B+ TREE

By Abhishek RS

CONCEPT

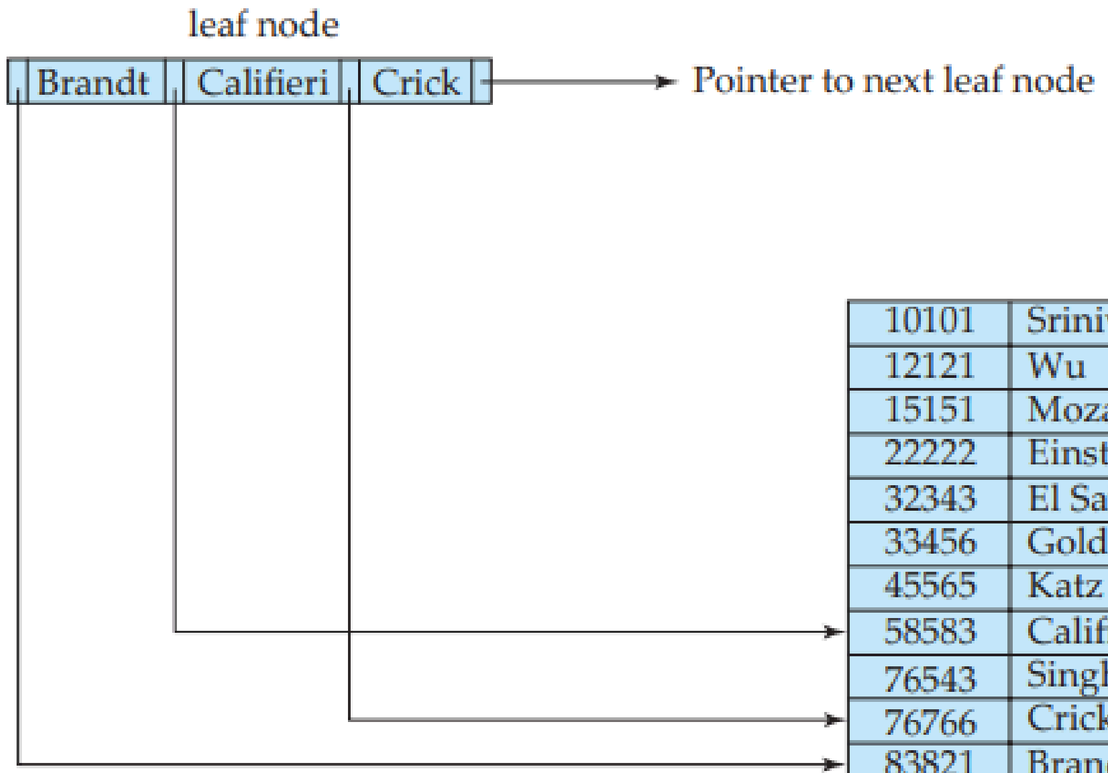
- Used to implement Database indexes.
- B+ Trees are self-balancing, which means that as data is added or removed from the tree, it automatically adjusts itself to maintain a balanced structure.
- This ensures that the search time remains relatively constant, regardless of the size of the tree.
- In B+ tree the leaf nodes contains the actual data pointers.
- All leaf nodes remain at the same height.
- Leaf nodes are linked using Linked List.
- Therefore it supports both random and sequential access.

STRUCTURE OF B+ TREE

- A B+-tree index is a multilevel index, but it has a structure that differs from that of the multilevel index-sequential file

