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

import java.io.IOException;

/\*\*

\* Docs and frequencies enumerator for {@link HighDFPackedIntsPostingLists}.

\*/

public class HighDFPackedIntsDocsEnum extends EarlybirdOptimizedPostingsEnum {

/\*\*

\* Pre-computed shifts, masks for {@link #deltaFreqListsReader}.

\* These pre-computed values should be read-only and shared across all reader threads.

\*

\* Notice:

\* - start int indices are NOT needed since there is not jumping within a slice.

\*/

private static final PackedLongsReaderPreComputedValues PRE\_COMPUTED\_VALUES =

new PackedLongsReaderPreComputedValues(

HighDFPackedIntsPostingLists.MAX\_DOC\_ID\_BIT

+ HighDFPackedIntsPostingLists.MAX\_FREQ\_BIT,

HighDFPackedIntsPostingLists.NUM\_BITS\_PER\_SLICE,

HighDFPackedIntsPostingLists.SLICE\_SIZE,

false);

/\*\* Packed ints reader for delta-freq pairs. \*/

private final IntBlockPoolPackedLongsReader deltaFreqListsReader;

/\*\* Skip list reader. \*/

protected final HighDFPackedIntsSkipListReader skipListReader;

/\*\* Number of remaining docs (delta-freq pairs) in a slice. \*/

private int numDocsRemaining;

/\*\*

\* Total number of docs (delta-freq pairs) in a slice.

\* This value is set every time a slice is loaded in {@link #loadNextDeltaFreqSlice()}.

\*/

private int numDocsInSliceTotal;

/\*\*

\* Number of bits used for frequency in a delta-freq slice.

\* This value is set every time a slice is loaded in {@link #loadNextDeltaFreqSlice()}.

\*/

private int bitsForFreq;

/\*\*

\* Frequency mask used to extract frequency from a delta-freq pair, in a delta-freq slice.

\* This value is set every time a slice is loaded in {@link #loadNextDeltaFreqSlice()}.

\*/

private int freqMask;

private boolean freqBitsIsZero;

/\*\*

\* Sole constructor.

\*

\* @param skipLists skip lists int block pool

\* @param deltaFreqLists delta-freq lists int block pool

\* @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

\* @param omitPositions whether positions are omitted in the posting list of which this enumerator

\* is created

\*/

public HighDFPackedIntsDocsEnum(

IntBlockPool skipLists,

IntBlockPool deltaFreqLists,

int postingListPointer,

int numPostings,

boolean omitPositions) {

super(postingListPointer, numPostings);

// Create skip list reader and get first skip entry.

this.skipListReader = new HighDFPackedIntsSkipListReader(

skipLists, postingListPointer, omitPositions);

this.skipListReader.getNextSkipEntry();

// Set number of remaining docs in this posting list.

this.numDocsRemaining = skipListReader.getNumDocsTotal();

// Create a delta-freq pair packed values reader.

this.deltaFreqListsReader = new IntBlockPoolPackedLongsReader(

deltaFreqLists,

PRE\_COMPUTED\_VALUES,

queryCostTracker,

QueryCostTracker.CostType.LOAD\_OPTIMIZED\_POSTING\_BLOCK);

loadNextDeltaFreqSlice();

loadNextPosting();

}

/\*\*

\* Load next delta-freq slice, return false if all docs exhausted.

\* Notice!! The caller of this method should make sure the current slice is all used up and

\* {@link #numDocsRemaining} is updated accordingly.

\*

\* @return whether a slice is loaded.

