package com.twitter.search.core.earlybird.facets;

import java.util.Map;

import java.util.Map.Entry;

import com.google.common.base.Preconditions;

import org.apache.lucene.facet.FacetsConfig;

import org.apache.lucene.index.ReaderUtil;

import org.apache.lucene.index.SortedSetDocValues;

import org.apache.lucene.util.BytesRef;

import org.apache.lucene.util.BytesRefBuilder;

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

public class EarlybirdFacetDocValueSet extends SortedSetDocValues {

private final AbstractFacetCountingArray countingArray;

private final InvertedIndex[] labelProviders;

private final String[] fieldNames;

private final int[] starts;

private final BytesRefBuilder ordCache;

private int totalTerms;

private int docID = -1;

private int currentFacet = FacetCountingArray.UNASSIGNED;

private int pointer = -1;

private boolean hasMoreOrds = false;

public static final String FIELD\_NAME = FacetsConfig.DEFAULT\_INDEX\_FIELD\_NAME;

/\*\*

\* Creates a new EarlybirdFacetDocValueSet from the provided FacetCountingArray.

\*/

public EarlybirdFacetDocValueSet(AbstractFacetCountingArray countingArray,

Map<String, FacetLabelProvider> labelProviderMap,

FacetIDMap facetIdMap) {

this.countingArray = countingArray;

labelProviders = new InvertedIndex[facetIdMap.getNumberOfFacetFields()];

fieldNames = new String[facetIdMap.getNumberOfFacetFields()];

for (Entry<String, FacetLabelProvider> entry : labelProviderMap.entrySet()) {

FacetLabelProvider labelProvider = entry.getValue();

if (labelProvider instanceof InvertedIndex) {

FacetIDMap.FacetField facetField = facetIdMap.getFacetFieldByFacetName(entry.getKey());

if (facetField != null) {

labelProviders[facetField.getFacetId()] = (InvertedIndex) labelProvider;

fieldNames[facetField.getFacetId()] = entry.getKey();

}

}

}

starts = new int[labelProviders.length + 1]; // build starts array

ordCache = new BytesRefBuilder();

totalTerms = 0;

for (int i = 0; i < labelProviders.length; ++i) {

if (labelProviders[i] != null) {

starts[i] = totalTerms;

int termCount = labelProviders[i].getNumTerms();

totalTerms += termCount;

}

}

// added to so that mapping from ord to index works via ReaderUtil.subIndex

starts[labelProviders.length] = totalTerms;

}

private long encodeOrd(int fieldId, int termId) {

assert starts[fieldId] + termId < starts[fieldId + 1];

return starts[fieldId] + termId;

}

@Override

public long nextOrd() {

if (!hasMoreOrds || currentFacet == FacetCountingArray.UNASSIGNED) {

return SortedSetDocValues.NO\_MORE\_ORDS;

}

// only 1 facet val

if (!FacetCountingArray.isPointer(currentFacet)) {

int termId = FacetCountingArray.decodeTermID(currentFacet);

int fieldId = FacetCountingArray.decodeFieldID(currentFacet);

hasMoreOrds = false;

return encodeOrd(fieldId, termId);

}

// multiple facets, follow the pointer to find all facets in the facetsPool.

if (pointer == -1) {

pointer = FacetCountingArray.decodePointer(currentFacet);

}

int facetID = countingArray.getFacetsPool().get(pointer);

int termId = FacetCountingArray.decodeTermID(facetID);

int fieldId = FacetCountingArray.decodeFieldID(facetID);

hasMoreOrds = FacetCountingArray.isPointer(facetID);

pointer++;

return encodeOrd(fieldId, termId);

}

@Override

public BytesRef lookupOrd(long ord) {

int idx = ReaderUtil.subIndex((int) ord, this.starts);

if (labelProviders[idx] != null) {

int termID = (int) ord - starts[idx];

BytesRef term = new BytesRef();

labelProviders[idx].getTerm(termID, term);

String name = fieldNames[idx];

String val = FacetsConfig.pathToString(new String[] {name, term.utf8ToString()});

ordCache.copyChars(val);

} else {

ordCache.copyChars("");

}

return ordCache.get();

}

@Override

public long lookupTerm(BytesRef key) {

throw new UnsupportedOperationException();

}

@Override

public long getValueCount() {

return totalTerms;

}

@Override

public int docID() {

return docID;

}

@Override

public int nextDoc() {

return ++docID;

}

@Override

public int advance(int target) {

Preconditions.checkState(target >= docID);

docID = target;

currentFacet = countingArray.getFacet(docID);

pointer = -1;

hasMoreOrds = true;

return docID;

}

@Override

public boolean advanceExact(int target) {

return advance(target) != FacetCountingArray.UNASSIGNED;

}

@Override

public long cost() {

return totalTerms;

}

}