WriteUp for Lab2

521120910234 Shiyi Huang

Design Decision

Exercise1

To implemente the BufferPool, I select **Least Recently Used (LRU)** eviction strategy, which is a popular method for managing a finite number of cache pages. In this design, each page accessed, whether for reading or writing, is moved to the front of a doubly linked list to maintain recency. When the cache reaches its capacity and a new page needs to be inserted, the BufferPool evicts the least recently used page—identified as the one closest to the tail of the list that is not currently marked as dirty. This approach ensures that the most recently used pages remain in the cache longer, which can significantly improve the performance of database operations by reducing the number of disk reads and writes.

Exercise2

Function `findLeafPage` is designed to efficiently search a B+ tree for a specific key field, `f`, using a recursive approach. It navigates through the tree from the root to the appropriate leaf node, locking internal nodes in read-only mode and the target leaf node with specified permissions to ensure data integrity during concurrent accesses. If `f` is null, the function targets the left-most leaf, aiding in the implementation of iterators. This method effectively harnesses the structured navigation properties of B+ trees to quickly locate and access the necessary data, optimizing both range queries and sequential data traversal.

Exercise3

The `splitLeafPage` and `splitInternalPage` functions are designed to manage space in a B+ tree when inserting new elements causes overflow. `splitLeafPage` handles leaf overflows by splitting the full leaf into two, redistributing tuples, and updating sibling and parent pointers; it ensures the new tuple is inserted into the correct leaf by comparing the insertion key with the median key of the split. `splitInternalPage` addresses overflow in internal nodes similarly by redistributing entries, pushing the median key up to the parent, and recursively splitting higher levels if necessary. Both functions ensure the tree maintains its balanced structure and update necessary pointers to reflect changes, making them essential for maintaining the efficiency and integrity of B+ tree operations during insertions.

Exercise4

The functions `stealFromLeafPage` , `stealFromLeftInternalPage` , and `stealFromRightInternalPage` are designed to manage and maintain the balance of B+ tree structures by redistributing data between sibling nodes when one becomes less than half full. `stealFromLeafPage` shifts tuples between leaf pages and updates the parent's keys to ensure both siblings maintain minimum occupancy. Similarly, `stealFromLeftInternalPage` and `stealFromRightInternalPage` adjust the distribution of entries between internal pages, pulling down or pushing up keys to the parent to maintain a balanced tree structure, ensuring that internal nodes also meet occupancy requirements and maintain proper tree integrity. These mechanisms are crucial for the B+ tree's performance, especially under high update and delete operations, by preventing excessive page splits or merges.

Exercise5

The functions `mergeLeafPages` and `mergeInternalPages` are designed to consolidate data within a B+ tree by merging two neighboring nodes (leaf or internal) into one when one of them falls below the required minimum occupancy after deletions. The `mergeLeafPages` function transfers all tuples from a right leaf to a left leaf, updating sibling links and adjusting the parent's entries, while `mergeInternalPages` moves all entries from a right internal page to a left internal page, incorporating a key from the parent to maintain the tree structure, both methods also marking the emptied right page for reuse.

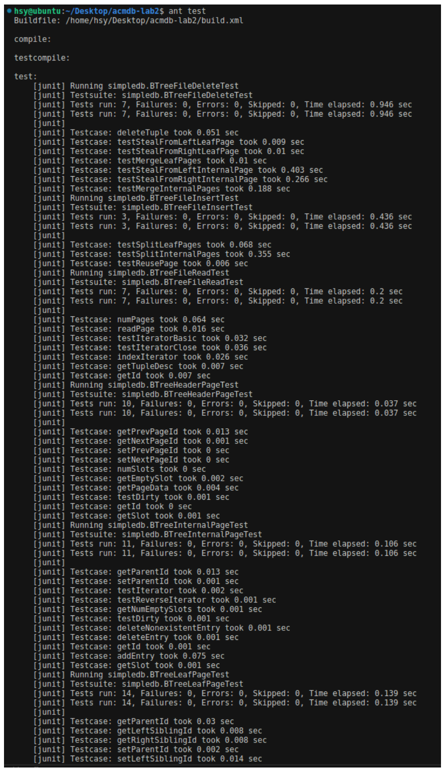
These functions, along with `stealFromLeafPage` , `stealFromLeftInternalPage` , and `stealFromRightInternalPage` , collectively support the deletion operations in a B+ tree by managing the redistribution and consolidation of nodes to maintain balance and occupancy rules. During deletion, if a node falls below half occupancy, the tree first tries to redistribute entries or tuples from adjacent siblings; if this is not sufficient to achieve balance, it then merges nodes. These mechanisms ensure the B+ tree remains balanced and efficient, preserving its optimal path lengths and search times even after multiple deletions.

