package com.twitter.home\_mixer.product.following

import com.twitter.finagle.thrift.ClientId

import com.twitter.finagle.tracing.Trace

import com.twitter.home\_mixer.model.HomeFeatures.RealGraphInNetworkScoresFeature

import com.twitter.home\_mixer.product.following.model.FollowingQuery

import com.twitter.home\_mixer.product.following.param.FollowingParam.ServerMaxResultsParam

import com.twitter.product\_mixer.component\_library.feature\_hydrator.query.social\_graph.SGSFollowedUsersFeature

import com.twitter.product\_mixer.core.functional\_component.transformer.CandidatePipelineQueryTransformer

import com.twitter.product\_mixer.core.model.marshalling.response.urt.operation.BottomCursor

import com.twitter.product\_mixer.core.model.marshalling.response.urt.operation.GapCursor

import com.twitter.product\_mixer.core.model.marshalling.response.urt.operation.TopCursor

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.MalformedCursor

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.PipelineFailure

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

import com.twitter.search.earlybird.{thriftscala => t}

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

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

import javax.inject.Inject

import javax.inject.Singleton

import scala.jdk.CollectionConverters.asJavaIterableConverter

@Singleton

case class FollowingEarlybirdQueryTransformer @Inject() (clientId: ClientId)

extends CandidatePipelineQueryTransformer[FollowingQuery, t.EarlybirdRequest] {

override def transform(query: FollowingQuery): t.EarlybirdRequest = {

val followedUserIds =

query.features.map(\_.get(SGSFollowedUsersFeature)).getOrElse(Seq.empty).toSet

val realGraphInNetworkFollowedUserIds =

query.features.map(\_.get(RealGraphInNetworkScoresFeature)).getOrElse(Map.empty).keySet

val userId = query.getRequiredUserId

val combinedUserIds = userId +: followedUserIds.toSeq

val baseFollowedUsersSearchOperator = new SearchOperator.Builder()

.setType(SearchOperator.Type.FEATURE\_VALUE\_IN\_ACCEPT\_LIST\_OR\_UNSET)

.addOperand(EarlybirdFieldConstant.DIRECTED\_AT\_USER\_ID\_CSF.getFieldName)

val followedUsersQuery =

baseFollowedUsersSearchOperator.addOperands(combinedUserIds.map(\_.toString).asJava).build()

val searchQuery = query.pipelineCursor

.map { cursor =>

val sinceIdQuery =

(id: Long) => new SearchOperator(SearchOperator.Type.SINCE\_ID, id.toString)

val maxIdQuery = // max ID is inclusive, so subtract 1

(id: Long) => new SearchOperator(SearchOperator.Type.MAX\_ID, (id - 1).toString)

(cursor.cursorType, cursor.id, cursor.gapBoundaryId) match {

case (Some(TopCursor), Some(sinceId), \_) =>

new Conjunction(sinceIdQuery(sinceId), followedUsersQuery)

case (Some(BottomCursor), Some(maxId), \_) =>

new Conjunction(maxIdQuery(maxId), followedUsersQuery)

case (Some(GapCursor), Some(maxId), Some(sinceId)) =>

new Conjunction(sinceIdQuery(sinceId), maxIdQuery(maxId), followedUsersQuery)

case (Some(GapCursor), \_, \_) =>

throw PipelineFailure(MalformedCursor, "Invalid cursor " + cursor.toString)

case \_ => followedUsersQuery

}

}.getOrElse(followedUsersQuery)

val metadataOptions = t.ThriftSearchResultMetadataOptions(

getInReplyToStatusId = true,

getReferencedTweetAuthorId = true,

getFromUserId = true

)

t.EarlybirdRequest(

searchQuery = t.ThriftSearchQuery(

serializedQuery = Some(searchQuery.serialize),

fromUserIDFilter64 = Some(combinedUserIds),

numResults = query.requestedMaxResults.getOrElse(query.params(ServerMaxResultsParam)),

rankingMode = t.ThriftSearchRankingMode.Recency,

resultMetadataOptions = Some(metadataOptions),

searcherId = query.getOptionalUserId,

),

getOlderResults = Some(true), // needed for archive access to older tweets

clientRequestID = Some(s"${Trace.id.traceId}"),

followedUserIds = Some(combinedUserIds),

numResultsToReturnAtRoot = Some(query.params(ServerMaxResultsParam)),

clientId = Some(clientId.name),

)

}

}