DBMS 3: Transactions + Indexing

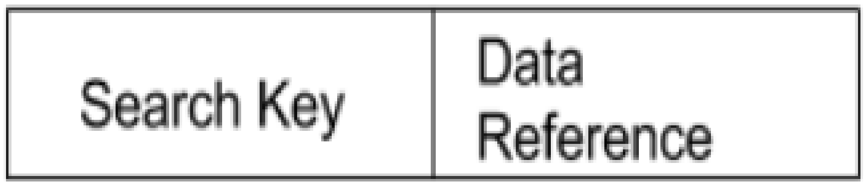
Indexing

Indexing is a way to optimize performance of a database by minimizing the number of disk accesses required when a query is processed.

An index or database index is a data structure which is used to quickly locate and access the data in a database table.

Indexes are created using some database columns.

- The first column is the Search key that contains a copy of the primary key or candidate key of the table. These values are stored in sorted order so that the corresponding data can be accessed quickly.
- The second column is the Data Reference 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

B-Tree

1. B-Tree is a self-balancing search tree. In most of the other self-balancing search trees, it is assumed that everything is in the main memory. To understand the use of B-Trees, we must think of the huge amount of data that cannot fit in the main memory.
2. When the number of keys is high, the data is read from the disk in the form of blocks. Disk access time is very high compared to main memory access time.
3. The main idea of using B-Trees is to reduce the number of disk accesses. Most of the tree operations (search, insert, delete, max, min, ..etc) require O(h) disk accesses where h is the height of the tree.
4. The height of B-Trees is kept low by putting the maximum possible keys in a B-Tree node. Generally, a B-Tree node size is kept equal to the disk block size. **Since h is low for B-Tree, total disk accesses for most of the operations are reduced significantly compared to balanced Binary Search Trees.**

Properties of B-Tree

1) All leaves are at the same level.

2) A B-Tree is defined by the term *minimum degree* ‘t’. The value of t depends upon disk block size.

3) Every node except the root must contain at least t-1 keys. The root may contain a minimum of 1 key.

4) All nodes (including root) may contain at most 2t – 1 keys.

5) Number of children of a node is equal to the number of keys in it plus 1.

6) All keys of a node are sorted in increasing order. The child between two keys k1 and k2 contains all keys in the range from k1 and k2.

7) Like other balanced Binary Search Trees, time complexity to search, insert and delete is O(Logn).