Writeup for LAB2

BufferPool.Java

Flushpage()

flushPage() is a private method in the BufferPool class that is responsible for flushing a specific page to disk. The method takes a PageId as a parameter, which indicates the ID of the page that needs to be flushed to disk.First, it checks if the pid2pg map (which maps PageId to Page) contains the specified PageId. If not, it means the page is not in the buffer pool and doesn't need to be flushed, so the method returns without performing any further actions.If the page is in the buffer pool, it retrieves the corresponding Page object from the pid2pg map.If the page is dirty, it retrieves the table ID from the page's PageId. It obtains the corresponding DbFile object from the database catalog using the table ID.The writePage() method of the DbFile is called, passing the dirty page as an argument. This method writes the page's contents to disk, effectively flushing it.After the page has been successfully written to disk, the markDirty() method of the page is called, passing false as the parameter to mark the page as not dirty anymore. This is done to ensure that future modifications to the page will be correctly tracked.

BTreeFile.Java

findLeafPage()

The findLeafPage() function is used to locate a leaf page in a B+-Tree based on a given search key. It starts from the root page and recursively follows the appropriate child pointers until it reaches a leaf page. If the current page is an internal page, it compares the search key with the keys in the page to determine the next child page to explore. Eventually, it returns the leaf page that contains the desired key or the closest key in the case of a range search.

splitLeafPage()

Create a new empty leaf page called rightPage.Determine the number of tuples to move from the original page to rightPage.Move the specified number of tuples from page to rightPage.Update the sibling pointers of the leaf pages affected by the split.Retrieve the middle key from the remaining tuple in page.Find the appropriate parent page for the new entry.Update the parent ID of page and rightPage to the parent page's ID.Create a new entry with the middle key, page's ID, and rightPage's ID, and insert it into the parent page.Mark the modified parent page, page, and rightPage as dirty.Based on a comparison with the given key field, return either page or rightPage for inserting a new tuple.

splitInternalPage()

Create a new empty internal page called rightPage.Determine the number of entries to move from the original page to rightPage. It's roughly half the number of entries in page, excluding the middle entry.Move the specified number of entries from page to rightPage.Update the middle entry by setting its left child to page and its right child to rightPage.Find the appropriate parent page for the new entry.Insert the middle entry into the parent page.Update the parent IDs of both

page and rightPage to the ID of the parent page.Update the parent pointers of the child pages in rightPage to point to rightPage.Mark the modified parent page, page, and rightPage as dirty.Based on a comparison with the given key field, return either page or rightPage to indicate which page the search should continue.

stealFromLeafPage()

Determine the number of tuples to move from the sibling page to the target page. It's roughly half the difference between the number of tuples in the sibling page and the target page.Iterate over the tuples in the sibling page, either from the beginning or the end, depending on whether it's the right or left sibling.Move the specified number of tuples from the sibling page to the target page.Update the key of the corresponding entry in the parent page to reflect the new middle key. This is done by obtaining the appropriate tuple from either the target or sibling page, depending on which one is on the left side.

mergeLeafPages()

Move all the tuples from the right page to the left page.Update the sibling pointers of the leaf pages to reflect the merge. The right sibling of the left page is set to the right sibling of the right page. If there is a right sibling, update its left sibling pointer to the left page.Mark the left page as dirty in the dirtypages map.Make the right page available for reuse by setting it as an empty page.Delete the entry in the parent page that corresponds to the two pages being merged. This is done using the deleteParentEntry() function.

mergeInternalPages()

Move all the entries from the right page to the left page.Update the parent pointers of the child pages in the moved entries to point to the correct pages.Make the right page available for reuse by setting it as an empty page.Delete the entry in the parent page that corresponds to the two pages being merged. This is done using the deleteParentEntry() function.

How long spent on LAB

About 20 hours. It’s harder than the last project. The coding burden is even heavier. I had to read the code given for a long time to decide what to do next.