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

import java.io.IOException;

import com.google.common.base.Preconditions;

import org.apache.lucene.facet.FacetsConfig;

import org.apache.lucene.index.DocValuesType;

import org.apache.lucene.index.IndexableField;

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

import com.twitter.search.core.earlybird.index.EarlybirdRealtimeIndexSegmentWriter;

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

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

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

/\*\*

\* Handler for docvalues in the indexing chain.

\*/

public class EarlybirdCSFDocValuesProcessor

implements EarlybirdRealtimeIndexSegmentWriter.StoredFieldsConsumer {

private final DocValuesManager docValuesManager;

public EarlybirdCSFDocValuesProcessor(DocValuesManager docValuesManager) {

this.docValuesManager = docValuesManager;

}

@Override

public void addField(int docID, IndexableField field) throws IOException {

final DocValuesType dvType = field.fieldType().docValuesType();

if (dvType != null) {

// ignore lucene facet fields for realtime index, we are handling it differently

if (field.name().startsWith(FacetsConfig.DEFAULT\_INDEX\_FIELD\_NAME)) {

return;

}

if (!(field.fieldType() instanceof EarlybirdFieldType)) {

throw new RuntimeException(

"fieldType must be an EarlybirdFieldType instance for field " + field.name());

}

EarlybirdFieldType fieldType = (EarlybirdFieldType) field.fieldType();

if (dvType == DocValuesType.NUMERIC) {

if (!(field.numericValue() instanceof Long)) {

throw new IllegalArgumentException(

"illegal type " + field.numericValue().getClass()

+ ": DocValues types must be Long");

}

ColumnStrideFieldIndex csfIndex =

docValuesManager.addColumnStrideField(field.name(), fieldType);

if (fieldType.getCsfFixedLengthNumValuesPerDoc() > 1) {

throw new UnsupportedOperationException("unsupported multi numeric values");

} else {

csfIndex.setValue(docID, field.numericValue().longValue());

}

} else if (dvType == DocValuesType.BINARY) {

ColumnStrideFieldIndex csfIndex =

docValuesManager.addColumnStrideField(field.name(), fieldType);

if (fieldType.getCsfFixedLengthNumValuesPerDoc() > 1) {

Preconditions.checkArgument(

csfIndex instanceof AbstractColumnStrideMultiIntIndex,

"Unsupported multi-value binary CSF class: " + csfIndex);

((AbstractColumnStrideMultiIntIndex) csfIndex).updateDocValues(

field.binaryValue(), docID);

}

} else {

throw new UnsupportedOperationException("unsupported DocValues.Type: " + dvType);

}

}

}

}