50.043 Database Lab 4 Report

Name	Student ID
Matthaeus Choo Zhi Jie (Zhu Zhijie)	1008103
Lee Jing Kang	1008240
Chong Juo Ru Faustina	1007967

50.043 Database System and Big Data ~ Lab 4

Summary and Design decisions

Exercise 1 ~ BTreeFile.findLeafPage()

In this implementation, recursive descent is performed via BTreeInternalPage.iterator(). The search process selects the left child corresponding to the first key greater than or equal to f, thereby ensuring a left-bias in the presence of duplicate keys. If f is null, the traversal consistently follows the leftmost path, which facilitates full table scans. Internal pages are always accessed with READ_ONLY permissions, while access to the target leaf node adheres to the caller's specified permissions. All page retrieval operations are conducted through getPage(...), enabling accurate tracking of dirty pages. Additionally, the system incorporates fail-fast mechanisms: if an internal page is empty or malformed, a DbException is raised to prevent undetected misrouting.

Exercise 2 ~ BTreeFile.splitLeafPage() and BTreeFile.splitInternalPage()

During the split operation, the process moves to the right side using reverse iterators, effectively transferring half of the tuples or entries. The separator key is then propagated upward in the structure. Specifically, for leaf nodes, the leftmost key from the newly created right page is copied up, while for internal nodes, the true middle entry is removed and its key is inserted into the parent node. Structural consistency is maintained by updating sibling links among leaves and ensuring all parent pointers are correctly set. Page allocation is managed exclusively through getEmptyPage(...), and any modified page is cached in dirtypages to ensure data integrity.

Most important part of this process involves selecting the appropriate separator and determining the correct insertion target. The insertion target is chosen based on the splitKey field, which preserves left-bias for duplicates and ensures correctness, particularly during cascading splits. This approach is for maintaining both data structure consistency and the integrity of duplicate handling.

Exercise 3 ~ BTreeFile.stealFromLeafPage(), BTreeFile.stealFromLeftInternalPage() and BTreeFile.stealFromRightInternalPage()

During implementation, we determine the precise number of entries to transfer in order to balance both pages at approximately half occupancy. Depending on whether the redistribution involved the left or right sibling, we employed either forward or reverse iterators to facilitate this process. Key rotation through the parent entry was necessary to maintain consistent separator values, with the parent's key field updated following each transfer.

Exercise 4 ~ BTreeFile.mergeLeafPages() and BTreeFile.mergeInternalPages()

When performing a left merge, all tuples from the right page are transferred to the left page. For leaf nodes, this process involves moving the relevant entries, updating the left and right sibling links to maintain structural integrity, marking the right page as reusable via setEmptyPage, and removing the separator key from the parent using deleteParentEntry.

For internal nodes, the procedure is slightly more involved. The parent separator key is moved down into the left page. Subsequently, all entries from the right page are streamed into the left page in the order. Parent pointers for all affected children must then be updated to reflect the new structure. After these steps, the parent entry is removed, and the right page is freed.

It is important to carefully handle separator keys to preserve key ordering and to use precise delete operations, such as deleteKeyAndLeftChild or deleteParentEntry, to prevent recordId drift. Furthermore, any cascading underflow conditions must be resolved through the parent-deletion process.

Missing/Incomplete Elements Of Our Code/Changes Made To The API

All Lab 1, 2, 3 and 4 code have been implemented. No Changes made to the API.

Dear Profs,

For our last testcase, we can't consistently pass the last system case, we only pass it about 75% of the time. May I know what is wrong?