CSE3103 : Database
FALL 2020

Nazmus Sakib
Assistant Professor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

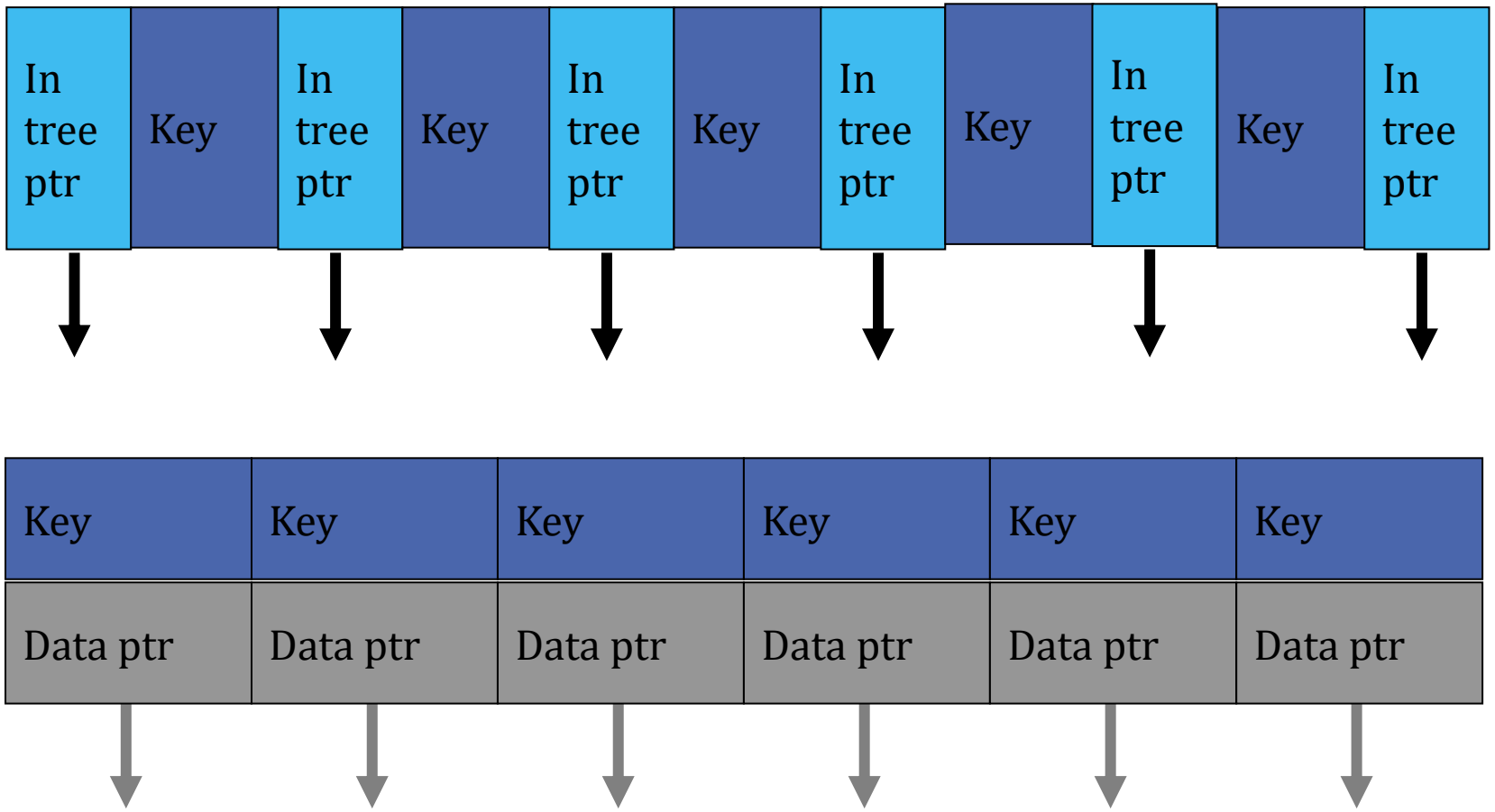
B+ trees

- Variant of B trees
- Two types of nodes
  - Internal nodes have no data pointers
  - Leaf nodes have no in-tree pointers
    - Were all null!

Advantages of B+ tree

- Keeps keys in sorted order for sequential traversing
- Uses a hierarchical index to minimize the number of disk reads
- Uses partially full blocks to speed insertions and deletions
- Keeps the index balanced with a recursive algorithm
- In addition, a B+ tree minimizes waste by making sure the interior nodes are at least half full. A B+ tree can handle an arbitrary number of insertions and deletions.

