M.Sc. (Computer Science)- Second Semester
Subject Name- “RDBMS”
Paper-I (Video Part-4)

By- Prof. Dileep Kumar Sahu

Assistant Professor

Department of Computer Application

Govt. Vishwanath Yadav Tamaskar Post Graduate Autonomous College, Durg (C.G.)

Email ID: dileepksahu20@gmail.com

Unit- V: Relational Database Design

BY - PROF. DILEEP KUMAR SAHU

Content

- B+ Tree File Organization

B+ Tree File Organization

- Most implementations of a dynamic multilevel index use a variation of the Btree data structure, called a B+ tree.
- In a B-tree, every value of the search field appears once at some level in the tree, along with a corresponding data pointer.
- In a B+ tree, data pointers are stored only at the leaf nodes of the tree; hence, the structure of leaf nodes differs from that of internal nodes.
- The leaf nodes have an entry for every value of the search field, along with a corresponding data pointer to the block containing the record with that value if the search field is a key field.

B+ Tree File Organization

- For a non-key search field, the pointer points to a block containing pointers to the data file records, creating an extra level of indirection (as we have seen before).
- The leaf nodes of a B+ tree are usually linked together to provide ordered access on the search field to the records.
- These leaf nodes are similar to the first level of a multilevel index. Internal nodes correspond to the other levels of the multilevel index.
- Some search field values from the leaf nodes are duplicated in the internal nodes to guide the search.

B+ Tree File Organization: Definition

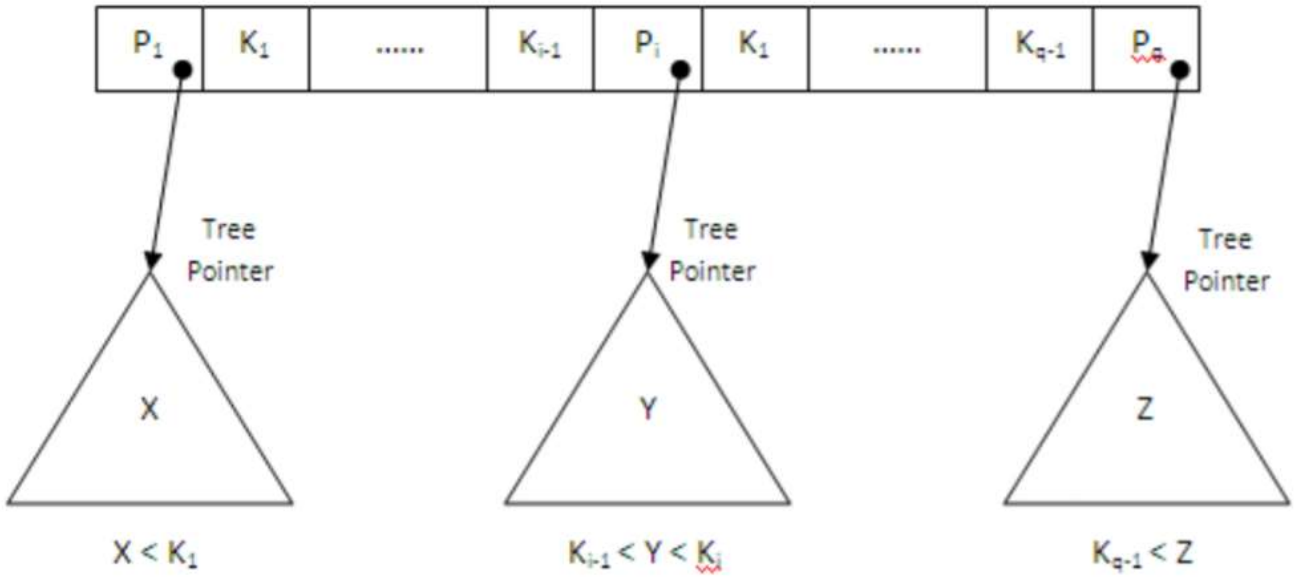
- The structure of the internal nodes of a B+-tree of order p is defined as follows:
  - Each internal node is of the form: where q <= p and each Pi is a tree pointer.
  - Within each internal node, K1 < K2 < …… < Kq-1.
  - For all search field values X in the sub-tree pointed at by Pi, we have Ki-1 < X <= Ki for 1 < i < q;
  - X <= Ki for i = 1; Ki-1 < X for i = q.
  - Each internal node has at most p tree pointers and p –1 search key values.
  - Each internal node, except the root, has at least #(p/2)# tree pointers and #p/2)# - 1 search key values (i.e. must not be less than half full). The root node has at least two tree pointers (one search key value) if it is an internal node.
  - An internal node with q tree pointers, q # p, has q – 1 search key field values.

B+ Tree File Organization: Definition

- The structure of the leaf nodes of a B+ tree of order p is defined as follows:
  - Each leaf node is of the form <, , ……, , Pnext> where q # p. Each Pri is a data pointer, and Pnext points to the next leaf node of the B+ tree.
  - Within each internal node, K1 < K2 < …… < Kq-1.
  - Each Pri is a data pointer pointing to the record whose search field value is Ki, or to a file block containing the record (or to a block of record pointers that point to records whose search field value is Ki, if the search field is not a key).
  - Each leaf node has at least #(p/2)# values (entries).
  - All leaf nodes are at the same level.

