Project 1: B+ tree implementation

Jay D. Bodra

ITLAB@UTA

Email: jay.bodra@mavs.uta.edu

URL: itlab.uta.edu

Talk Outline

- Understanding B+ tree concept in minibase
- Algorithm for _Delete()
- Demo

Understanding B+ tree concept in minibase

- HeaderPage
  - Header Page is the root node
  - There is only one Header Page
  - When you run the program a header page is created by the BTreeFile(String filename, int keytype, int keysize, int delete_fashion) constructor which will be pointing to an INVALID_PAGE

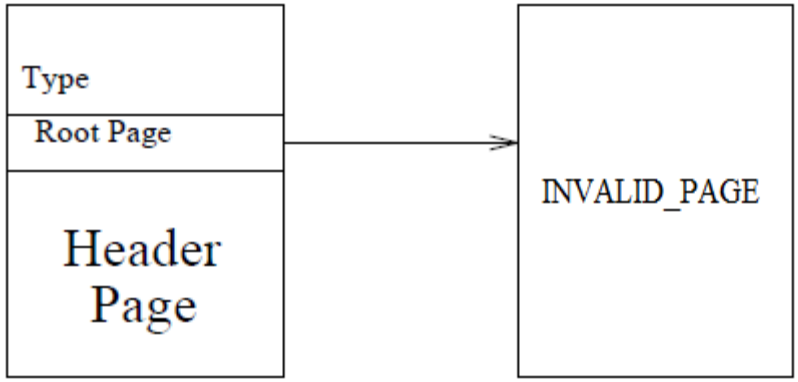
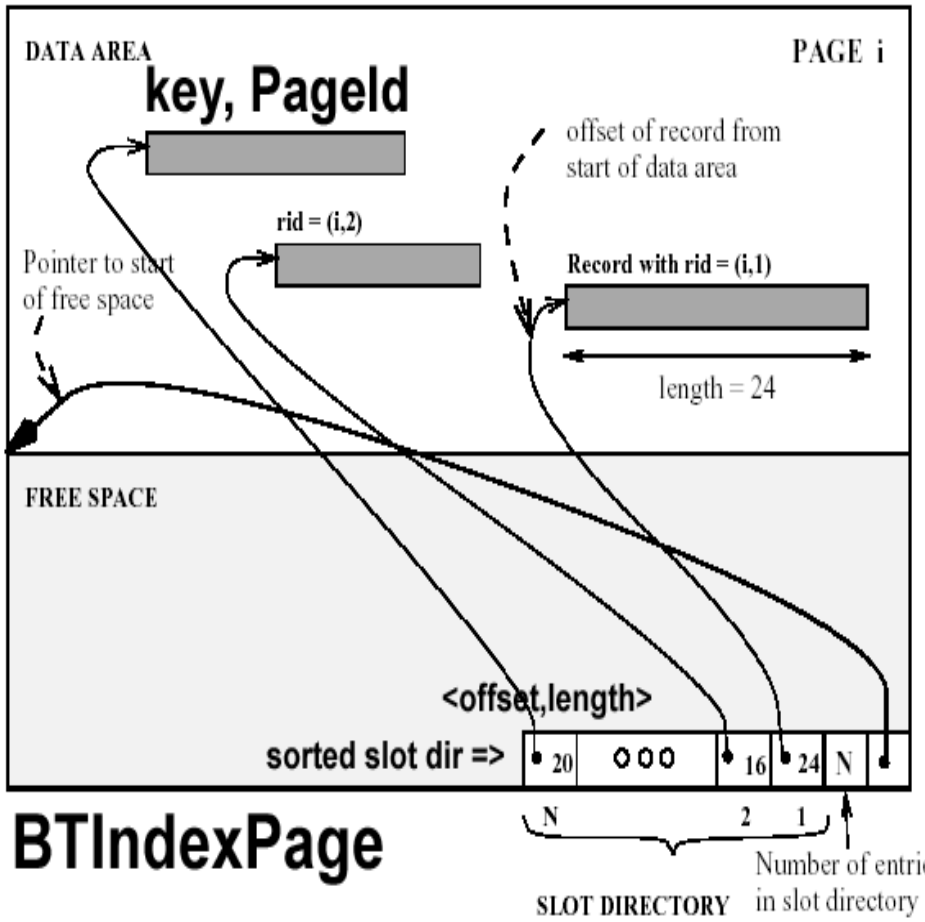


