package com.twitter.search.earlybird.queryparser;

import java.util.Set;

import com.google.common.collect.ImmutableSet;

import com.twitter.search.queryparser.query.BooleanQuery;

import com.twitter.search.queryparser.query.Conjunction;

import com.twitter.search.queryparser.query.Disjunction;

import com.twitter.search.queryparser.query.Operator;

import com.twitter.search.queryparser.query.Phrase;

import com.twitter.search.queryparser.query.Query;

import com.twitter.search.queryparser.query.QueryParserException;

import com.twitter.search.queryparser.query.QueryVisitor;

import com.twitter.search.queryparser.query.SpecialTerm;

import com.twitter.search.queryparser.query.Term;

import com.twitter.search.queryparser.query.annotation.Annotation;

import com.twitter.search.queryparser.query.annotation.FieldNameWithBoost;

/\*\*

\* Detects whether the query tree has certain field annotations.

\*/

public class DetectFieldAnnotationVisitor extends QueryVisitor<Boolean> {

private final ImmutableSet<String> fieldNames;

/\*\*

\* This visitor will return true if the query tree has a FIELD annotation with any of the given

\* field names. If the set is empty, any FIELD annotation will match.

\*/

public DetectFieldAnnotationVisitor(Set<String> fieldNames) {

this.fieldNames = ImmutableSet.copyOf(fieldNames);

}

/\*\*

\* This visitor will return true if the query tree has a FIELD annotation.

\*/

public DetectFieldAnnotationVisitor() {

this.fieldNames = ImmutableSet.of();

}

@Override

public Boolean visit(Disjunction disjunction) throws QueryParserException {

return visitQuery(disjunction) || visitBooleanQuery(disjunction);

}

@Override

public Boolean visit(Conjunction conjunction) throws QueryParserException {

return visitQuery(conjunction) || visitBooleanQuery(conjunction);

}

@Override

public Boolean visit(Phrase phrase) throws QueryParserException {

return visitQuery(phrase);

}

@Override

public Boolean visit(Term term) throws QueryParserException {

return visitQuery(term);

}

@Override

public Boolean visit(Operator operator) throws QueryParserException {

return visitQuery(operator);

}

@Override

public Boolean visit(SpecialTerm special) throws QueryParserException {

return visitQuery(special);

}

private Boolean visitQuery(Query query) throws QueryParserException {

if (query.hasAnnotations()) {

for (Annotation annotation : query.getAnnotations()) {

if (!Annotation.Type.FIELD.equals(annotation.getType())) {

continue;

}

if (fieldNames.isEmpty()) {

return true;

}

FieldNameWithBoost value = (FieldNameWithBoost) annotation.getValue();

if (fieldNames.contains(value.getFieldName())) {

return true;

}

}

}

return false;

}

private boolean visitBooleanQuery(BooleanQuery query) throws QueryParserException {

for (Query subQuery : query.getChildren()) {

if (subQuery.accept(this)) {

return true;

}

}

return false;

}

}