Indexing is process of associating a key with location of corresponding data record

Create index on field in table 🡪 creates another data structure that holds field value and pointer to the record it relates to.

* Index structure is sorted
* Allowing binary searches to be performed on it
* Indices require additional space on disk along the table using MyISAM engine.
* It reaches size limit if more fields withing same table are indexed.

Why indexing

Sample Query

SELECT  
company\_id,  
units,  
unit\_cost  
FROM  
costing  
WHERE  
company\_id = 18

Table

Description automatically generated

Costing table with unit\_cost column

* When table has more records , it becomes time consuming process

How index works ?

* Index is data structure (most commonly its B-tree its balances tree , not binary tree) that stores the value for specific column in a table

Cuts down the time complexity O(n) to o(log n)

* B-tree is data structure of choice to implement Database indexes
* B-Tree is generalization of binary search tree (BST)
* BST is sorted data structure composed of nodes

**But Why only B-Tree?**

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. B-tree is a fat tree. The height of B-Trees is kept low by putting the maximum possible keys in a B-Tree node. Generally, the B-Tree node size is kept equal to the disk block size. Since the height of the B-tree is low so total disk accesses for most of the operations are reduced significantly compared to balanced Binary Search Trees like AVL Tree, Red-Black Tree, ..etc.

**Properties of B-Tree:**

* All leaves are at the same level.
* A B-Tree is defined by the term minimum degree ‘t’. The value of t depends upon disk block size.
* Every node except the root must contain at least t-1 keys. The root may contain a minimum of 1 key.
* All nodes (including root) may contain at most 2t-1 keys.
* The number of children of a node is equal to the number of keys in it plus 1.
* 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.
* B-Tree grows and shrinks from the root which is unlike Binary Search Tree. Binary Search Trees grow downward and also shrink downward.
* Like other balanced Binary Search Trees, the time complexity to search, insert and delete is O(Logn).

**Advantages:**

* Speed up SELECT query
* Helps to make a row unique or without duplicates(primary,unique)
* If index is set to fill-text index, then we can search against large string values. for example to find a word from a sentence etc.

**Disadvantages**

* Indexes take additional disk space.
* indexes slow down INSERT,UPDATE and DELETE, but will speed up UPDATE if the WHERE condition has an indexed field. INSERT, UPDATE and DELETE becomes slower because on each operation the indexes must also be updated.

**What is the difference between a key and an index in DBMS? Can’t you use the primary key to find any piece of data?**

*The analogy*:

Each library(database) has books (tables). Each book(table) has pages(rows). Each page(row) has a unique page number to identify it (key-value), so you could say that the *key* to finding a certain page is through its number. But if you want to find a certain page, you don’t know exactly where to look for it, right? You’ll have to turn page by page until you reach the page number you were looking for. Wouldn’t it be easier if you had several bookmarks that tell you on which page they were placed? If you look for page 1267 and you have a bookmark on page 1266 then you go straight to that bookmark and now you only have one page to turn to get to your page. The index is like a set of bookmarks for a book. It helps you find the desired rows much faster.