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;

import it.unimi.dsi.fastutil.ints.Int2IntOpenHashMap;

public class ColumnStrideMultiIntIndex extends AbstractColumnStrideMultiIntIndex {

private final Int2IntOpenHashMap[] values;

private final int maxSize;

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

super(name, numIntsPerField);

values = new Int2IntOpenHashMap[numIntsPerField];

for (int i = 0; i < numIntsPerField; i++) {

values[i] = new Int2IntOpenHashMap(maxSize); // default unset value is 0

}

this.maxSize = maxSize;

}

public ColumnStrideMultiIntIndex(String name, Int2IntOpenHashMap[] values, int maxSize) {

super(name, values.length);

this.values = values;

this.maxSize = maxSize;

}

@Override

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

values[valueIndex].put(docID, value);

}

@Override

public int get(int docID, int valueIndex) {

return values[valueIndex].get(docID);

}

@Override

public ColumnStrideFieldIndex optimize(

DocIDToTweetIDMapper originalTweetIdMapper,

DocIDToTweetIDMapper optimizedTweetIdMapper) throws IOException {

return new OptimizedColumnStrideMultiIntIndex(

this, originalTweetIdMapper, optimizedTweetIdMapper);

}

@Override

public FlushHandler getFlushHandler() {

return new FlushHandler(this);

}

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

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

private static final String MAX\_SIZE\_PROP = "maxSize";

public FlushHandler() {

super();

}

public FlushHandler(ColumnStrideMultiIntIndex objectToFlush) {

super(objectToFlush);

}

@Override

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

ColumnStrideMultiIntIndex index = getObjectToFlush();

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

flushInfo.addIntProperty(MAX\_SIZE\_PROP, index.maxSize);

out.writeInt(index.values.length);

for (int i = 0; i < index.values.length; i++) {

Int2IntOpenHashMap map = index.values[i];

out.writeInt(map.size());

for (Int2IntOpenHashMap.Entry entry : map.int2IntEntrySet()) {

out.writeInt(entry.getIntKey());

out.writeInt(entry.getIntValue());

}

}

}

@Override

protected ColumnStrideMultiIntIndex doLoad(FlushInfo flushInfo, DataDeserializer in)

throws IOException {

int numIntsPerField = in.readInt();

int maxSize = flushInfo.getIntProperty(MAX\_SIZE\_PROP);

Int2IntOpenHashMap[] values = new Int2IntOpenHashMap[numIntsPerField];

for (int i = 0; i < numIntsPerField; i++) {

int size = in.readInt();

Int2IntOpenHashMap map = new Int2IntOpenHashMap(maxSize);

for (int j = 0; j < size; j++) {

map.put(in.readInt(), in.readInt());

}

values[i] = map;

}

return new ColumnStrideMultiIntIndex(

flushInfo.getStringProperty(NAME\_PROP\_NAME), values, maxSize);

}

}

}