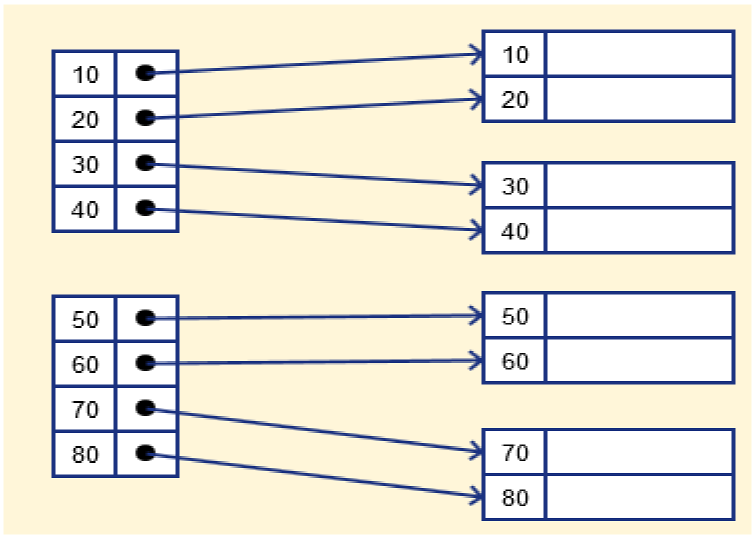
Primary Index

Primary Index is an ordered file which is fixed length size with two fields. The first field is the same a primary key and second, filed is pointed to that specific data block. In the primary Index, there is always one to one relationship between the entries in the index table.
The primary Indexing in DBMS is also further divided into two types.

- Dense Index
- Sparse Index

Dense Index

In a dense index, a record is created for every search key valued in the database. This helps you to search faster but needs more space to store index records. In this Indexing, method records contain search key value and points to the real record on the disk.

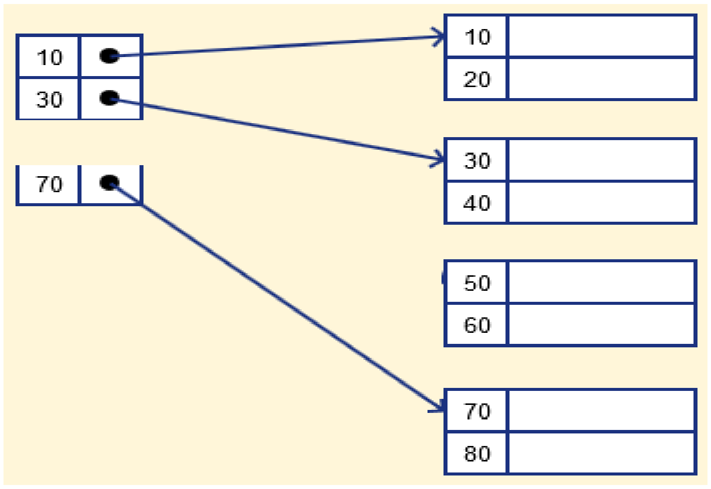


Sparse Index

It is an index record that appears for only some of the values in the file. Sparse Index helps you to resolve the issues of dense Indexing in DBMS. In this method of indexing technique, a range of index columns stores the same data block address, and when data needs to be retrieved, the block address will be fetched.

However, sparse Index stores index records for only some search-key values. It needs less space, less maintenance overhead for insertion, and deletions but It is slower compared to the dense Index for locating records.

Below is an database index Example of Sparse Index



Secondary Index

The secondary Index in DBMS can be generated by a field which has a unique value for each record, and it should be a candidate key. It is also known as a non-clustering index.

