package com.twitter.search.core.earlybird.index.inverted;

import java.io.IOException;

import javax.annotation.Nullable;

import org.apache.lucene.util.packed.PackedInts;

/\*\*

\* A PostingsEnum for iterating over LowDFPackedIntsPostingLists.

\*

\* Can be used with positions and without positions.

\*/

public class LowDFPackedIntsPostingsEnum extends EarlybirdOptimizedPostingsEnum {

private static final int SKIP\_INTERVAL = 128;

private final PackedInts.Reader packedDocIds;

@Nullable

private final PackedInts.Reader packedPositions;

private final int lastPostingPointer;

private final int largestDocID;

private int currentPositionPointer;

/\*\* Pointer to the next posting that will be loaded. \*/

private int nextPostingPointer;

/\*\*

\* Creates a new PostingsEnum for all postings in a given term.

\*/

public LowDFPackedIntsPostingsEnum(

PackedInts.Reader packedDocIds,

@Nullable

PackedInts.Reader packedPositions,

int postingListPointer,

int numPostings) {

super(postingListPointer, numPostings);

this.packedDocIds = packedDocIds;

this.packedPositions = packedPositions;

this.nextPostingPointer = postingListPointer;

this.lastPostingPointer = postingListPointer + numPostings - 1;

this.largestDocID = (int) packedDocIds.get(lastPostingPointer);

loadNextPosting();

// Treat each term as a single block load.

queryCostTracker.track(QueryCostTracker.CostType.LOAD\_OPTIMIZED\_POSTING\_BLOCK);

}

@Override

protected void loadNextPosting() {

if (nextPostingPointer <= lastPostingPointer) {

nextDocID = (int) packedDocIds.get(nextPostingPointer);

nextFreq = 1;

} else {

// all postings fully processed

nextDocID = NO\_MORE\_DOCS;

nextFreq = 0;

}

nextPostingPointer++;

}

@Override

protected void startCurrentDoc() {

if (packedPositions != null) {

/\*\*

\* Remember where we were at the beginning of this doc, so that we can iterate over the

\* positions for this doc if needed.

\* Adjust by `- 1 - getCurrentFreq()` because we already advanced beyond the last posting in

\* the previous loadNextPosting() calls.

\* @see #nextDocNoDel()

\*/

currentPositionPointer = nextPostingPointer - 1 - getCurrentFreq();

}

}

@Override

protected void skipTo(int target) {

assert target != NO\_MORE\_DOCS : "Should be handled in parent class advance method";

// now we know there must be a doc in this block that we can return

int skipIndex = nextPostingPointer + SKIP\_INTERVAL;

while (skipIndex <= lastPostingPointer && target > packedDocIds.get(skipIndex)) {

nextPostingPointer = skipIndex;

skipIndex += SKIP\_INTERVAL;

}

}

@Override

public int nextPosition() throws IOException {

if (packedPositions == null) {

return -1;

} else if (currentPositionPointer < packedPositions.size()) {

return (int) packedPositions.get(currentPositionPointer++);

} else {

return -1;

}

}

@Override

public int getLargestDocID() throws IOException {

return largestDocID;

}

@Override

public long cost() {

// cost would be -1 if this enum is exhausted.

final int cost = lastPostingPointer - nextPostingPointer + 1;

return cost < 0 ? 0 : cost;

}

}