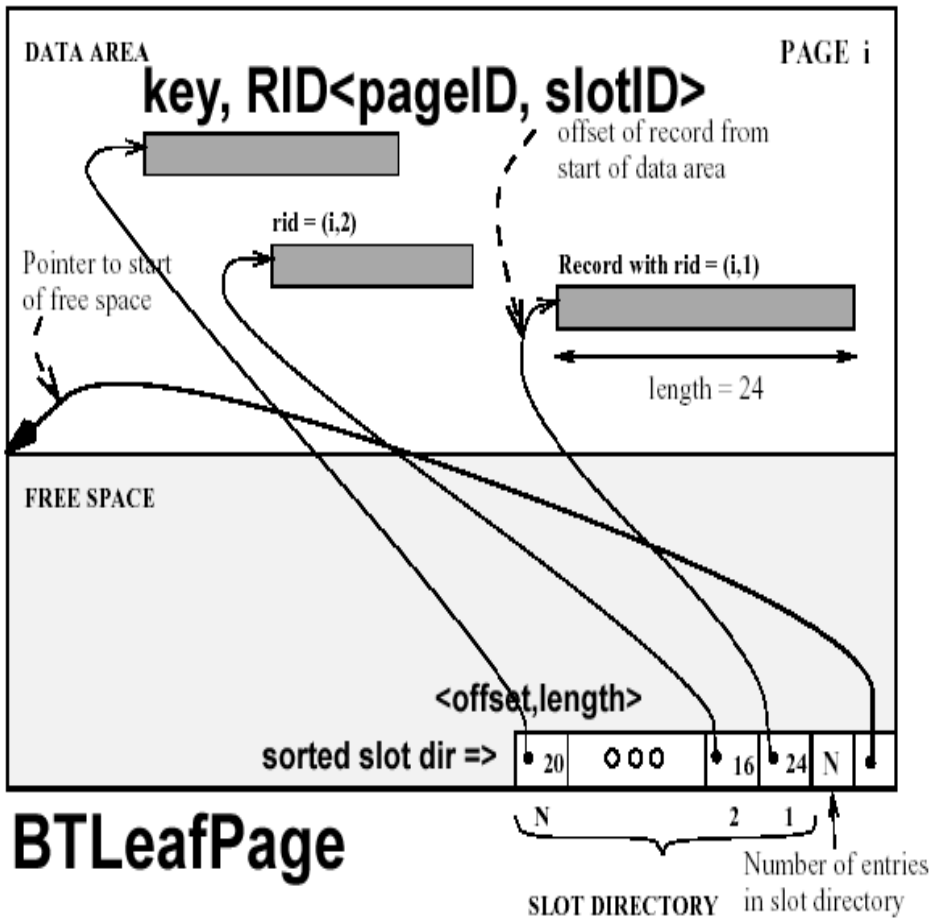
Figure 1: Initial B+ tree headerPage points to an INVALID_PAGE

Understanding B+ tree concept in minibase(1)

- IndexPage
  - Index Page is represented in the form of <key, PageId>
  - It points to the left leaf Page by its pageId and the rest leafPages that it points to are stored in form of <key, PageId>
- LeafPage
  - Leaf Page is represented in the form of <key, PageNo, SlotNo>
  - Leaf Pages are organized as a doubly link list
  - Previous pointer of Left most leaf page points to INVALID_PAGE
  - Next Pointer of the right most leaf page points to INVALID_PAGE

Understanding B+ tree concept in minibase(2)





An overview of given code

- To create a new Leaf Page or IndexPage
  - BTLeafPage newRootPage=new BTLeafPage(headerPage.get_keyType()); defined under BTLeafPage class
  - What will the code for IndexPage?
- PageId newRootPageId = newRootPage.getCurPage();
  - getCurPage() is a method defined under HFPage Class which returns the page number of current page
- currentPage.getType() == NodeType.***INDEX***
  - getType() is a method defined under HFPage Class which return the type of page – leaf or Index
- nextPageId=currentIndexPage.getPageNoByKey(key)
  - getPageNoByKey is a method defined under IndexPage which will return the pageId or pageno of the page holding the key that is passed

An overview of given code contd..

- delEntry(keyDataEntry(key,rid))
  - delete a data entry in the leaf page. Defined btree.BTLeafPage
- available_space()
  - returns the amount of available space on the page. Defined under heap.HFPage. [hint: use it with gobal constants MAX_SPACE and HFpage.DPFIXED to get the correct size of a page]
  - For each LeafPage and IndexPage add 4 as slot size to the above calculation. What will be slot size during merging?
- updateHeader(PageId)
  - Useful to make the root to point to a particular page; IndexPage or LeafPage or InvalidPage. Defined under btree.BTreeFile.updateHeader

An overview of given code contd..

- parentPage.getSibling(key, siblingPageId);
  - getSibling is useful in merging to get the direction to modify an IndexPage. return 0 if no sibling; -1 if left sibling; 1 if right sibling. For index page use it carefully. Defined under btree.page.BTIndexPage
- redistribute(BTIndexPage indexPage, BTIndexPage parentIndexPage, int direction, Key deletedKey)
  - Used to do redistribution. Retuen a bolean value. Defined under btree.page.BTIndexPage
- getKeyLength(KeyClass key)
  - Useful to merge IndexPage as it contains key and pointers. return the length of the key. Defined under btree.BT

An overview of given code contd..

- Since it is a disk based implementation. There is a need to keep changes. HOW?
  - pinPage(PageId pin_pgid, Page page, boolean emptyPage)
    - Check if this page is in buffer pool, otherwise find a frame for this page, read in and pin it. Can be found under bufmgr.BufMgr
  - unpinPage(PageId pgid, boolean dirty)
    - To unpin a page specified by a pageId. Can be found under bufmgr.BufMgr
  - freePage(PageId globalPageId)
    - User should call this method if she needs to delete a page. Can be found under bufmgr.BufMgr

Partial algorithm for _Delete()

- Deletion can occur in two possible ways
  - On a leaf page
  - Or on a index page

- On a leaf page;
  - Different cases after deletion are
    - At least half full after deletion
    - Tree only has root node
    - Whole tree is empty
  - Cannot redistribute nor merge
    - Rare case!!
- On a index page; Similar case as leaf Page just logic to perform merging changes [Hint: Use recursion]

IMPORTANT INFORMATION

- All the team with even number need to check for 40% occupancy to perform a merge or redistribution
- And all the team with odd number need to check for 70% occupancy to perform a merge or redistribution
- Don’t Panic. You will get your team number soon !!
- If you don’t get a team number send you team information so that we can assign one.
- HAPPY CODING !!

Thank You !!!