Change Made to API

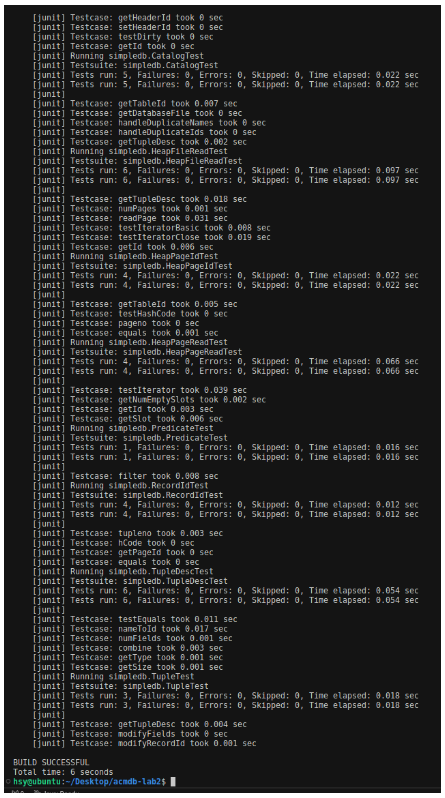
The main changes to the API involve the additional definition of three classes: `ResourceLock` for locking, `LockController` for lock control, and `DoublyLinkedNode` for doubly linked list support, to facilitate the LRU strategy.

