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

import java.io.IOException;

import java.util.Map;

import java.util.Set;

import com.google.common.collect.Sets;

import org.apache.lucene.index.FieldInfos;

import org.apache.lucene.index.Fields;

import org.apache.lucene.index.LeafReader;

import org.apache.lucene.index.NumericDocValues;

import org.apache.lucene.index.PostingsEnum;

import org.apache.lucene.index.Term;

import org.apache.lucene.search.DocIdSetIterator;

import com.twitter.search.common.schema.base.Schema;

import com.twitter.search.core.earlybird.facets.AbstractFacetCountingArray;

import com.twitter.search.core.earlybird.facets.FacetIDMap;

import com.twitter.search.core.earlybird.facets.FacetLabelProvider;

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

/\*\*

\* Base class for atomic Earlybird segment readers.

\*/

public abstract class EarlybirdIndexSegmentAtomicReader extends LeafReader {

public static final int TERM\_NOT\_FOUND = -1;

private final DeletedDocs.View deletesView;

private final EarlybirdIndexSegmentData segmentData;

protected final EarlybirdIndexSegmentData.SyncData syncData;

private FieldInfos fieldInfos;

/\*\*

\* Creates a new atomic reader for this Earlybird segment.

\*/

public EarlybirdIndexSegmentAtomicReader(EarlybirdIndexSegmentData segmentData) {

super();

this.segmentData = segmentData;

this.syncData = segmentData.getSyncData();

this.deletesView = segmentData.getDeletedDocs().getView();

// fieldInfos will be initialized lazily if required

this.fieldInfos = null;

}

public int getSmallestDocID() {

return syncData.getSmallestDocID();

}

public final FacetIDMap getFacetIDMap() {

return segmentData.getFacetIDMap();

}

public final Map<String, FacetLabelProvider> getFacetLabelProviders() {

return segmentData.getFacetLabelProviders();

}

public AbstractFacetCountingArray getFacetCountingArray() {

return segmentData.getFacetCountingArray();

}

public final FacetLabelProvider getFacetLabelProviders(Schema.FieldInfo field) {

String facetName = field.getFieldType().getFacetName();

return facetName != null && segmentData.getFacetLabelProviders() != null

? segmentData.getFacetLabelProviders().get(facetName) : null;

}

@Override

public FieldInfos getFieldInfos() {

if (fieldInfos == null) {

// TwitterInMemoryIndexReader is constructed per query, and this call is only needed for

// optimize. We wouldn't want to create a new FieldInfos per search, so we deffer it.

Schema schema = segmentData.getSchema();

final Set<String> fieldSet = Sets.newHashSet(segmentData.getPerFieldMap().keySet());

fieldSet.addAll(segmentData.getDocValuesManager().getDocValueNames());

fieldInfos = schema.getLuceneFieldInfos(input -> input != null && fieldSet.contains(input));

}

return fieldInfos;

}

/\*\*

\* Returns the ID that was assigned to the given term in

\* {@link com.twitter.search.core.earlybird.index.inverted.InvertedRealtimeIndex}

\*/

public abstract int getTermID(Term t) throws IOException;

/\*\*

\* Returns the oldest posting for the given term

\* NOTE: This method may return a deleted doc id.

\*/

public abstract int getOldestDocID(Term t) throws IOException;

@Override

public abstract NumericDocValues getNumericDocValues(String field) throws IOException;

/\*\*

\* Determines if this reader has any documents to traverse. Note that it is possible for the tweet

\* ID mapper to have documents, but for this reader to not see them yet. In this case, this method

\* will return false.

\*/

public boolean hasDocs() {

return segmentData.numDocs() > 0;

}

/\*\*

\* Returns the newest posting for the given term

\*/

public final int getNewestDocID(Term term) throws IOException {

PostingsEnum td = postings(term);

if (td == null) {

return EarlybirdIndexSegmentAtomicReader.TERM\_NOT\_FOUND;

}

if (td.nextDoc() != DocIdSetIterator.NO\_MORE\_DOCS) {

return td.docID();

} else {

return EarlybirdIndexSegmentAtomicReader.TERM\_NOT\_FOUND;

}

}

public final DeletedDocs.View getDeletesView() {

return deletesView;

}

@Override

public final Fields getTermVectors(int docID) {

// Earlybird does not use term vectors.

return null;

}

public EarlybirdIndexSegmentData getSegmentData() {

return segmentData;

}

public Schema getSchema() {

return segmentData.getSchema();

}

}