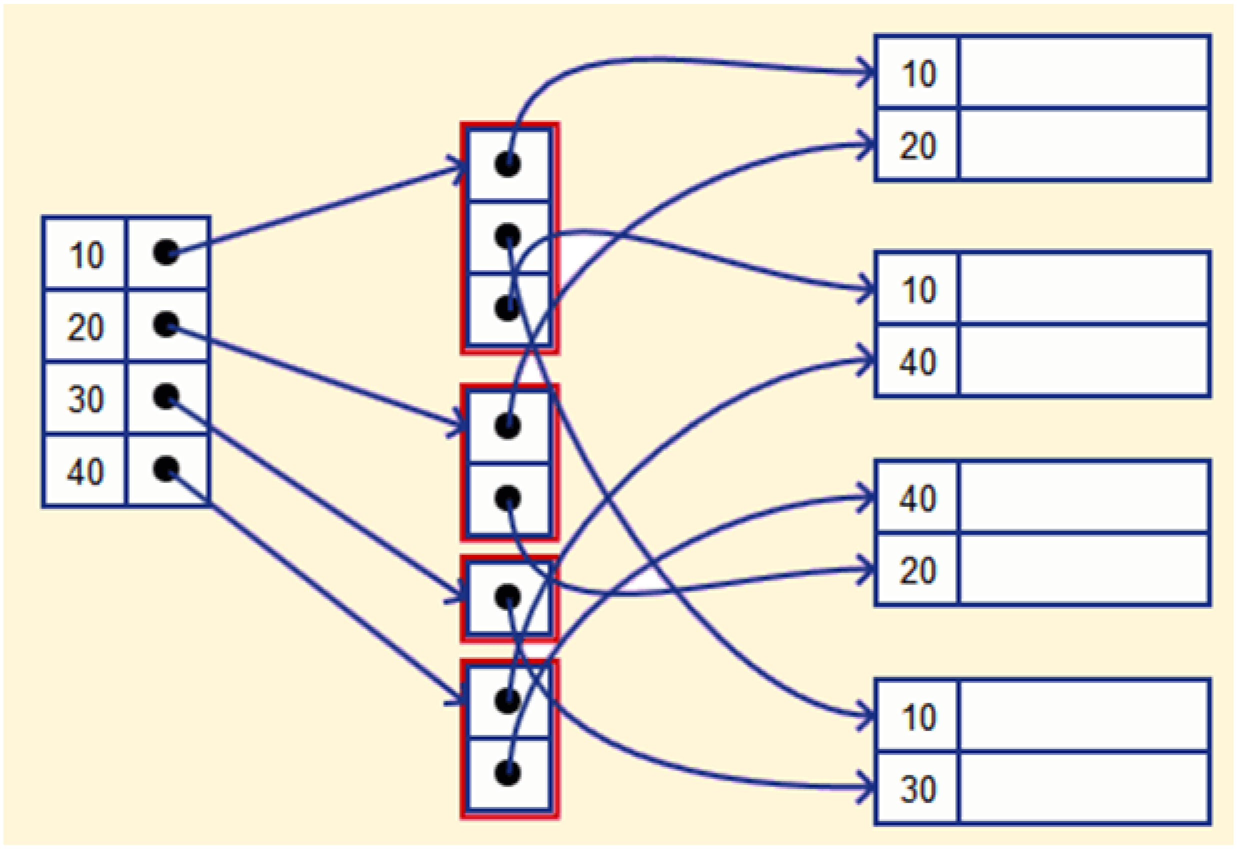
This two-level database indexing technique is used to reduce the mapping size of the first level. For the first level, a large range of numbers is selected because of this; the mapping size always remains small.

Example of secondary Indexing

Let's understand secondary indexing with a database index example:

In a bank account database, data is stored sequentially by acc_no; you may want to find all accounts in of a specific branch of ABC bank.

Here, you can have a secondary index in DBMS for every search-key. Index record is a record point to a bucket that contains pointers to all the records with their specific search-key value.



Clustering Index

In a clustered index, records themselves are stored in the Index and not pointers. Sometimes the Index is created on non-primary key columns which might not be unique for each record. In such a situation, you can group two or more columns to get the unique values and create an index which is called clustered Index. This also helps you to identify the record faster.

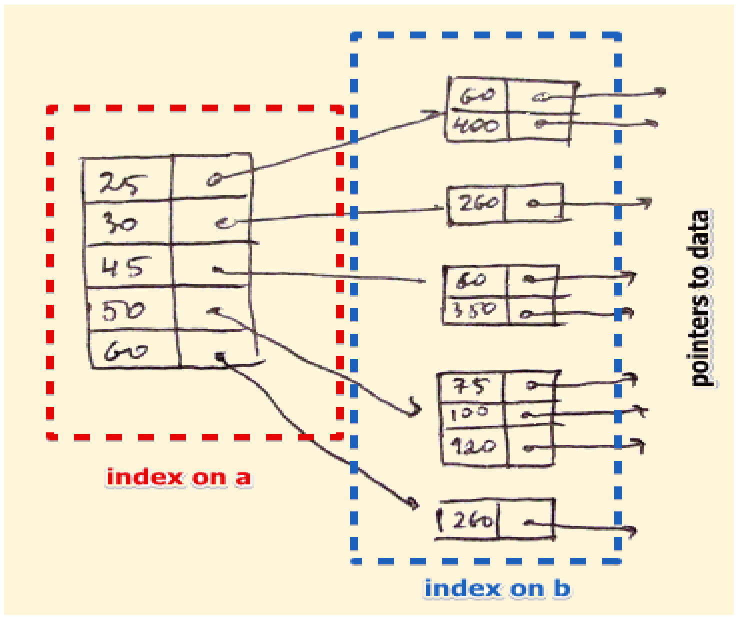
Example:

Let's assume that a company recruited many employees in various departments. In this case, clustering indexing in DBMS should be created for all employees who belong to the same dept.

