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.Int2ByteOpenHashMap;

public class ColumnStrideByteIndex extends ColumnStrideFieldIndex implements Flushable {

private final Int2ByteOpenHashMap values;

private final int maxSize;

public ColumnStrideByteIndex(String name, int maxSize) {

super(name);

values = new Int2ByteOpenHashMap(maxSize); // default unset value is 0

this.maxSize = maxSize;

}

private ColumnStrideByteIndex(String name, Int2ByteOpenHashMap values, int maxSize) {

super(name);

this.values = values;

this.maxSize = maxSize;

}

@Override

public void setValue(int docID, long value) {

values.put(docID, (byte) value);

}

@Override

public long get(int docID) {

return values.get(docID);

}

@Override

public ColumnStrideFieldIndex optimize(

DocIDToTweetIDMapper originalTweetIdMapper,

DocIDToTweetIDMapper optimizedTweetIdMapper) throws IOException {

return new OptimizedColumnStrideByteIndex(this, originalTweetIdMapper, optimizedTweetIdMapper);

}

@Override

public FlushHandler getFlushHandler() {

return new FlushHandler(this);

}

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

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

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

public FlushHandler() {

super();

}

public FlushHandler(ColumnStrideByteIndex objectToFlush) {

super(objectToFlush);

}

@Override

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

ColumnStrideByteIndex index = getObjectToFlush();

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

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

out.writeInt(index.values.size());

for (Int2ByteOpenHashMap.Entry entry : index.values.int2ByteEntrySet()) {

out.writeInt(entry.getIntKey());

out.writeByte(entry.getByteValue());

}

}

@Override

protected ColumnStrideByteIndex doLoad(FlushInfo flushInfo, DataDeserializer in)

throws IOException {

int size = in.readInt();

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

Int2ByteOpenHashMap map = new Int2ByteOpenHashMap(maxSize);

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

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

}

return new ColumnStrideByteIndex(flushInfo.getStringProperty(NAME\_PROP\_NAME), map, maxSize);

}

}

}