package com.twitter.search.earlybird.search.queries;

import java.io.IOException;

import java.util.Set;

import com.google.common.base.Preconditions;

import org.apache.lucene.index.LeafReaderContext;

import org.apache.lucene.index.Term;

import org.apache.lucene.search.ConstantScoreScorer;

import org.apache.lucene.search.Explanation;

import org.apache.lucene.search.IndexSearcher;

import org.apache.lucene.search.Query;

import org.apache.lucene.search.Scorer;

import org.apache.lucene.search.ScoreMode;

import org.apache.lucene.search.Weight;

import com.twitter.search.core.earlybird.index.EarlybirdIndexSegmentAtomicReader;

import com.twitter.search.core.earlybird.index.util.RangeFilterDISI;

import com.twitter.search.earlybird.index.EarlybirdSingleSegmentSearcher;

/\*\*

\* A MatchAllDocsQuery implementation that does not assume that doc IDs are assigned sequentially.

\* Instead, it wraps the EarlybirdIndexSegmentAtomicReader into a RangeFilterDISI, and uses

\* this iterator to traverse only the valid doc IDs in this segment.

\*

\* Note that org.apache.lucene.index.MatchAllDocsQuery is final, so we cannot extend it.

\*/

public class MatchAllDocsQuery extends Query {

private static class MatchAllDocsWeight extends Weight {

private final Weight luceneWeight;

public MatchAllDocsWeight(Query query, Weight luceneWeight) {

super(query);

this.luceneWeight = luceneWeight;

}

@Override

public void extractTerms(Set<Term> terms) {

luceneWeight.extractTerms(terms);

}

@Override

public Explanation explain(LeafReaderContext context, int doc) throws IOException {

return luceneWeight.explain(context, doc);

}

@Override

public Scorer scorer(LeafReaderContext context) throws IOException {

Preconditions.checkState(context.reader() instanceof EarlybirdIndexSegmentAtomicReader,

"Expected an EarlybirdIndexSegmentAtomicReader, but got a "

+ context.reader().getClass().getName() + " instance.");

EarlybirdIndexSegmentAtomicReader reader =

(EarlybirdIndexSegmentAtomicReader) context.reader();

return new ConstantScoreScorer(

this, 1.0f, ScoreMode.COMPLETE\_NO\_SCORES, new RangeFilterDISI(reader));

}

@Override

public boolean isCacheable(LeafReaderContext ctx) {

return luceneWeight.isCacheable(ctx);

}

}

@Override

public Weight createWeight(IndexSearcher searcher, ScoreMode scoreMode, float boost) {

org.apache.lucene.search.MatchAllDocsQuery luceneMatchAllDocsQuery =

new org.apache.lucene.search.MatchAllDocsQuery();

Weight luceneWeight = luceneMatchAllDocsQuery.createWeight(searcher, scoreMode, boost);

if (!(searcher instanceof EarlybirdSingleSegmentSearcher)) {

return luceneWeight;

}

return new MatchAllDocsWeight(this, luceneWeight);

}

@Override

public int hashCode() {

return 0;

}

@Override

public boolean equals(Object obj) {

return obj instanceof MatchAllDocsQuery;

}

// Copied from org.apache.lucene.search.MatchAllDocsWeight

@Override

public String toString(String field) {

return "\*:\*";

}

}