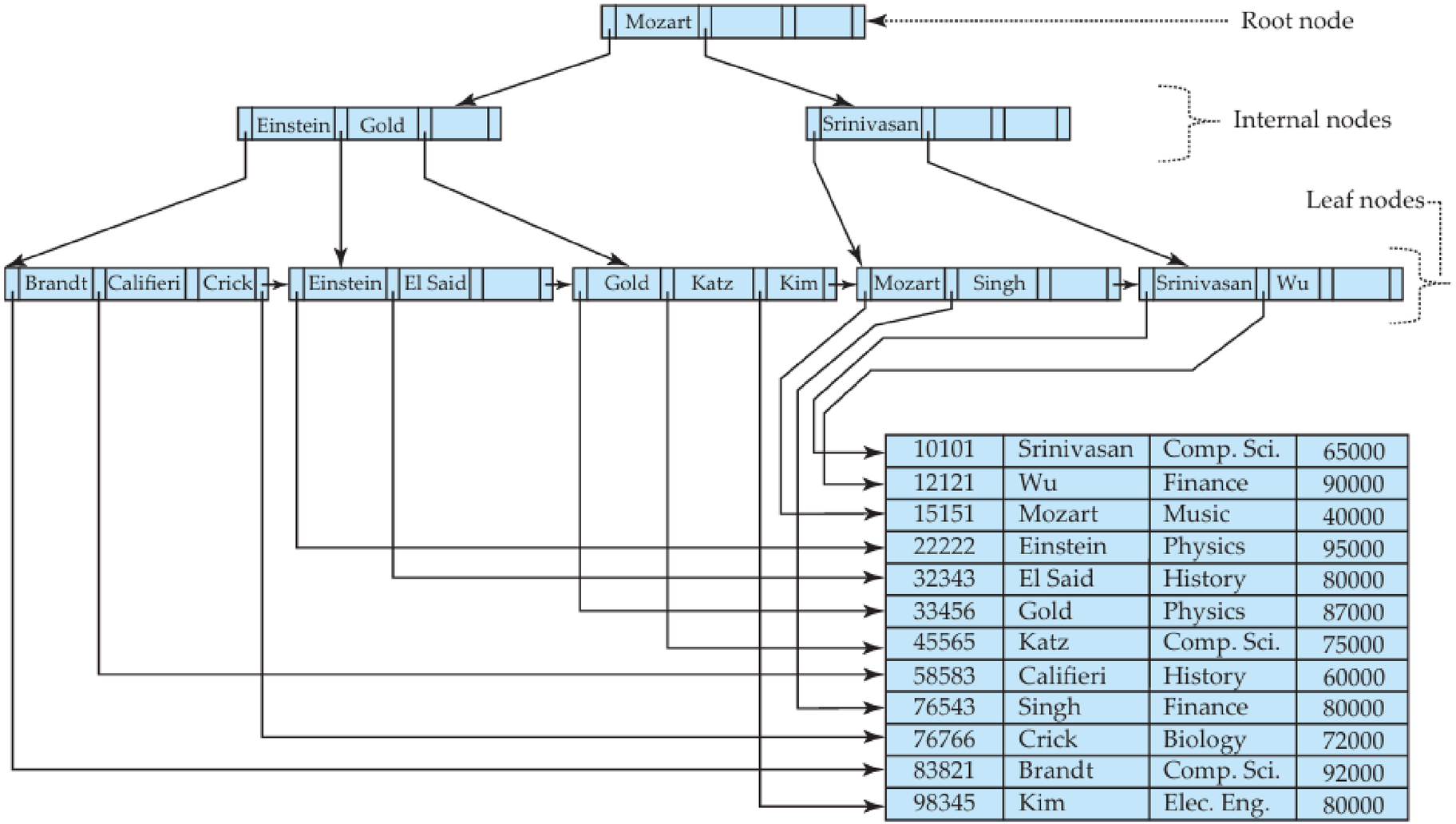
- This shows a typical node of a B+- tree. It contains up to n − 1 search-key values K1, K2,..., Kn − 1, and n pointers P1, P2,..., Pn. The search-key values within a node are kept in sorted order; thus, if i < j, then Ki < K j .

SIMPLE IMPLEMENTATION OF B+ TREE



10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	80000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	60000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

instructor file



10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	80000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	60000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

ADVANTAGES

- Data stored in a B+ tree can be accessed both sequentially and directly.
- It takes an equal number of disk accesses to fetch records.
- A B+ tree with ‘l’ levels can store more entries in its internal nodes compared to a B-tree having the same ‘l’ levels.

DISADVANTAGES

- Because of internal nodes storing multiple keys and pointers, B+ Trees consume more memory compared to simpler indexing methods like hashing.
- Maintaining the balance of the tree requires restructuring operations (like node splitting/merging), which may slow down performance in write-heavy databases.
- Though searching in a B+ Tree is efficient (O(log n)), accessing actual data requires traversing from the root to the leaf node, which may add overhead.

THANK YOU