For More Info Visit: whatisdbms.com
B-Tree Indexing in DBMS

Before we proceed to B-tree indexing lets understand what index means. An Index can be simply defined as an optional structure associated with a table cluster that enables the speed access of data. One can reduce the disk I/O by this. Creating an index, a small set of randomly distributed rows from the table can be retrieved. Without an index, it would be so difficult and database had to perform a full table scan to find a particular value. In full table scan, the database has to look at one row at a time to find desired value. One of the most common types of database index is B-trees (Balanced trees). This index is a default for many storage engines on MySQL. B-tree index is well ordered set of values that are divided into ranges.

There are many reasons for using B-trees such as

- Provides the best way to retrieve the wide range of queries.
- Speeds up the data access.
- Excellent for highly selective indexes and primary keys.
- Retrieves the data sorted by indexed columns, used as concatenated indexes.
- Provides universal applicability.
- They are storage-friendly, work on any layer of the storage hierarchy.

Let's take an example as to explain how B-tree indexing is helpful. Imagine books are arranged in the college library based on the alphabetical manner, the library has books of all departments such as Automobile, Aeronautical, Bio-tech, Chemical, Civil, Electronics and so on. After entering the library, you see that ground-floor contains books by department name A-G, first-floor H-N, second-floor O-U and third-floor V-Z. So based on your requirement you can quickly find the required book. Consider equivalent database search now, just Imagine books database table, with a B-tree index on the dpt_name column. To find your book of civil, you can simply perform below query

SELECT * FROM books WHERE

dpt_name = 'civil'

Initially, database examines the root node of B-tree index that defines four ranges to four floors in a library. Each node in the B-tree is a block. Blocks on every level of the tree except last are called branch blocks where entries are set of range and pointer to a block on the next level of tree. Those on last level are called leaf blocks which consist of a key value (an example is 'civil').

B-tree indexes have the sub-types. They are-

- Index-organized tables: It is different from the heap-organized due to the fact that data itself is the index here.
- Reverse key indexes: In this type, the bytes of an index are in a reverse order of the original index. For example, 346 is stored as 643.
- Descending indexes: It stores the data as the descending order in a particular column.

Unlike binary search trees, B-trees are optimized for systems that read and write a large block of data, they are a good example of data structure for external memory and commonly used in databases and file systems. A binary tree has two child nodes (left and right node), the time required to find a node in a tree is proportional to its height and balance (balance here indicates both sub-trees have almost the same height) which makes a good indexing data structure where the access time is a log of number of nodes. It is better to have the whole index structure in memory but with large sizes, it is not possible to place all the indexes in memory. For a binary tree the branching factor is 2 where the nodes are highly granular, one has to do many round trips to arrive at a final node but B-tree, on the other hand has high branching factor and so it is very easy to get required node.