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

import java.io.IOException;

/\*\*

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

\*/

public class HighDFPackedIntsDocsAndPositionsEnum extends HighDFPackedIntsDocsEnum {

/\*\*

\* Pre-computed shifts, masks, and start int indices for {@link #positionListsReader}.

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

\*

\* Notice:

\* - start int indices are NEEDED since there IS jumping within a slice in

\* {@link #doAdditionalSkip()} and {@link #startCurrentDoc()}.

\*/

private static final PackedLongsReaderPreComputedValues PRE\_COMPUTED\_VALUES =

new PackedLongsReaderPreComputedValues(

HighDFPackedIntsPostingLists.MAX\_POSITION\_BIT,

HighDFPackedIntsPostingLists.POSITION\_SLICE\_NUM\_BITS\_WITHOUT\_HEADER,

HighDFPackedIntsPostingLists.POSITION\_SLICE\_SIZE\_WITHOUT\_HEADER,

true);

/\*\*

\* Int block pool holding the positions for the read posting list. This is mainly used while

\* reading slice headers in {@link #loadNextPositionSlice()}.

\*/

private final IntBlockPool positionLists;

/\*\* Packed ints reader for positions. \*/

private final IntBlockPoolPackedLongsReader positionListsReader;

/\*\* Total number of positions in the current position slice. \*/

private int numPositionsInSliceTotal;

/\*\*

\* Number of remaining positions for {@link #currentDocID}; this value is decremented every time

\* {@link #nextPosition()} is called.

\*/

private int numPositionsRemainingForCurrentDocID;

/\*\*

\* Pointer to the first int, which contains the position slice header, of the next position slice.

\* This value is used to track which slice will be loaded when {@link #loadNextPositionSlice()} is

\* called.

\*/

private int nextPositionSlicePointer;

/\*\*

\* Create a docs and positions enumerator.

\*/

public HighDFPackedIntsDocsAndPositionsEnum(

IntBlockPool skipLists,

IntBlockPool deltaFreqLists,

IntBlockPool positionLists,

int postingListPointer,

int numPostings,

boolean omitPositions) {

super(skipLists, deltaFreqLists, postingListPointer, numPostings, omitPositions);

this.positionLists = positionLists;

this.positionListsReader = new IntBlockPoolPackedLongsReader(

positionLists,

PRE\_COMPUTED\_VALUES,

queryCostTracker,

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

// Load the first position slice.

this.nextPositionSlicePointer = skipListReader.getPositionCurrentSlicePointer();

loadNextPositionSlice();

}

/\*\*

\* Prepare for current doc:

\* - skipping over unread positions for the current doc.

\* - reset remaining positions for current doc to {@link #currentFreq}.

\*

\* @see #nextDocNoDel()

\*/

@Override

protected void startCurrentDoc() {

// Locate next position for current doc by skipping over unread positions from the previous doc.

if (numPositionsRemainingForCurrentDocID != 0) {

int numPositionsRemainingInSlice =

numPositionsInSliceTotal - positionListsReader.getPackedValueIndex();

while (numPositionsRemainingInSlice <= numPositionsRemainingForCurrentDocID) {

numPositionsRemainingForCurrentDocID -= numPositionsRemainingInSlice;

nextPositionSlicePointer += HighDFPackedIntsPostingLists.SLICE\_SIZE;

loadNextPositionSlice();

numPositionsRemainingInSlice = numPositionsInSliceTotal;

}

positionListsReader.setPackedValueIndex(

positionListsReader.getPackedValueIndex() + numPositionsRemainingForCurrentDocID);

}

// Number of remaining positions for current doc is current freq.

numPositionsRemainingForCurrentDocID = getCurrentFreq();

}

/\*\*

\* Put positions reader to the start of next position slice and reset number of bits per packed

\* value for next position slice.

\*/

private void loadNextPositionSlice() {

final int header = positionLists.get(nextPositionSlicePointer);

final int bitsForPosition = HighDFPackedIntsPostingLists.getNumBitsForPosition(header);

numPositionsInSliceTotal = HighDFPackedIntsPostingLists.getNumPositionsInSlice(header);

positionListsReader.jumpToInt(

nextPositionSlicePointer + HighDFPackedIntsPostingLists.POSITION\_SLICE\_HEADER\_SIZE,

bitsForPosition);

}

/\*\*

\* Return next position for current doc.

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

\*/

@Override

public int nextPosition() throws IOException {

// Return -1 immediately if all positions are used up for current doc.

if (numPositionsRemainingForCurrentDocID == 0) {

return -1;

}

if (positionListsReader.getPackedValueIndex() < numPositionsInSliceTotal) {

// Read next position in current slice.

final int nextPosition = (int) positionListsReader.readPackedLong();

numPositionsRemainingForCurrentDocID--;

return nextPosition;

} else {

// All positions in current slice is used up, load next slice.

nextPositionSlicePointer += HighDFPackedIntsPostingLists.SLICE\_SIZE;

loadNextPositionSlice();

return nextPosition();

}

}

/\*\*

\* Set {@link #positionListsReader} to the correct location and correct number of bits per packed

\* value for the delta-freq slice on which this enum is landed after skipping.

\*

\* @see #skipTo(int)

\*/

@Override

protected void doAdditionalSkip() {

nextPositionSlicePointer = skipListReader.getPositionCurrentSlicePointer();

loadNextPositionSlice();

// Locate the exact position in slice.

final int skipListEntryEncodedMetadata = skipListReader.getEncodedMetadataCurrentSlice();

positionListsReader.setPackedValueIndex(

HighDFPackedIntsPostingLists.getPositionOffsetInSlice(skipListEntryEncodedMetadata));

numPositionsRemainingForCurrentDocID = 0;

}

}