package com.twitter.recosinjector.clients

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.logging.Logger

import com.twitter.socialgraph.thriftscala.\_

import com.twitter.storehaus.ReadableStore

import com.twitter.util.Future

class SocialGraph(

socialGraphIdStore: ReadableStore[IdsRequest, IdsResult]

)(

implicit statsReceiver: StatsReceiver) {

import SocialGraph.\_

private val log = Logger()

private val followedByNotMutedByStats = statsReceiver.scope("followedByNotMutedBy")

private def fetchIdsFromSocialGraph(

userId: Long,

ids: Seq[Long],

relationshipTypes: Map[RelationshipType, Boolean],

lookupContext: Option[LookupContext] = IncludeInactiveUnionLookupContext,

stats: StatsReceiver

): Future[Seq[Long]] = {

if (ids.isEmpty) {

stats.counter("fetchIdsEmpty").incr()

Future.Nil

} else {

val relationships = relationshipTypes.map {

case (relationshipType, hasRelationship) =>

SrcRelationship(

source = userId,

relationshipType = relationshipType,

hasRelationship = hasRelationship,

targets = Some(ids)

)

}.toSeq

val idsRequest = IdsRequest(

relationships = relationships,

pageRequest = SelectAllPageRequest,

context = lookupContext

)

socialGraphIdStore

.get(idsRequest)

.map { \_.map(\_.ids).getOrElse(Nil) }

.rescue {

case e =>

stats.scope("fetchIdsFailure").counter(e.getClass.getSimpleName).incr()

log.error(s"Failed with message ${e.toString}")

Future.Nil

}

}

}

// which of the users in candidates follow userId and have not muted userId

def followedByNotMutedBy(userId: Long, candidates: Seq[Long]): Future[Seq[Long]] = {

fetchIdsFromSocialGraph(

userId,

candidates,

FollowedByNotMutedRelationships,

IncludeInactiveLookupContext,

followedByNotMutedByStats

)

}

}

object SocialGraph {

val SelectAllPageRequest = Some(PageRequest(selectAll = Some(true)))

val IncludeInactiveLookupContext = Some(LookupContext(includeInactive = true))

val IncludeInactiveUnionLookupContext = Some(

LookupContext(includeInactive = true, performUnion = Some(true))

)

val FollowedByNotMutedRelationships: Map[RelationshipType, Boolean] = Map(

RelationshipType.FollowedBy -> true,

RelationshipType.MutedBy -> false

)

}