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

import java.io.IOException;

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;

public class OptimizedColumnStrideMultiIntIndex

extends AbstractColumnStrideMultiIntIndex implements Flushable {

private final int[] values;

public OptimizedColumnStrideMultiIntIndex(String name, int maxSize, int numIntsPerField) {

super(name, numIntsPerField);

values = new int[Math.multiplyExact(maxSize, numIntsPerField)];

}

public OptimizedColumnStrideMultiIntIndex(

ColumnStrideMultiIntIndex columnStrideMultiIntIndex,

DocIDToTweetIDMapper originalTweetIdMapper,

DocIDToTweetIDMapper optimizedTweetIdMapper) throws IOException {

super(columnStrideMultiIntIndex.getName(), columnStrideMultiIntIndex.getNumIntsPerField());

int maxDocId = optimizedTweetIdMapper.getPreviousDocID(Integer.MAX\_VALUE);

values = new int[columnStrideMultiIntIndex.getNumIntsPerField() \* (maxDocId + 1)];

int docId = optimizedTweetIdMapper.getNextDocID(Integer.MIN\_VALUE);

while (docId != DocIDToTweetIDMapper.ID\_NOT\_FOUND) {

int originalDocId = originalTweetIdMapper.getDocID(optimizedTweetIdMapper.getTweetID(docId));

for (int i = 0; i < columnStrideMultiIntIndex.getNumIntsPerField(); ++i) {

setValue(docId, i, columnStrideMultiIntIndex.get(originalDocId, i));

}

docId = optimizedTweetIdMapper.getNextDocID(docId);

}

}

private OptimizedColumnStrideMultiIntIndex(String name, int numIntsPerField, int[] values) {

super(name, numIntsPerField);

this.values = values;

}

@Override

public void setValue(int docID, int valueIndex, int value) {

values[docID \* getNumIntsPerField() + valueIndex] = value;

}

@Override

public int get(int docID, int valueIndex) {

return values[docID \* getNumIntsPerField() + valueIndex];

}

@Override

public FlushHandler getFlushHandler() {

return new FlushHandler(this);

}

public static final class FlushHandler

extends Flushable.Handler<OptimizedColumnStrideMultiIntIndex> {

private static final String INTS\_PER\_FIELD\_PROP\_NAME = "intsPerField";

private static final String NAME\_PROP\_NAME = "fieldName";

public FlushHandler() {

super();

}

public FlushHandler(OptimizedColumnStrideMultiIntIndex objectToFlush) {

super(objectToFlush);

}

@Override

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

OptimizedColumnStrideMultiIntIndex columnStrideMultiIntIndex = getObjectToFlush();

flushInfo.addStringProperty(NAME\_PROP\_NAME, columnStrideMultiIntIndex.getName());

flushInfo.addIntProperty(INTS\_PER\_FIELD\_PROP\_NAME,

columnStrideMultiIntIndex.getNumIntsPerField());

out.writeIntArray(columnStrideMultiIntIndex.values);

}

@Override

protected OptimizedColumnStrideMultiIntIndex doLoad(FlushInfo flushInfo, DataDeserializer in)

throws IOException {

int[] values = in.readIntArray();

return new OptimizedColumnStrideMultiIntIndex(

flushInfo.getStringProperty(NAME\_PROP\_NAME),

flushInfo.getIntProperty(INTS\_PER\_FIELD\_PROP\_NAME),

values);

}

}

}