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

import java.io.IOException;

import java.util.Arrays;

import java.util.Map;

import com.twitter.search.common.util.io.flushable.DataDeserializer;

import com.twitter.search.common.util.io.flushable.DataSerializer;

import com.twitter.search.common.util.io.flushable.FlushInfo;

import com.twitter.search.common.util.io.flushable.Flushable;

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

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

public class OptimizedFacetCountingArray extends AbstractFacetCountingArray {

private final int[] facetsMap;

/\*\*

\* Creates a new, empty FacetCountingArray with the given size.

\*/

public OptimizedFacetCountingArray(int maxDocIdInclusive) {

super();

facetsMap = new int[maxDocIdInclusive];

Arrays.fill(facetsMap, UNASSIGNED);

}

private OptimizedFacetCountingArray(int[] facetsMap, IntBlockPool facetsPool) {

super(facetsPool);

this.facetsMap = facetsMap;

}

@Override

protected int getFacet(int docID) {

return facetsMap[docID];

}

@Override

protected void setFacet(int docID, int facetID) {

facetsMap[docID] = facetID;

}

@Override

public AbstractFacetCountingArray rewriteAndMapIDs(

Map<Integer, int[]> termIDMapper,

DocIDToTweetIDMapper originalTweetIdMapper,

DocIDToTweetIDMapper optimizedTweetIdMapper) {

throw new UnsupportedOperationException(

"OptimizedFacetCountingArray instances should never be rewritten.");

}

@Override

public FlushHandler getFlushHandler() {

return new FlushHandler(this);

}

public static final class FlushHandler extends Flushable.Handler<OptimizedFacetCountingArray> {

private static final String FACETS\_POOL\_PROP\_NAME = "facetsPool";

public FlushHandler() {

}

public FlushHandler(OptimizedFacetCountingArray objectToFlush) {

super(objectToFlush);

}

@Override

public void doFlush(FlushInfo flushInfo, DataSerializer out) throws IOException {

OptimizedFacetCountingArray objectToFlush = getObjectToFlush();

out.writeIntArray(objectToFlush.facetsMap);

objectToFlush.getFacetsPool().getFlushHandler().flush(

flushInfo.newSubProperties(FACETS\_POOL\_PROP\_NAME), out);

}

@Override

public OptimizedFacetCountingArray doLoad(FlushInfo flushInfo, DataDeserializer in)

throws IOException {

int[] facetsMap = in.readIntArray();

IntBlockPool facetsPool = new IntBlockPool.FlushHandler().load(

flushInfo.getSubProperties(FACETS\_POOL\_PROP\_NAME), in);

return new OptimizedFacetCountingArray(facetsMap, facetsPool);

}

}

}