Digital Assignment - 3

Reg No : 19BCS0012
Name : G.N.L. Harsh
Course : E-DBMS
Date : 03/12/2024

1. Explain in detail about B Tree & B+ Tree..

BTree

* B Tree is a self-balancing tree, & it is a m-way tree where m defines the orders of the tree. B Tree is a generalization of the Binary search tree, in which a node can have more than one key & more than two children depending upon the value of m.

* B-Tree is known as a self-balancing tree as its nodes are sorted in the inorder traversal. In B-Tree a node can have more than two children.

Height = $\log_{m/2}(N)$ (M - order of tree, N - number of nodes)

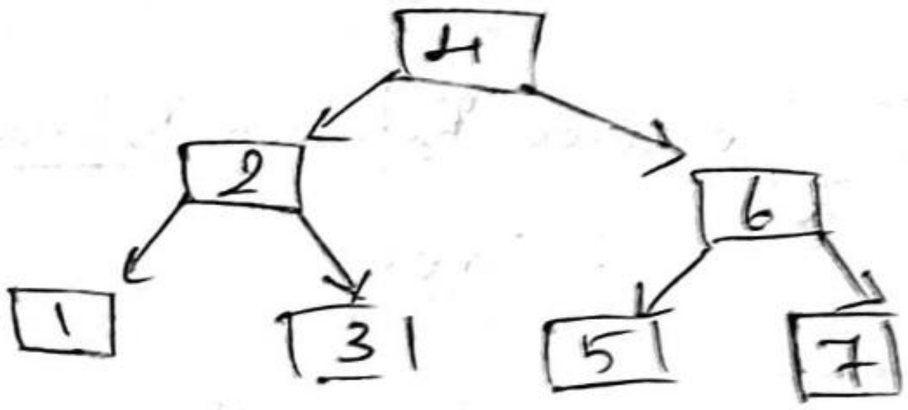
* Height is adjusted automatically at each update. In B-tree data is sorted in specific order (lowest-left, highest-right). To insert data in B-tree it is complicated than Binary tree.

* Condition:

* All the leaf nodes of the Btree must be at the same level.

* Above the left nodes, then should be no empty sub tree.

