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

import java.io.IOException;

import com.google.common.base.Preconditions;

import org.apache.lucene.index.BinaryDocValues;

import org.apache.lucene.index.LeafReader;

import org.apache.lucene.util.BytesRef;

import com.twitter.search.common.encoding.docvalues.CSFTypeUtil;

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

public abstract class AbstractColumnStrideMultiIntIndex

extends ColumnStrideFieldIndex implements Flushable {

private static final int NUM\_BYTES\_PER\_INT = java.lang.Integer.SIZE / java.lang.Byte.SIZE;

private final int numIntsPerField;

protected AbstractColumnStrideMultiIntIndex(String name, int numIntsPerField) {

super(name);

this.numIntsPerField = numIntsPerField;

}

public int getNumIntsPerField() {

return numIntsPerField;

}

@Override

public long get(int docID) {

throw new UnsupportedOperationException();

}

/\*\*

\* Returns the value stored at the given index for the given doc ID.

\*/

public abstract int get(int docID, int valueIndex);

/\*\*

\* Sets the value stored at the given index for the given doc ID.

\*/

public abstract void setValue(int docID, int valueIndex, int val);

@Override

public void load(LeafReader atomicReader, String field) throws IOException {

BinaryDocValues docValues = atomicReader.getBinaryDocValues(field);

int numBytesPerDoc = numIntsPerField \* NUM\_BYTES\_PER\_INT;

for (int docID = 0; docID < atomicReader.maxDoc(); docID++) {

Preconditions.checkState(docValues.advanceExact(docID));

BytesRef scratch = docValues.binaryValue();

Preconditions.checkState(

scratch.length == numBytesPerDoc,

"Unexpected doc value length for field " + field

+ ": Should be " + numBytesPerDoc + ", but was " + scratch.length);

scratch.length = NUM\_BYTES\_PER\_INT;

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

setValue(docID, i, asInt(scratch));

scratch.offset += NUM\_BYTES\_PER\_INT;

}

}

}

public void updateDocValues(BytesRef ref, int docID) {

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

setValue(docID, i, CSFTypeUtil.convertFromBytes(ref.bytes, ref.offset, i));

}

}

private static int asInt(BytesRef b) {

return asInt(b, b.offset);

}

private static int asInt(BytesRef b, int pos) {

int p = pos;

return (b.bytes[p++] << 24) | (b.bytes[p++] << 16) | (b.bytes[p++] << 8) | (b.bytes[p] & 0xFF);

}

}