package com.twitter.search.common.query;

import java.io.IOException;

import java.util.Iterator;

import java.util.Set;

import org.apache.lucene.index.FilteredTermsEnum;

import org.apache.lucene.index.Terms;

import org.apache.lucene.index.TermsEnum;

import org.apache.lucene.search.MultiTermQuery;

import org.apache.lucene.util.AttributeSource;

import org.apache.lucene.util.BytesRef;

public class MultiTermDisjunctionQuery extends MultiTermQuery {

private final Set<BytesRef> values;

/\*\* Creates a new MultiTermDisjunctionQuery instance. \*/

public MultiTermDisjunctionQuery(String field, Set<BytesRef> values) {

super(field);

this.values = values;

}

@Override

protected TermsEnum getTermsEnum(Terms terms, AttributeSource atts)

throws IOException {

final TermsEnum termsEnum = terms.iterator();

final Iterator<BytesRef> it = values.iterator();

return new FilteredTermsEnum(termsEnum) {

@Override protected AcceptStatus accept(BytesRef term) throws IOException {

return AcceptStatus.YES;

}

@Override public BytesRef next() throws IOException {

while (it.hasNext()) {

BytesRef termRef = it.next();

if (termsEnum.seekExact(termRef)) {

return termRef;

}

}

return null;

}

};

}

@Override

public String toString(String field) {

StringBuilder builder = new StringBuilder();

builder.append("MultiTermDisjunctionQuery[");

for (BytesRef termVal : this.values) {

builder.append(termVal);

builder.append(",");

}

builder.setLength(builder.length() - 1);

builder.append("]");

return builder.toString();

}

}