package com.twitter.search.earlybird.index;

import com.twitter.search.core.earlybird.index.TimeMapper;

import com.twitter.search.core.earlybird.index.inverted.IntBlockPool;

import com.twitter.search.core.earlybird.index.util.SearchSortUtils;

import com.twitter.search.earlybird.search.queries.SinceUntilFilter;

public abstract class AbstractInMemoryTimeMapper implements TimeMapper {

// Reverse map: timestamp to first doc ID seen with that timestamp.

// This is two arrays: the timestamps (sorted), and the doc ids.

protected final IntBlockPool reverseMapTimes;

protected final IntBlockPool reverseMapIds;

protected volatile int reverseMapLastIndex;

public AbstractInMemoryTimeMapper() {

this.reverseMapTimes = new IntBlockPool(ILLEGAL\_TIME, "time\_mapper\_times");

this.reverseMapIds = new IntBlockPool(ILLEGAL\_TIME, "time\_mapper\_ids");

this.reverseMapLastIndex = -1;

}

protected AbstractInMemoryTimeMapper(int reverseMapLastIndex,

IntBlockPool reverseMapTimes,

IntBlockPool reverseMapIds) {

this.reverseMapTimes = reverseMapTimes;

this.reverseMapIds = reverseMapIds;

this.reverseMapLastIndex = reverseMapLastIndex;

}

@Override

public final int getLastTime() {

return reverseMapLastIndex == -1 ? ILLEGAL\_TIME : reverseMapTimes.get(reverseMapLastIndex);

}

@Override

public final int getFirstTime() {

return reverseMapLastIndex == -1 ? ILLEGAL\_TIME : reverseMapTimes.get(0);

}

@Override

public final int findFirstDocId(int timeSeconds, int smallestDocID) {

if (timeSeconds == SinceUntilFilter.NO\_FILTER || reverseMapLastIndex == -1) {

return smallestDocID;

}

final int index = SearchSortUtils.binarySearch(

new IntArrayComparator(), 0, reverseMapLastIndex, timeSeconds, false);

if (index == reverseMapLastIndex && reverseMapTimes.get(index) < timeSeconds) {

// Special case for out of bounds time.

return smallestDocID;

}

return reverseMapIds.get(index);

}

protected abstract void setTime(int docID, int timeSeconds);

protected void doAddMapping(int docID, int timeSeconds) {

setTime(docID, timeSeconds);

int lastTime = getLastTime();

if (timeSeconds > lastTime) {

// Found a timestamp newer than any timestamp we've seen before.

// Add a reverse mapping to this tweet (the first seen with this timestamp).

//

// When indexing out of order tweets, we could have gaps in the timestamps recorded in

// reverseMapTimes. For example, if we get 3 tweets with timestamp T0, T0 + 5, T0 + 3, then we

// will only record T0 and T0 + 5 in reverseMapTimes. However, this should not be an issue,

// because reverseMapTimes is only used by findFirstDocId(), and it's OK for that method to

// return a smaller doc ID than strictly necessary (in this case, findFirstDocId(T0 + 3) will

// return the doc ID of the second tweet, instead of returning the doc ID of the third tweet).

reverseMapTimes.add(timeSeconds);

reverseMapIds.add(docID);

reverseMapLastIndex++;

}

}

private class IntArrayComparator implements SearchSortUtils.Comparator<Integer> {

@Override

public int compare(int index, Integer value) {

return Integer.compare(reverseMapTimes.get(index), value);

}

}

}