package com.twitter.search.earlybird.search;

import java.io.IOException;

import com.google.common.base.Preconditions;

import com.google.common.primitives.Longs;

import org.apache.lucene.index.NumericDocValues;

import com.twitter.common\_internal.bloomfilter.BloomFilter;

import com.twitter.search.common.schema.earlybird.EarlybirdFieldConstants.EarlybirdFieldConstant;

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

import com.twitter.search.earlybird.thrift.ThriftSocialFilterType;

/\*\*

\* Filter class used by the SearchResultsCollector to filter social tweets

\* from the hits.

\*/

public class SocialFilter {

private interface Acceptor {

boolean accept(long fromUserLong, byte[] userIDInBytes);

}

private NumericDocValues fromUserID;

private final Acceptor acceptor;

private final long searcherId;

private final BloomFilter trustedFilter;

private final BloomFilter followFilter;

private class FollowsAcceptor implements Acceptor {

@Override

public boolean accept(long fromUserLong, byte[] userIdInBytes) {

return followFilter.contains(userIdInBytes);

}

}

private class TrustedAcceptor implements Acceptor {

@Override

public boolean accept(long fromUserLong, byte[] userIdInBytes) {

return trustedFilter.contains(userIdInBytes);

}

}

private class AllAcceptor implements Acceptor {

@Override

public boolean accept(long fromUserLong, byte[] userIdInBytes) {

return trustedFilter.contains(userIdInBytes)

|| followFilter.contains(userIdInBytes)

|| fromUserLong == searcherId;

}

}

public SocialFilter(

ThriftSocialFilterType socialFilterType,

final long searcherId,

final byte[] trustedFilter,

final byte[] followFilter) throws IOException {

Preconditions.checkNotNull(socialFilterType);

Preconditions.checkNotNull(trustedFilter);

Preconditions.checkNotNull(followFilter);

this.searcherId = searcherId;

this.trustedFilter = new BloomFilter(trustedFilter);

this.followFilter = new BloomFilter(followFilter);

switch (socialFilterType) {

case FOLLOWS:

this.acceptor = new FollowsAcceptor();

break;

case TRUSTED:

this.acceptor = new TrustedAcceptor();

break;

case ALL:

this.acceptor = new AllAcceptor();

break;

default:

throw new UnsupportedOperationException("Invalid social filter type passed");

}

}

public void startSegment(EarlybirdIndexSegmentAtomicReader indexReader) throws IOException {

fromUserID =

indexReader.getNumericDocValues(EarlybirdFieldConstant.FROM\_USER\_ID\_CSF.getFieldName());

}

/\*\*

\* Determines if the given doc ID should be accepted.

\*/

public boolean accept(int internalDocID) throws IOException {

if (!fromUserID.advanceExact(internalDocID)) {

return false;

}

long fromUserLong = fromUserID.longValue();

byte[] userIDInBytes = Longs.toByteArray(fromUserLong);

return acceptor.accept(fromUserLong, userIDInBytes);

}

}