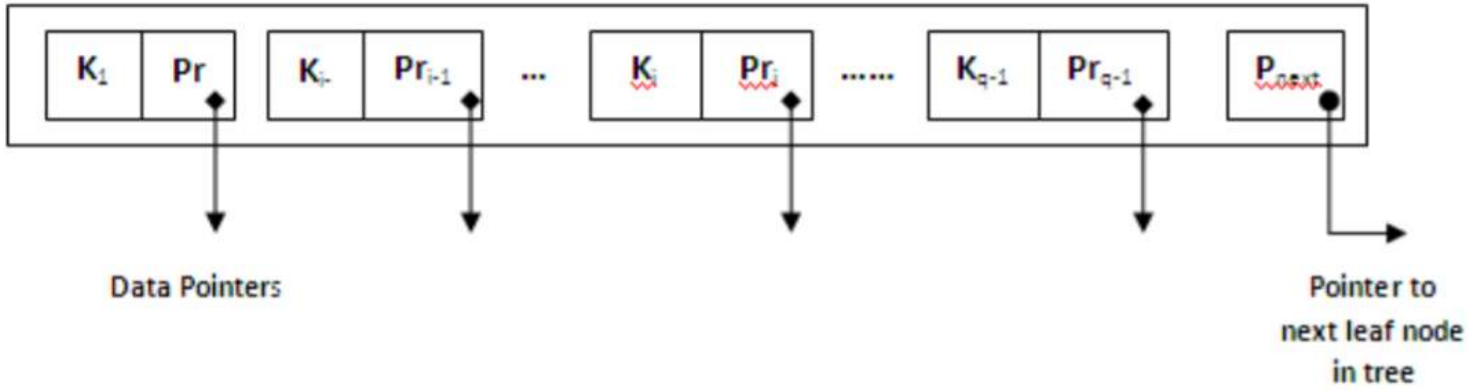
B+ Tree File Organization: Definition

- The following figure depicts the general structure of an internal node of a B+ tree of order p:



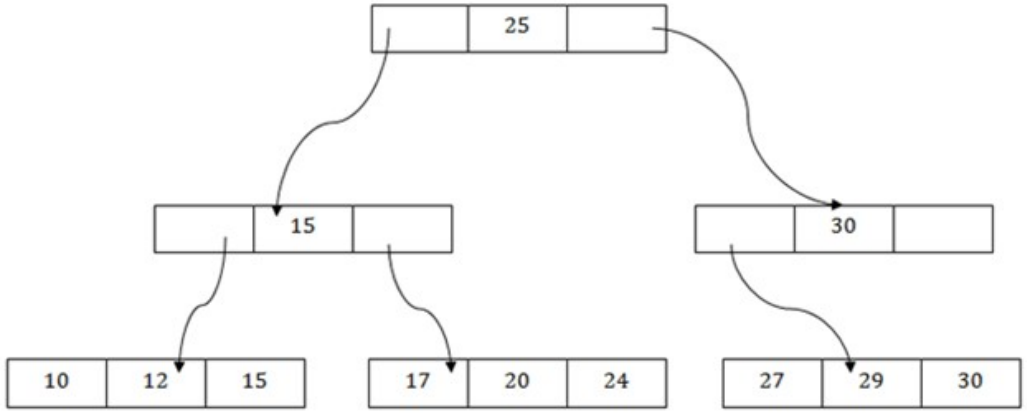
B+ Tree File Organization: Definition

- And the following figure depicts the general structure of a leaf node of a B+ tree of order p:



B+ Tree File Organization: Example

- There is one root node of the tree, i.e., 25.
- There is an intermediary layer with nodes.
- They do not store the actual record. They have only pointers to the leaf node.
- The nodes to the left of the root node contain the prior value of the root and nodes to the right contain next value of the root, i.e., 15 and 30 respectively.
- There is only one leaf node which has only values, i.e., 10, 12, 17, 20, 24, 27 and 29.
- Searching for any record is easier as all the leaf nodes are balanced.
- In this method, searching any record can be traversed through the single path and accessed easily.



B+ Tree File Organization: Example

Advantages:

- In this method, searching becomes very easy as all the records are stored only in the leaf nodes and sorted the sequential linked list.
- Traversing through the tree structure is easier and faster.
- The size of the B+ tree has no restrictions, so the number of records can increase or decrease and the B+ tree structure can also grow or shrink.
- It is a balanced tree structure, and any insert/update/delete does not affect the performance of tree.

Limitations:

- This method is inefficient for the static method.

Difference between B Tree and B+ Tree

- In a **B tree**, search keys and data are stored in internal or leaf nodes. But in **B+-tree**, data is stored only in leaf nodes.
- Searching any data in a **B+ tree** is very easy because all data are found in leaf nodes. In a **B tree**, data cannot be found in leaf nodes.
- In a **B tree**, data may be found in leaf nodes or internal nodes. Deletion of internal nodes is very complicated. In a **B+ tree**, data is only found in leaf nodes. Deletion of leaf nodes is very easy as it can be directly deleted.
- Insertion in **B tree** is more complicated than **B+ tree**.
- **B+ trees** store redundant search key but **B tree** has no redundant value.
- In a **B+ tree**, leaf nodes data are ordered as a sequential linked list but in **B tree** the leaf node cannot be stored using a linked list. Many database systems' implementations prefer the structural simplicity of a B+ tree.

The **basic difference** lies between how they make use of the **internal storage**.

Thank You

BY - PROF. DILEEP KUMAR SAHU