THE UNIVERSITY OF TEXAS AT ARLINGTON

COMPUTER SCIENCE

DBMS MODELS AND IMPLEMENTAION

A PROJECT REPORT ON

B+ TREE (FULL DELETION)

Team : 6

ISHWOR TIMILSINA (1001256415)

GAURAV THAPA (1001154764)

PROF. SHARMA CHAKRAVARTHY

DATE : 10/01/2015

- **Overall Status**

  We have written all the required code in the single allocated method i.e. _Delete. We have not created or implemented any self-made methods. We have created a global constant MIN_OCCUPANCY and set it to 0.5 ; to multiply to the space calculated later. We have used following methods pre-provided to us: -

  - currentPage.getType() == NodeType.LEAF/INDEX
  - pinPage(currentPageId)/unpinPage(currentPageId)
  - BTIndexPage, BTSortedPage, BTLeafPage
  - RID, PageId, KeyDataEntry, KeyClass
  - _Delete
  - getCurPage()
  - parentPage.getSibling(key, siblingPageId)
  - currentPage.available_space()
  - MAX_SPACE, HFPage.DPFIXED, HFPage.SIZE_OF_SLOT
  - siblingPage.redistribute(currentPage, parentPage, direction, key)
  - getNextpage(), setNextPage()
  - getFirst(rid), getNext(rid)
  - insertKey(key, pageId), deleteKey(key)
  - updateHeader(currentPageId)
  - currentPage.delEntry(KeyDataEntry)

  _Delete Code Flow

  1. Create global constant MIN_OCCUPANCY and set to 0.5
  2. Find if the current page is index or leaf
  3. If Index, do following
     i. Initiate the current index page
     ii. Check if the index page has more than the allowed minimum space
     iii. If yes, create sibling page
     iv. Try redistributing
     v. If failed, try merging as follows
        a. Take the first entry of the current page
        b. Add the entry in the sibling page
        c. Delete the entry from the current page
        d. Repeat until no entry left in the current page
        e. To maintain the linked list, do the following
        - If the sibling is on right side of the current page, set the previous page of current page as the left page of the sibling page and vice versa. If there is no previous page, just set the id -1 as the previous page of the sibling page
        - If the sibling is on the left side of the current page, set the next page of current page as the right page of the sibling page and vice versa. If there is no next page, just set the id -1 as the next page of the sibling page

    f. Get the last entry of the sibling page, tally it with the keys in the parent page to find out the key pointing to the current page. Pull it down to the sibling page. Delete the key from the parent page.

  vi. If the parent page as no entry, set the root pointer to the current sibling page.

4. If the current page in LEAF, do the following
   i. Initiate the current leaf page
   ii. Check if the leaf page has more than the allowed minimum space
   iii. If yes, Create sibling page
   iv. Try redistributing
   v. If failed, try merging as follows
      a. Take the first entry of the current page
      b. Add the entry in the sibling page
      c. Delete the entry from the current page
      d. Repeat until no entry left in the current page
      e. To maintain the linked list, do the following
      - If the sibling is on right side of the current page, set the previous page of current page as the left page of the sibling page and vice versa. If there is no previous page, just set the id -1 as the previous page of the sibling page
      - If the sibling is on the left side of the current page, set the next page of current page as the right page of the sibling page and vice versa. If there is no next page, just set the id -1 as the next page of the sibling page
      f. Get the last entry of the sibling page, tally it with the keys in the parent page to find out the key pointing to the current page. Delete the key from the parent page.
   vi. If the parent page as no entry, set the root pointer to the current sibling page.

• Division of Labor

Since we did this project in a team, we divided our labor. The overall estimated time it took us to do this project was 90 hours.

1. Ishwor Timilsina (48 hours)
   i. Algorithm to pseudocode transformation
   ii. Minibase study
   iii. Previous implemented methods study
   iv. Coding
   v. Testing

2. Gaurav Thapa (42 hours)
   i. Documentation
   ii. Testing
   iii. Coding
   iv. B+ tree deletion examples research

- **Logical errors encountered and handled**

  During the coding and implementation, we encountered a lot of errors that we had to handle and correct.
  One of the errors we faced was during the return of the key (recursion) in both index and leaf conditions. We solved that by assigning a key to delete the pointer key on the parent, deleted the key and then return the same key for the recursion.
  Another error we faced was during the checking of available space. We had first subtracted the SIZE_OF_SLOT from MAX_SPACE and DPFIXED. It was not working sometimes and sometimes showing the errors. We later solved it by adding it instead of subtracting.
  We also faced difficulties while finding out which key to delete from the parent. We were first tallying all the keys on the parent with the first entry of the current page that was merged into the sibling page. We later solved it by tallying all the keys on the parent with the last entry of the sibling page after merging.
  There was also an initial nullpointerexception at the parent page initiation. We solved it by applying an if …else condition. We carried the operation only if the parentPageId was not null, else we returned null.
  We encountered several other errors, one of which was – the error due to the linking of invalid page while maintaining the linked list. We solved it by applying as if …else condition. We set the link to the sibling page no matter what but didn't set the reverse link if there was an invalid page linked.