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

import java.io.IOException;

import java.util.HashSet;

import java.util.Iterator;

import java.util.Set;

import org.apache.lucene.analysis.TokenStream;

import org.apache.lucene.analysis.tokenattributes.CharTermAttribute;

import org.apache.lucene.index.Term;

import org.apache.lucene.search.BooleanClause;

import org.apache.lucene.search.BooleanQuery;

import org.apache.lucene.search.Query;

import org.apache.lucene.search.TermQuery;

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

import com.twitter.search.common.schema.earlybird.EarlybirdFieldConstants.EarlybirdFieldConstant;

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

import com.twitter.search.earlybird.thrift.ThriftTermRequest;

public abstract class FacetSkipList {

public static class SkipTokenStream extends TokenStream {

private CharTermAttribute termAtt = addAttribute(CharTermAttribute.class);

private Iterator<Schema.FieldInfo> iterator;

private Set<Schema.FieldInfo> facetFields = new HashSet<>();

public void add(Schema.FieldInfo field) {

this.facetFields.add(field);

}

@Override

public final boolean incrementToken() throws IOException {

if (iterator == null) {

iterator = facetFields.iterator();

}

while (iterator.hasNext()) {

Schema.FieldInfo field = iterator.next();

if (field.getFieldType().isStoreFacetSkiplist()) {

termAtt.setEmpty();

termAtt.append(EarlybirdFieldConstant.getFacetSkipFieldName(field.getName()));

return true;

}

}

return false;

}

}

/\*\*

\* Returns a Term query to search in the given facet field.

\*/

public static Term getSkipListTerm(Schema.FieldInfo facetField) {

if (facetField.getFieldType().isStoreFacetSkiplist()) {

return new Term(EarlybirdFieldConstant.INTERNAL\_FIELD.getFieldName(),

EarlybirdFieldConstant.getFacetSkipFieldName(facetField.getName()));

}

return null;

}

/\*\*

\* Returns a disjunction query that searches in all facet fields in the given facet count state.

\*/

public static Query getSkipListQuery(FacetCountState facetCountState) {

Set<Schema.FieldInfo> fieldsWithSkipLists =

facetCountState.getFacetFieldsToCountWithSkipLists();

if (fieldsWithSkipLists == null || fieldsWithSkipLists.isEmpty()) {

return null;

}

Query skipLists;

if (fieldsWithSkipLists.size() == 1) {

skipLists = new TermQuery(getSkipListTerm(fieldsWithSkipLists.iterator().next()));

} else {

BooleanQuery.Builder disjunctionBuilder = new BooleanQuery.Builder();

for (Schema.FieldInfo facetField : fieldsWithSkipLists) {

disjunctionBuilder.add(

new TermQuery(new Term(

EarlybirdFieldConstant.INTERNAL\_FIELD.getFieldName(),

EarlybirdFieldConstant.getFacetSkipFieldName(facetField.getName()))),

BooleanClause.Occur.SHOULD);

}

skipLists = disjunctionBuilder.build();

}

return skipLists;

}

/\*\*

\* Returns a term request that can be used to get term statistics for the skip list term

\* associated with the provided facet. Returns null, if this FacetField is configured to not

\* store a skiplist.

\*/

public static ThriftTermRequest getSkipListTermRequest(Schema schema, String facetName) {

return getSkipListTermRequest(schema.getFacetFieldByFacetName(facetName));

}

/\*\*

\* Returns a term request that can be used to get term statistics for the skip list term

\* associated with the provided facet. Returns null, if this FacetField is configured to not

\* store a skiplist.

\*/

public static ThriftTermRequest getSkipListTermRequest(Schema.FieldInfo facetField) {

return facetField != null && facetField.getFieldType().isStoreFacetSkiplist()

? new ThriftTermRequest(

EarlybirdFieldConstant.getFacetSkipFieldName(facetField.getName()))

.setFieldName(EarlybirdFieldConstant.INTERNAL\_FIELD.getFieldName())

: null;

}

/\*\*

\* Returns a term request using the specified fieldName. This is only a temporary solution until

\* Blender can access the Schema to pass the FacetIDMap into the method above.

\*

\* @deprecated Temporary solution until Blender

\*/

@Deprecated

public static ThriftTermRequest getSkipListTermRequest(String fieldName) {

return new ThriftTermRequest(EarlybirdFieldConstant.getFacetSkipFieldName(fieldName))

.setFieldName(EarlybirdFieldConstant.INTERNAL\_FIELD.getFieldName());

}

}