B+ tree nodes



Properties

- If m is the order of the tree
  - Every internal node has at most m children.
  - Every internal node (except root) has at least $\lceil m / 2 \rceil$ children.
  - The root has at least two children if it is not a leaf node.
  - Every leaf has at most $m - 1$ keys
  - An internal node with k children has $k - 1$ keys.
  - All leaves appear in the same level

Insertions

- **def** insert (entry) :
  Find target leaf *L*
  if *L* has less than *m* – 2 entries :
    add the entry
  else :
    Allocate new leaf L'
    Pick the m/2 highest keys of *L* and move them to *L'*
    Insert ***highest key*** of *L* and corresponding address leaf into the parent node
    If the parent is full :
      Split it and add the middle key to its parent node
    Repeat until a parent is found that is not full

Deletions

- **def** delete (record) :

  Locate target leaf and remove the entry

  If leaf is less than half full:

    Try to re-distribute, taking from sibling (adjacent node with same parent)

    If re-distribution fails:

      Merge leaf and sibling

      Delete entry to one of the two merged leaves

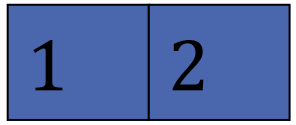
      Merge could propagate to root

Insertions

- Assume a B+ tree of degree 3.

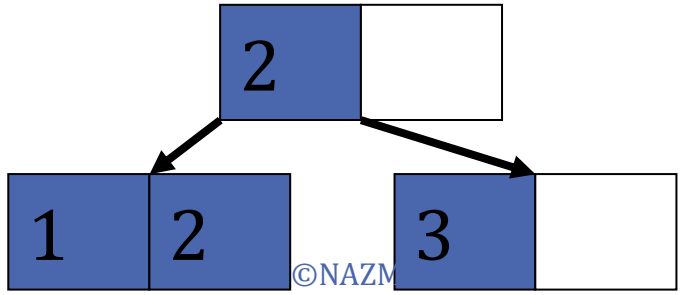
- Step 1: insert 1

- Step 2: insert 2



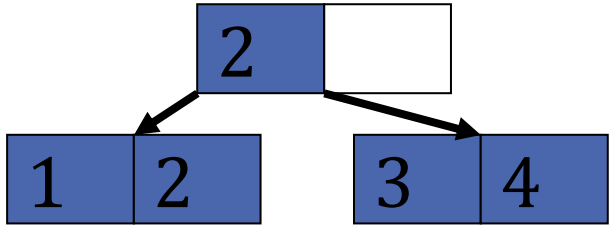
- Step 3: insert 3

  - Split node in middle

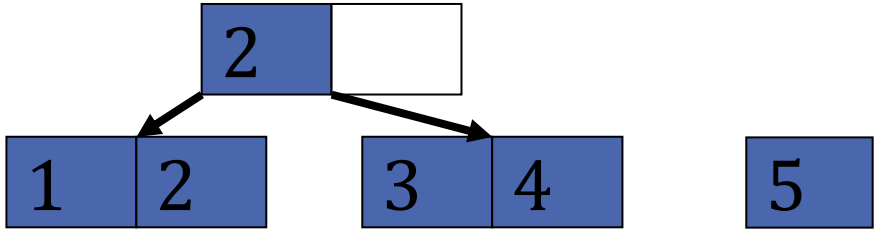


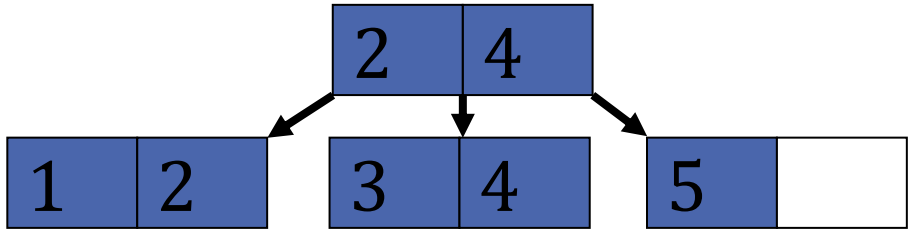
Insertions

- Step 4: insert 4



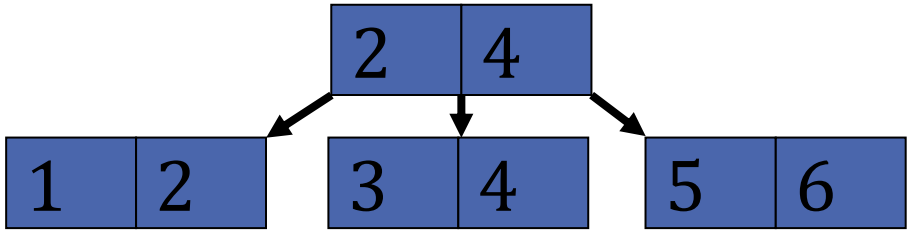
- Step 5: insert 5
  - Split
  - Move up