It is considered in a single cluster, and index points point to the cluster as a whole. Here, Department _no is a non-unique key.

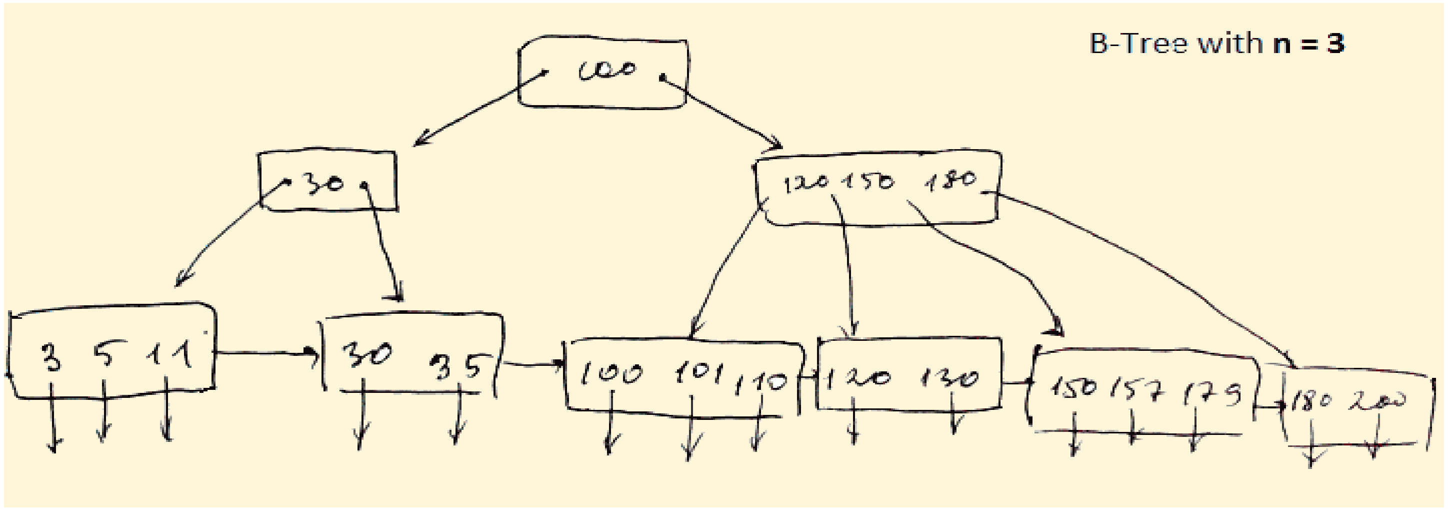
Multilevel Index

Multilevel Indexing in Database is created when a primary index does not fit in memory. In this type of indexing method, you can reduce the number of disk accesses to short any record and kept on a disk as a sequential file and create a sparse base on that file.



B-Tree Index

B-tree index is the widely used data structures for tree based indexing in DBMS. It is a multilevel format of tree based indexing in DBMS technique which has balanced binary search trees. All leaf nodes of the B tree signify actual data pointers.
Moreover, all leaf nodes are interlinked with a link list, which allows a B tree to support both random and sequential access.



- Lead nodes must have between 2 and 4 values.
- Every path from the root to leaf are mostly on an equal length.
- Non-leaf nodes apart from the root node have between 3 and 5 children nodes.
- Every node which is not a root or a leaf has between n/2 and n children.

Btree is an **example** of multilevel indexing. Record pointers will be present at leaf nodes as well as on internal nodes.

B+ tree we will have data (record pointers) only at leaf level.

Advantages of Indexing

- It helps you to reduce the total number of I/O operations needed to retrieve that data, so you don't need to access a row in the database from an index structure.
- Offers Faster search and retrieval of data to users.
- Indexing also helps you to reduce tablespace as you don't need to link to a row in a table, as there is no need to store the ROWID in the Index. Thus you will able to reduce the tablespace.
- You can't sort data in the lead nodes as the value of the primary key classifies it.

Disadvantages of Indexing

- To perform the indexing database management system, you need a primary key on the table with a unique value.
- You can't perform any other indexes in Database on the Indexed data.
- You are not allowed to partition an index-organized table.
- SQL Indexing Decrease performance in INSERT, DELETE, and UPDATE query.