Alexander Peseckis

Win San

Steven Hizmi

Programming Project 3: B+ Tree Index Manager

For this project, we implemented a B+ tree index on a search key that is of type integer. A B+ tree index is widely used in many database management systems because it supports very efficient equality and range search. For this assignment, the index stores data entries in the form of key and record id pairs.

With regards to our implementation, we build the index by repeatedly calling the insertEntry method for each record in the relation when the index is constructed. If the tree only contains the root node, we insert the data entry into the root. If the tree contains more than just the root node, we perform a binary search to find the appropriate leaf node and insert the record accordingly. If the leaf node is full, we split it into two nodes, redistribute the keys evenly, and copy the middle key up to the parent node. If the parent node isn't full, then we just copy the key into the node. If the parent node is full, we redistribute the keys evenly, and push the middle key up the tree. This can propagate recursively to other nodes. The maximum number of pages that are pinned is equal to the height of the tree. All pages eventually get unpinned by the time the insertEntry method returns. Selecting a range of records is done by calling startScan. It works by performing binary search to first find the lowest-value search key. During the process, each page is unpinned after it is read and used to determine the next node to traverse down to. We then utilize the right sibling pointers of each leaf node to avoid unnecessarily traversing the tree up and down multiple times during the range search. During the process, we again unpin pages as soon as we read them and use them to determine the next leaf to traverse to.

Tests:

- test4: Tests whether the bTree can handle negative values.
- test5: Tests whether the bTree can handle an empty Tree.
- test6: Tests whether the bTree can handle index out of bounds.