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

import java.io.Closeable;

import java.io.IOException;

import org.apache.lucene.document.Document;

import org.apache.lucene.index.IndexReader;

import org.apache.lucene.index.LeafReaderContext;

import org.apache.lucene.search.Collector;

import org.apache.lucene.search.IndexSearcher;

import org.apache.lucene.search.LeafCollector;

import org.apache.lucene.search.Query;

import org.apache.lucene.search.Scorable;

import org.apache.lucene.search.ScoreMode;

import org.apache.lucene.store.Directory;

import com.twitter.search.core.earlybird.index.column.ColumnStrideFieldIndex;

import com.twitter.search.core.earlybird.index.column.DocValuesUpdate;

/\*\*

\* IndexSegmentWriter combines some common functionality between the Lucene and Realtime index

\* segment writers.

\*/

public abstract class EarlybirdIndexSegmentWriter implements Closeable {

public EarlybirdIndexSegmentWriter() {

}

/\*\*

\* Gets the segment data this segment write is associated with.

\* @return

\*/

public abstract EarlybirdIndexSegmentData getSegmentData();

/\*\*

\* Appends terms from the document to the document matching the query. Does not replace a field or

\* document, actually adds to the the field in the segment.

\*/

public final void appendOutOfOrder(Query query, Document doc) throws IOException {

runQuery(query, docID -> appendOutOfOrder(doc, docID));

}

protected abstract void appendOutOfOrder(Document doc, int docId) throws IOException;

/\*\*

\* Deletes a document in this segment that matches this query.

\*/

public void deleteDocuments(Query query) throws IOException {

runQuery(query, docID -> getSegmentData().getDeletedDocs().deleteDoc(docID));

}

/\*\*

\* Updates the docvalues of a document in this segment that matches this query.

\*/

public void updateDocValues(Query query, String field, DocValuesUpdate update)

throws IOException {

runQuery(query, docID -> {

ColumnStrideFieldIndex docValues =

getSegmentData().getDocValuesManager().getColumnStrideFieldIndex(field);

if (docValues == null) {

return;

}

update.update(docValues, docID);

});

}

private void runQuery(final Query query, final OnHit onHit) throws IOException {

try (IndexReader reader = getSegmentData().createAtomicReader()) {

new IndexSearcher(reader).search(query, new Collector() {

@Override

public LeafCollector getLeafCollector(LeafReaderContext context) throws IOException {

return new LeafCollector() {

@Override

public void setScorer(Scorable scorer) {

}

@Override

public void collect(int docID) throws IOException {

onHit.hit(docID);

}

};

}

@Override

public ScoreMode scoreMode() {

return ScoreMode.COMPLETE\_NO\_SCORES;

}

});

}

}

private interface OnHit {

void hit(int docID) throws IOException;

}

/\*\*

\* Adds a new document to this segment. In production, this method should be called only by

\* Expertsearch.

\*/

public abstract void addDocument(Document doc) throws IOException;

/\*\*

\* Adds a new tweet to this segment. This method should be called only by Earlybird.

\*/

public abstract void addTweet(Document doc, long tweetId, boolean docIsOffensive)

throws IOException;

/\*\*

\* Returns the total number of documents in the segment.

\*/

public abstract int numDocs() throws IOException;

/\*\*

\* Returns the number of documents in this segment without taking deleted docs into account.

\* E.g. if 10 documents were added to this segments, and 5 were deleted,

\* this method still returns 10.

\*/

public abstract int numDocsNoDelete() throws IOException;

/\*\*

\* Forces the underlying index to be merged down to a single segment.

\*/

public abstract void forceMerge() throws IOException;

/\*\*

\* Appends the provides Lucene indexes to this segment.

\*/

public abstract void addIndexes(Directory... dirs) throws IOException;

}