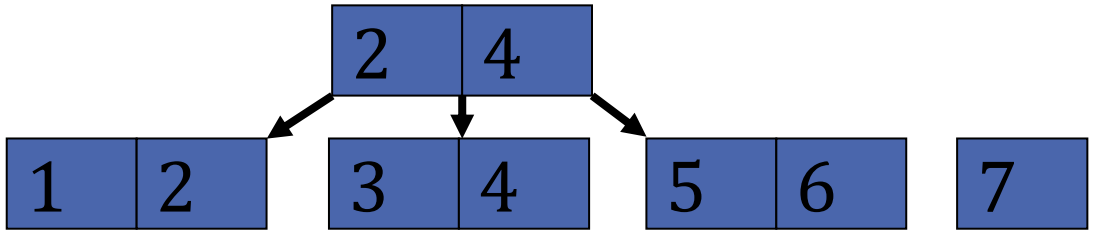


Insertions

- Step 6: insert 6

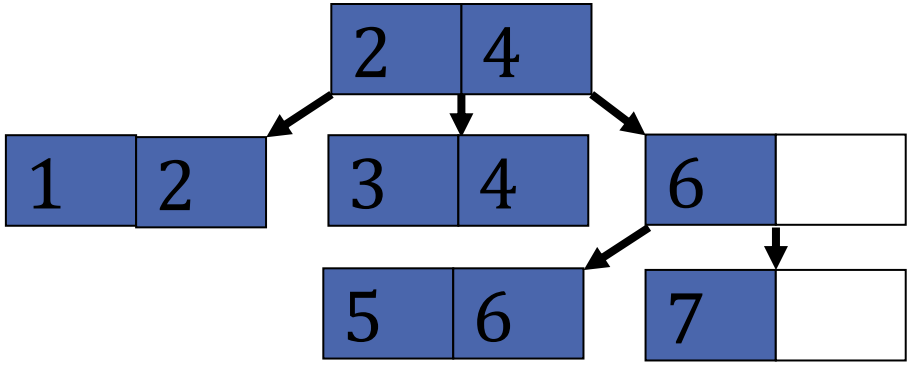


- Step 7: insert 7

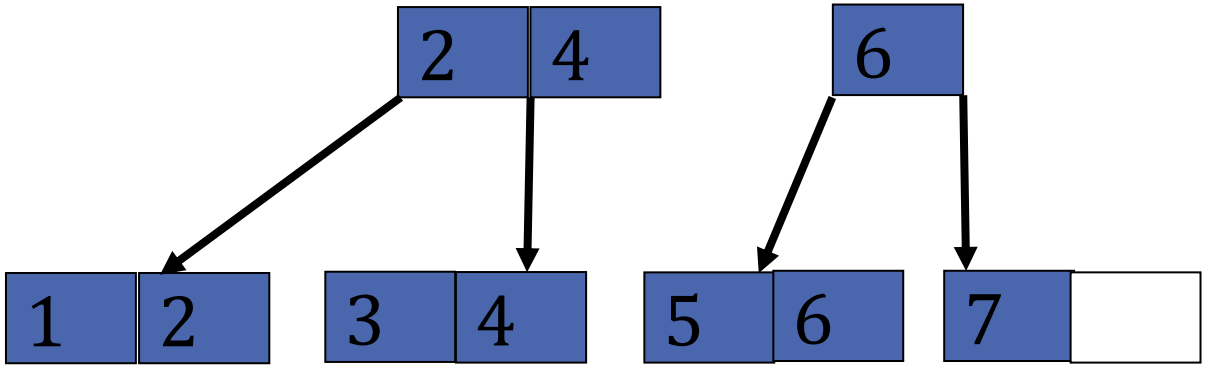


Step 7 continued

- Split



- Promote



Step 7 continued

- Split after the promotion