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

import java.util.List;

import java.util.Map;

import java.util.Map.Entry;

import com.google.common.base.Preconditions;

import com.google.common.collect.Maps;

import org.apache.lucene.facet.FacetResult;

import com.twitter.search.common.facets.CountFacetSearchParam;

import com.twitter.search.common.facets.FacetSearchParam;

import com.twitter.search.common.facets.thriftjava.FacetFieldRequest;

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

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

/\*\*

\* Global facet aggregator across all fields.

\*

\*/

public class FacetCountAggregator implements FacetTermCollector {

// keys for the following aggregators are fieldIds

private final Map<Integer, PerfieldFacetCountAggregator> aggregators;

private final Map<Integer, FacetSearchParam> facetSearchParamMap;

/\*\*

\* Creates a new facet aggregator.

\*/

public FacetCountAggregator(

List<FacetSearchParam> facetSearchParams,

Schema schema,

FacetIDMap facetIDMap,

Map<String, InvertedIndex> labelProviderMap) {

aggregators = Maps.newHashMap();

facetSearchParamMap = Maps.newHashMap();

// Check params:

for (FacetSearchParam facetSearchParam : facetSearchParams) {

if (!(facetSearchParam instanceof CountFacetSearchParam)) {

throw new IllegalArgumentException(

"this collector only supports CountFacetSearchParam; got " + facetSearchParam);

}

if (facetSearchParam.getFacetFieldRequest().getPath() != null

&& !facetSearchParam.getFacetFieldRequest().getPath().isEmpty()) {

throw new IllegalArgumentException(

"this collector dosen't support hierarchical facets: "

+ facetSearchParam.getFacetFieldRequest().getPath());

}

String field = facetSearchParam.getFacetFieldRequest().getField();

Schema.FieldInfo facetField =

schema == null ? null : schema.getFacetFieldByFacetName(field);

if (facetField == null || !labelProviderMap.containsKey(facetField.getName())) {

throw new IllegalStateException("facet field: " + field + " is not defined");

}

int fieldId = facetIDMap.getFacetField(facetField).getFacetId();

Preconditions.checkState(!aggregators.containsKey(fieldId));

Preconditions.checkState(!facetSearchParamMap.containsKey(fieldId));

aggregators.put(fieldId, new PerfieldFacetCountAggregator(field,

labelProviderMap.get(facetField.getName())));

facetSearchParamMap.put(fieldId, facetSearchParam);

}

}

/\*\*

\* Returns the top facets.

\*/

public Map<FacetFieldRequest, FacetResult> getTop() {

Map<FacetFieldRequest, FacetResult> map = Maps.newHashMap();

for (Entry<Integer, PerfieldFacetCountAggregator> entry : aggregators.entrySet()) {

FacetSearchParam facetSearchParam = facetSearchParamMap.get(entry.getKey());

map.put(facetSearchParam.getFacetFieldRequest(), entry.getValue().getTop(facetSearchParam));

}

return map;

}

@Override

public boolean collect(int docID, long termID, int fieldID) {

PerfieldFacetCountAggregator perfieldAggregator = aggregators.get(fieldID);

if (perfieldAggregator != null) {

perfieldAggregator.collect((int) termID);

return true;

} else {

return false;

}

}

}