package com.twitter.search.earlybird.queryparser;

import java.util.Map;

import com.google.common.annotations.VisibleForTesting;

import org.apache.lucene.search.Query;

import com.twitter.decider.Decider;

import com.twitter.search.common.query.MappableField;

import com.twitter.search.common.schema.base.FieldWeightDefault;

import com.twitter.search.common.schema.base.ImmutableSchemaInterface;

import com.twitter.search.common.schema.earlybird.EarlybirdCluster;

import com.twitter.search.common.search.TerminationTracker;

import com.twitter.search.common.search.termination.QueryTimeout;

import com.twitter.search.earlybird.common.userupdates.UserScrubGeoMap;

import com.twitter.search.earlybird.common.userupdates.UserTable;

import com.twitter.search.earlybird.partition.MultiSegmentTermDictionaryManager;

import com.twitter.search.earlybird.querycache.QueryCacheManager;

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

public class LuceneRelevanceQueryVisitor extends EarlybirdLuceneQueryVisitor {

public LuceneRelevanceQueryVisitor(

ImmutableSchemaInterface schema,

QueryCacheManager queryCacheManager,

UserTable userTable,

UserScrubGeoMap userScrubGeoMap,

TerminationTracker terminationTracker,

Map<String, FieldWeightDefault> fieldWeightMap,

Map<MappableField, String> mappableFieldMap,

MultiSegmentTermDictionaryManager multiSegmentTermDictionaryManager,

Decider decider,

EarlybirdCluster earlybirdCluster,

QueryTimeout queryTimeout) {

super(

schema,

queryCacheManager,

userTable,

userScrubGeoMap,

terminationTracker,

fieldWeightMap,

mappableFieldMap,

multiSegmentTermDictionaryManager,

decider,

earlybirdCluster,

queryTimeout);

}

@VisibleForTesting

protected LuceneRelevanceQueryVisitor(

ImmutableSchemaInterface schema,

QueryCacheManager queryCacheManager,

UserTable userTable,

UserScrubGeoMap userScrubGeoMap,

EarlybirdCluster earlybirdCluster) {

super(schema,

queryCacheManager,

userTable,

userScrubGeoMap,

earlybirdCluster,

queryCacheManager.getDecider());

}

@Override

protected Query visitSinceIDOperator(SearchOperator op) {

// since\_id is handled by the blender for relevance queries, so don't filter on it.

return null;

}

}