Conclusion

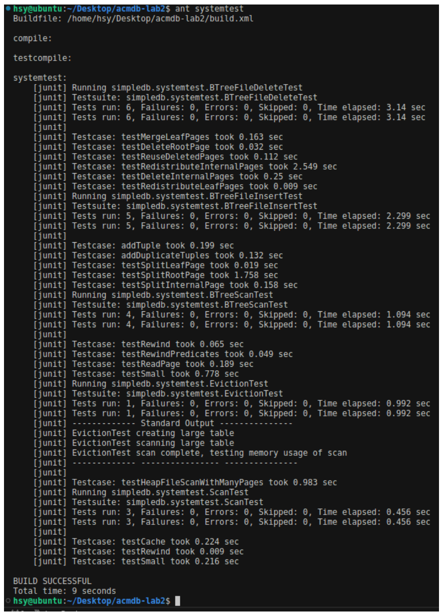
I have completed all the requirements for this experiment, which took me approximately 15 hours. The difficulty mainly lied in understanding B+ trees and implementing their search/delete algorithms into Java code, which posed quite a challenge. Finally, providing screenshots of passing all the test cases was the last step.



```
hsy@ubuntu:~/Desktop/acmdb-lab2$ ant test
Buildfile: /home/hsy/Desktop/acmdb-lab2/build.xml

compile:

testcompile:

test:
    [junit] Running simpledb.BTreeFileDeleteTest
    [junit] Testsuite: simpledb.BTreeFileDeleteTest
    [junit] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.946 sec
    [junit] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.946 sec
    [junit]
    [junit] Testcase: deleteTuple took 0.051 sec
    [junit] Testcase: testStealFromLeftLeafPage took 0.009 sec
    [junit] Testcase: testStealFromRightLeafPage took 0.01 sec
    [junit] Testcase: testMergeLeafPages took 0.01 sec
    [junit] Testcase: testStealFromLeftInternalPage took 0.403 sec
    [junit] Testcase: testStealFromRightInternalPage took 0.266 sec
    [junit] Testcase: testMergeInternalPages took 0.188 sec
    [junit] Running simpledb.BTreeFileInsertTest
    [junit] Testsuite: simpledb.BTreeFileInsertTest
    [junit] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.436 sec
    [junit] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.436 sec
    [junit]
    [junit] Testcase: testSplitLeafPages took 0.068 sec
    [junit] Testcase: testSplitInternalPages took 0.355 sec
    [junit] Testcase: testReusePage took 0.006 sec
    [junit] Running simpledb.BTreeFileReadTest
    [junit] Testsuite: simpledb.BTreeFileReadTest
    [junit] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.2 sec
    [junit] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.2 sec
    [junit]
    [junit] Testcase: numPages took 0.064 sec
    [junit] Testcase: readPage took 0.016 sec
    [junit] Testcase: testIteratorBasic took 0.032 sec
    [junit] Testcase: testIteratorClose took 0.036 sec
    [junit] Testcase: indexIterator took 0.026 sec
    [junit] Testcase: getTupleDesc took 0.007 sec
    [junit] Testcase: getId took 0.007 sec
    [junit] Running simpledb.BTreeHeaderPageTest
    [junit] Testsuite: simpledb.BTreeHeaderPageTest
    [junit] Tests run: 10, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.037 sec
    [junit] Tests run: 10, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.037 sec
    [junit]
    [junit] Testcase: getPrevPageId took 0.013 sec
    [junit] Testcase: getNextPageId took 0.001 sec
    [junit] Testcase: setPrevPageId took 0 sec
    [junit] Testcase: setNextPageId took 0 sec
    [junit] Testcase: numSlots took 0 sec
    [junit] Testcase: getEmptySlot took 0.002 sec
    [junit] Testcase: getPageData took 0.004 sec
    [junit] Testcase: testDirty took 0.001 sec
    [junit] Testcase: getId took 0 sec
    [junit] Testcase: getSlot took 0.001 sec
    [junit] Running simpledb.BTreeInternalPageTest
    [junit] Testsuite: simpledb.BTreeInternalPageTest
    [junit] Tests run: 11, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.106 sec
    [junit] Tests run: 11, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.106 sec
    [junit]
    [junit] Testcase: getParentId took 0.013 sec
    [junit] Testcase: setParentId took 0.001 sec
    [junit] Testcase: testIterator took 0.002 sec
    [junit] Testcase: testReverseIterator took 0.001 sec
    [junit] Testcase: getNumEmptySlots took 0.001 sec
    [junit] Testcase: testDirty took 0.001 sec
    [junit] Testcase: deleteNonexistentEntry took 0.001 sec
    [junit] Testcase: deleteEntry took 0.001 sec
    [junit] Testcase: getId took 0.001 sec
    [junit] Testcase: addEntry took 0.075 sec
    [junit] Testcase: getSlot took 0.001 sec
    [junit] Running simpledb.BTreeLeafPageTest
    [junit] Testsuite: simpledb.BTreeLeafPageTest
    [junit] Tests run: 14, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.139 sec
    [junit] Tests run: 14, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.139 sec
    [junit]
    [junit] Testcase: getParentId took 0.03 sec
    [junit] Testcase: getLeftSiblingId took 0.008 sec
    [junit] Testcase: getRightSiblingId took 0.008 sec
    [junit] Testcase: setParentId took 0.002 sec
    [junit] Testcase: setLeftSiblingId took 0.014 sec
```



```
    [junit] Testcase: getHeaderId took 0 sec
    [junit] Testcase: setHeaderId took 0 sec
    [junit] Testcase: testDirty took 0 sec
    [junit] Testcase: getId took 0 sec
    [junit] Running simpledb.CatalogTest
    [junit] Testsuite: simpledb.CatalogTest
    [junit] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.022 sec
    [junit] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.022 sec
    [junit]
    [junit] Testcase: getTableId took 0.007 sec
    [junit] Testcase: getDatabaseFile took 0 sec
    [junit] Testcase: handleDuplicateNames took 0 sec
    [junit] Testcase: handleDuplicateIds took 0 sec
    [junit] Testcase: getTupleDesc took 0.002 sec
    [junit] Running simpledb.HeapFileReadTest
    [junit] Testsuite: simpledb.HeapFileReadTest
    [junit] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.097 sec
    [junit] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.097 sec
    [junit]
    [junit] Testcase: getTupleDesc took 0.018 sec
    [junit] Testcase: numPages took 0.001 sec
    [junit] Testcase: readPage took 0.031 sec
    [junit] Testcase: testIteratorBasic took 0.008 sec
