package com.twitter.recos.user\_tweet\_graph.store

import com.twitter.simclusters\_v2.common.UserId

import com.twitter.socialgraph.thriftscala.EdgesRequest

import com.twitter.socialgraph.thriftscala.EdgesResult

import com.twitter.socialgraph.thriftscala.PageRequest

import com.twitter.socialgraph.thriftscala.RelationshipType

import com.twitter.socialgraph.thriftscala.SrcRelationship

import com.twitter.socialgraph.thriftscala.SocialGraphService

import com.twitter.storehaus.ReadableStore

import com.twitter.util.Duration

import com.twitter.util.Future

import com.twitter.util.Time

class UserRecentFollowersStore(

sgsClient: SocialGraphService.MethodPerEndpoint)

extends ReadableStore[UserRecentFollowersStore.Query, Seq[UserId]] {

override def get(key: UserRecentFollowersStore.Query): Future[Option[Seq[UserId]]] = {

val edgeRequest = EdgesRequest(

relationship = SrcRelationship(key.userId, RelationshipType.FollowedBy),

// Could have a better guess at count when k.maxAge != None

pageRequest = Some(PageRequest(count = key.maxResults))

)

val lookbackThresholdMillis = key.maxAge

.map(maxAge => (Time.now - maxAge).inMilliseconds)

.getOrElse(0L)

sgsClient

.edges(Seq(edgeRequest))

.map(\_.flatMap {

case EdgesResult(edges, \_, \_) =>

edges.collect {

case e if e.createdAt >= lookbackThresholdMillis =>

e.target

}

})

.map(Some(\_))

}

}

object UserRecentFollowersStore {

case class Query(

userId: UserId,

// maxResults - if Some(count), we return only the `count` most recent follows

maxResults: Option[Int] = None,

// maxAge - if Some(duration), return only follows since `Time.now - duration`

maxAge: Option[Duration] = None)

}