\* @see #loadNextPosting()

\* @see #skipTo(int)

\*/

private boolean loadNextDeltaFreqSlice() {

// Load nothing if no docs are remaining.

if (numDocsRemaining == 0) {

return false;

}

final int encodedMetadata = skipListReader.getEncodedMetadataCurrentSlice();

final int bitsForDelta = HighDFPackedIntsPostingLists.getNumBitsForDelta(encodedMetadata);

bitsForFreq = HighDFPackedIntsPostingLists.getNumBitsForFreq(encodedMetadata);

numDocsInSliceTotal = HighDFPackedIntsPostingLists.getNumDocsInSlice(encodedMetadata);

freqMask = (1 << bitsForFreq) - 1;

freqBitsIsZero = bitsForFreq == 0;

// Locate and reset the reader for this slice.

final int bitsPerPackedValue = bitsForDelta + bitsForFreq;

deltaFreqListsReader.jumpToInt(

skipListReader.getDeltaFreqCurrentSlicePointer(), bitsPerPackedValue);

return true;

}

/\*\*

\* Load next delta-freq pair from the current slice and set the computed

\* {@link #nextDocID} and {@link #nextFreq}.

\*/

@Override

protected final void loadNextPosting() {

assert numDocsRemaining >= (numDocsInSliceTotal - deltaFreqListsReader.getPackedValueIndex())

: "numDocsRemaining should be equal to or greater than number of docs remaining in slice";

if (deltaFreqListsReader.getPackedValueIndex() < numDocsInSliceTotal) {

// Current slice is not exhausted.

final long nextDeltaFreqPair = deltaFreqListsReader.readPackedLong();

/\*\*

\* Optimization: No need to do shifts and masks if number of bits for frequency is 0.

\* Also, the stored frequency is the actual frequency - 1.

\* @see

\* HighDFPackedIntsPostingLists#copyPostingList(org.apache.lucene.index.PostingsEnum, int)

\*/

if (freqBitsIsZero) {

nextFreq = 1;

nextDocID += (int) nextDeltaFreqPair;

} else {

nextFreq = (int) ((nextDeltaFreqPair & freqMask) + 1);

nextDocID += (int) (nextDeltaFreqPair >>> bitsForFreq);

}

numDocsRemaining--;

} else {

// Current slice is exhausted, get next skip entry and load next slice.

skipListReader.getNextSkipEntry();

if (loadNextDeltaFreqSlice()) {

// Next slice is loaded, load next posting again.

loadNextPosting();

} else {

// All docs are exhausted, mark this enumerator as exhausted.

assert numDocsRemaining == 0;

nextDocID = NO\_MORE\_DOCS;

nextFreq = 0;

}

}

}

/\*\*

\* Skip over slices to approach the given target as close as possible.

\*/

@Override

protected final void skipTo(int target) {

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

int numSlicesToSkip = 0;

int numDocsToSkip = 0;

int numDocsRemainingInSlice = numDocsInSliceTotal - deltaFreqListsReader.getPackedValueIndex();

// Skipping over slices.

while (skipListReader.peekPreviousDocIDNextSlice() < target) {

skipListReader.getNextSkipEntry();

nextDocID = skipListReader.getPreviousDocIDCurrentSlice();

numDocsToSkip += numDocsRemainingInSlice;

int header = skipListReader.getEncodedMetadataCurrentSlice();

numDocsRemainingInSlice = HighDFPackedIntsPostingLists.getNumDocsInSlice(header);

numSlicesToSkip++;

}

// If skipped any slices, load the new slice.

if (numSlicesToSkip > 0) {

numDocsRemaining -= numDocsToSkip;

final boolean hasNextSlice = loadNextDeltaFreqSlice();

assert hasNextSlice;

assert numDocsRemaining >= numDocsInSliceTotal && numDocsInSliceTotal > 0;

// Do additional skip for the delta freq slice that was just loaded.

doAdditionalSkip();

loadNextPosting();

}

}

/\*\*

\* Subclass should override this method if want to do additional skip on its data structure.

\*/

protected void doAdditionalSkip() {

// No-op in this class.

}

/\*\*

\* Get the largest doc ID from {@link #skipListReader}.

\*/

@Override

public int getLargestDocID() throws IOException {

return skipListReader.getLargestDocID();

}

/\*\*

\* Return {@link #numDocsRemaining} as a proxy of cost.

\*

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

\*/

@Override

public long cost() {

return numDocsRemaining;

}

}