    [junit] Testcase: testIteratorClose took 0.019 sec
    [junit] Testcase: getId took 0.006 sec
    [junit] Running simpledb.HeapPageIdTest
    [junit] Testsuite: simpledb.HeapPageIdTest
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.022 sec
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.022 sec
    [junit]
    [junit] Testcase: getTableId took 0.005 sec
    [junit] Testcase: testHashCode took 0 sec
    [junit] Testcase: pageno took 0 sec
    [junit] Testcase: equals took 0.001 sec
    [junit] Running simpledb.HeapPageReadTest
    [junit] Testsuite: simpledb.HeapPageReadTest
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.066 sec
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.066 sec
    [junit]
    [junit] Testcase: testIterator took 0.039 sec
    [junit] Testcase: getNumEmptySlots took 0.002 sec
    [junit] Testcase: getId took 0.003 sec
    [junit] Testcase: getSlot took 0.006 sec
    [junit] Running simpledb.PredicateTest
    [junit] Testsuite: simpledb.PredicateTest
    [junit] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.016 sec
    [junit] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.016 sec
    [junit]
    [junit] Testcase: filter took 0.008 sec
    [junit] Running simpledb.RecordIdTest
    [junit] Testsuite: simpledb.RecordIdTest
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.012 sec
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.012 sec
    [junit]
    [junit] Testcase: tupleno took 0.003 sec
    [junit] Testcase: hCode took 0 sec
    [junit] Testcase: getPageId took 0 sec
    [junit] Testcase: equals took 0 sec
    [junit] Running simpledb.TupleDescTest
    [junit] Testsuite: simpledb.TupleDescTest
    [junit] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.054 sec
    [junit] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.054 sec
    [junit]
    [junit] Testcase: testEquals took 0.011 sec
    [junit] Testcase: nameToId took 0.017 sec
    [junit] Testcase: numFields took 0.001 sec
    [junit] Testcase: combine took 0.003 sec
    [junit] Testcase: getType took 0.001 sec
    [junit] Testcase: getSize took 0.001 sec
    [junit] Running simpledb.TupleTest
    [junit] Testsuite: simpledb.TupleTest
    [junit] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.018 sec
    [junit] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.018 sec
    [junit]
    [junit] Testcase: getTupleDesc took 0.004 sec
    [junit] Testcase: modifyFields took 0 sec
    [junit] Testcase: modifyRecordId took 0.001 sec

BUILD SUCCESSFUL
Total time: 6 seconds
hsy@ubuntu:~/Desktop/acmdb-lab2$
```



```
hsy@ubuntu:~/Desktop/acmdb-lab2$ ant systemtest
Buildfile: /home/hsy/Desktop/acmdb-lab2/build.xml

compile:

testcompile:

systemtest:
    [junit] Running simpledb.systemtest.BTreeFileDeleteTest
    [junit] Testsuite: simpledb.systemtest.BTreeFileDeleteTest
    [junit] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 3.14 sec
    [junit] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 3.14 sec
    [junit]
    [junit] Testcase: testMergeLeafPages took 0.163 sec
    [junit] Testcase: testDeleteRootPage took 0.032 sec
    [junit] Testcase: testReuseDeletedPages took 0.112 sec
    [junit] Testcase: testRedistributeInternalPages took 2.549 sec
    [junit] Testcase: testDeleteInternalPages took 0.25 sec
    [junit] Testcase: testRedistributeLeafPages took 0.009 sec
    [junit] Running simpledb.systemtest.BTreeFileInsertTest
    [junit] Testsuite: simpledb.systemtest.BTreeFileInsertTest
    [junit] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.299 sec
    [junit] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.299 sec
    [junit]
    [junit] Testcase: addTuple took 0.199 sec
    [junit] Testcase: addDuplicateTuples took 0.132 sec
    [junit] Testcase: testSplitLeafPage took 0.019 sec
    [junit] Testcase: testSplitRootPage took 1.758 sec
    [junit] Testcase: testSplitInternalPage took 0.158 sec
    [junit] Running simpledb.systemtest.BTreeScanTest
    [junit] Testsuite: simpledb.systemtest.BTreeScanTest
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.094 sec
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.094 sec
    [junit]
    [junit] Testcase: testRewind took 0.065 sec
    [junit] Testcase: testRewindPredicates took 0.049 sec
    [junit] Testcase: testReadPage took 0.189 sec
    [junit] Testcase: testSmall took 0.778 sec
    [junit] Running simpledb.systemtest.EvictionTest
    [junit] Testsuite: simpledb.systemtest.EvictionTest
    [junit] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.992 sec
    [junit] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.992 sec
    [junit] ------------- Standard Output ---------------
    [junit] EvictionTest creating large table
    [junit] EvictionTest scanning large table
    [junit] EvictionTest scan complete, testing memory usage of scan
    [junit] ------------- ---------------- ---------------
    [junit]
    [junit] Testcase: testHeapFileScanWithManyPages took 0.983 sec
    [junit] Running simpledb.systemtest.ScanTest
    [junit] Testsuite: simpledb.systemtest.ScanTest
    [junit] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.456 sec
    [junit] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.456 sec
    [junit]
    [junit] Testcase: testCache took 0.224 sec
    [junit] Testcase: testRewind took 0.009 sec
    [junit] Testcase: testSmall took 0.216 sec

BUILD SUCCESSFUL
Total time: 9 seconds
hsy@ubuntu:~/Desktop/acmdb-lab2$
```