Eric Yang

304263623

MultiMap::findEqual(std::string key) const

Create two iterators (one to track the current node and one to track the prev node)

While the current iterator is valid

Check if the previous node is valid

If it is, set the iterator to that node and return the iterator to the current node

If not use prev() on both iterators.

If the current node's key is equal to the wanted key,

If the iterator is valid and both iterators are marking the same node,

prev() the current iterator until it points at the first value in the node.

else if the current node's key is less than the target key,

move on to the next node

else if the current node's key is greater than the target key,

move to the previous node

If no valid iterators have been returned, return an invalid iterator

MultiMap::Iterator::next()

if the current iterator is invalid, the function returns false without doing anything.

If the current node has more than one value associated with it, we move to the next value in that node's linked list.

else if the node has a right child, the function will use an inorder traversal to find the next node in that subtree

else the function will use the inorder traversal to find the highest ancestor.

If the whole tree has been traversed, next() returns false.

else next() sets the iterator to mark the position and first value of that node.

This function satisfies the condition O(log N) on average and O(N) worst case, because the function will not visit every node when moving through the tree to find the next node unless they are on opposite ends

Database::search(const std::vector<SearchCriterion>& searchCriteria, const std::vector<SortCriterion>& sortCriteria, std::vector<int>& results)

if searchCriteria isn't a valid parameter, clear the results vector and return ERROR\_RESULT

Go through each searchCriterion

if the min and max are both empty, clear the results and return ERROR\_RESULT

Check if the criterion's field names match the any schema's names

If names match, check if the field is indexed

If it is, the index will now mark the MultiMap in m\_fieldIndex that holds the searchCriteria

If there are no indexed fields, clear results and return ERROR\_RESULT

Create a new vector to hold the results that match, and two iterators to traverse.

Locate which m\_rows match searchCriterion

If there is a maxValue but no minValue,

Call findEqualOrPredecessor() for the maxValue on the proper Multimap and push back values, calling prev() on the max iterator each iteration.

If there is a minValue but no maxValue,

Call findEqualOrSuccessor() for the minValue on the proper Multimap and push back values, calling next() on the min iterator each iteration.

If both minValue and maxValue exist,

Call findEqualOrSuccessor() for the minValue on the proper Multimap and call findEqualOrPredecessor() for the maxValue on the proper Multimap.

If the results vector is empty, set results vector and matches vector equal to each other

Otherwise only add the vectors that overlap in both the current matches and the previous matches (results vector)

use an array to hold the values in the vector

use a recursive sorting function to sort the rows in order while passing the sorting criteria

place the sorted array into the results vector

My sorting function doesn't work properly. It's somewhat sorts the results but it leaves some random rows in the middle. I used the same sorting functions from Homework 3. I can't figure out the time complexity because it is incomplete.