* B-tree height should lie as low as possible

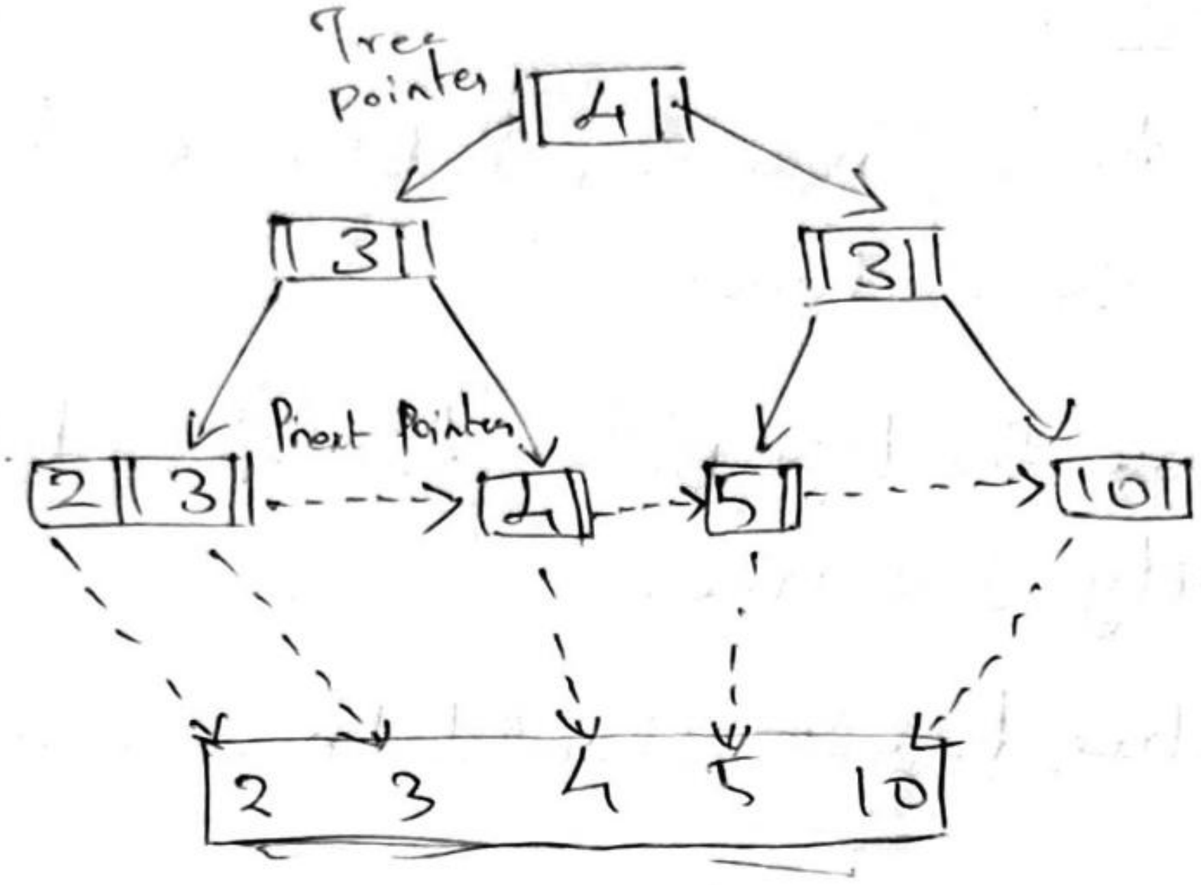


B+ Tree:

* B+ Tree eliminates drawback B-tree used for indexing by storing data pointers only at the leaf nodes of the tree. Thus, the structure of leaf nodes of a B+tree is quite differs from the structure of internal nodes of a B+ tree.

* It may be noted here that since data pointers are present only at leaf nodes the leaf nodes must necessarily store all the key values along with their corresponding data pointers to the disk file block, in order to access to the records.

B+ Tree



Difference Between B Tree & B+ Tree

B Tree	B+ Tree
1. All internal & leaf nodes have data pointers	1. Only leaf nodes have data pointers.
2. Since all keys are not available at leaf, search often takes more time.	2. All keys are at leaf nodes, hence search is faster and accurate.
3. No duplicate of key is maintained in the tree	3. Duplicate of keys are maintained and all nodes are present at leaf.
4. Insertion takes more time & it is not	4. Insertion is easier & the results are always the same.
5. Deletion of internal node is very complex & tree has to undergo lot of transformation.	5. Deletion of any node is easy because all nodes are found at leaf.

Why is the B+ Tree used

(usually Preferal as an access structure to a data file)

A/ A B+ Tree is used to store the records very efficiently by storing the records in an indexed manner using the B+tree index structure. Due to the multi-level indexing, the data accusing becomes faster and easier.

A/ B+ Tree has on root, any number of intermediary nods (usually one) & a leaf node. Here all leaf nodes will have the active records stored. Intermediary nodes will have only pointers to the leaf nodes, it not has any data. Any node will have only two leaves this is the Basics of any B+ Tree.

2. Explain about Problems to be considered in Database Tunning: Index, Tunning & Tuning Queries in Relational system.

→ Data base Tuning :-

* Database Tuning describes a group of activities used to optimize & homogenize the performance of a database.

* It usually overlaps with query tuning, but refers to design of the database files selection of the DBMS application, & configuration of the database environment (OS, CPU, etc..).

Index tuning :-

* Query Performance as well as speed improvement of a database can be done using indexes.

* The Process of enhancing the selection of index as to called Index Tuning.

* Index Tuning is Part of database tuning for selecting & creating indexes. The index tuning goal is to

→ Query Tuning :-

Ans SQL Tuning or SQL Query Tuning - SQL Statements are used to modries data from the data base. We can get same result by writing different SQL queries.

Ans A Query issues too many disk access (ex: an external match data from the database we can get same results by writing different SQL queries.

Problems to be considered in Tuning :-

1. How to avoid excessive lock contention?
2. How to minimize overhead of logging & unnecessary dumping of data.
3. How to optimize buffer size and Scheduling of process?
4. How to allocate resources such as disks, RAM & Process for most efficient utilization?

1) How to avoid excessive lock contention?

- Avoid situations in which many processes are attempting to perform updates or inserts on the same data page.
- Avoid transaction that include user interaction.
- Keep transaction that modify data as short as Possible.
- Keep transaction in one batch.

2) Reducing logging Overhead.

- All changes to regular data & index pages are written to the log buffer before being written to disk by longer Process. SQL statement processing must wait for log data to be written to Disk..

→ on commit
→ until changes are made to meta data.
→ when the log buffer is full

Force these should be in node.

8.) Optimizing Buffer Size & scheduling of Process.

* If the buffer size are too small, then small Processing delays will cause the buffer to fill.

* For Optimizing the scheduling, identifying of priority is important.

* Enlarge the buffers size using the data flow properties.

* Identifying how to allocate resources such as disks, RAM & Processors for most efficient utilization.

a.) lack of Resource monitoring.

* As a database administration, there are several implementations to optimize a database.

* Excessive load leads to excessive Consumption of resources. It includes excessive usage of CPU, memory and even IO components in some instances.