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

import java.io.IOException;

import org.apache.lucene.util.BytesRef;

/\*\*

\* Extend {@link EarlybirdPostingsEnum} to add more functionalities for docs (and positions)

\* enumerator of {@link OptimizedPostingLists}.

\*/

public abstract class EarlybirdOptimizedPostingsEnum extends EarlybirdPostingsEnum {

/\*\* Current doc and its frequency. \*/

private int currentDocID = -1;

private int currentFreq = 0;

/\*\*

\* Next doc and its frequency.

\* These values should be set at {@link #loadNextPosting()}.

\*/

protected int nextDocID;

protected int nextFreq;

/\*\* Pointer to the enumerated posting list. \*/

protected final int postingListPointer;

/\*\* Total number of postings in the enumerated posting list. \*/

protected final int numPostingsTotal;

/\*\* Query cost tracker. \*/

protected final QueryCostTracker queryCostTracker;

/\*\*

\* Sole constructor.

\*

\* @param postingListPointer pointer to the posting list for which this enumerator is created

\* @param numPostings number of postings in the posting list for which this enumerator is created

\*/

public EarlybirdOptimizedPostingsEnum(int postingListPointer, int numPostings) {

this.postingListPointer = postingListPointer;

this.numPostingsTotal = numPostings;

// Get the thread local query cost tracker.

this.queryCostTracker = QueryCostTracker.getTracker();

}

/\*\*

\* Set {@link #currentDocID} and {@link #currentFreq} and load next posting.

\* This method will de-dup if duplicate doc IDs are stored.

\*

\* @return {@link #currentDocID}

\* @see {@link #nextDoc()}

\*/

@Override

protected final int nextDocNoDel() throws IOException {

currentDocID = nextDocID;

// Return immediately if exhausted.

if (currentDocID == NO\_MORE\_DOCS) {

return NO\_MORE\_DOCS;

}

currentFreq = nextFreq;

loadNextPosting();

// In case duplicate doc ID is stored.

while (currentDocID == nextDocID) {

currentFreq += nextFreq;

loadNextPosting();

}

startCurrentDoc();

return currentDocID;

}

/\*\*

\* Called when {@link #nextDocNoDel()} advances to a new docID.

\* Subclasses can do extra accounting as needed.

\*/

protected void startCurrentDoc() {

// No-op in this class.

}

/\*\*

\* Loads the next posting, setting the nextDocID and nextFreq.

\*

\* @see #nextDocNoDel()

\*/

protected abstract void loadNextPosting();

/\*\*

\* Subclass should implement {@link #skipTo(int)}.

\*

\* @see org.apache.lucene.search.DocIdSetIterator#advance(int)

\*/

@Override

public final int advance(int target) throws IOException {

// Skipping to NO\_MORE\_DOCS or beyond largest doc ID.

if (target == NO\_MORE\_DOCS || target > getLargestDocID()) {

currentDocID = nextDocID = NO\_MORE\_DOCS;

currentFreq = nextFreq = 0;

return NO\_MORE\_DOCS;

}

// Skip as close as possible.

skipTo(target);

// Calling nextDoc to reach the target, or go beyond it if target does not exist.

int doc;

do {

doc = nextDoc();

} while (doc < target);

return doc;

}

/\*\*

\* Used in {@link #advance(int)}.

\* This method should skip to the given target as close as possible, but NOT reach the target.

\*

\* @see #advance(int)

\*/

protected abstract void skipTo(int target);

/\*\*

\* Return loaded {@link #currentFreq}.

\*

\* @see org.apache.lucene.index.PostingsEnum#freq()

\* @see #nextDocNoDel()

\*/

@Override

public final int freq() throws IOException {

return currentFreq;

}

/\*\*

\* Return loaded {@link #currentDocID}.

\*

\* @see org.apache.lucene.index.PostingsEnum#docID() ()

\* @see #nextDocNoDel()

\*/

@Override

public final int docID() {

return currentDocID;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Not Supported Information \*

\* @see org.apache.lucene.index.PostingsEnum \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

@Override

public int nextPosition() throws IOException {

return -1;

}

@Override

public int startOffset() throws IOException {

return -1;

}

@Override

public int endOffset() throws IOException {

return -1;

}

@Override

public BytesRef getPayload() throws IOException {

return null;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Helper methods for subclasses \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

protected int getCurrentFreq() {

return currentFreq;

}

}