package com.twitter.recosinjector.edges

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.recos.util.Action

import com.twitter.socialgraph.thriftscala.{

Action => SocialGraphAction,

FollowGraphEvent,

FollowType,

WriteEvent

}

import com.twitter.util.Future

/\*\*

\* Converts a WriteEvent to UserUserGraph's messages, including Mention and Mediatag messages

\*/

class SocialWriteEventToUserUserGraphBuilder()(override implicit val statsReceiver: StatsReceiver)

extends EventToMessageBuilder[WriteEvent, UserUserEdge] {

private val followOrFrictionlessFollowCounter =

statsReceiver.counter("num\_follow\_or\_frictionless")

private val notFollowOrFrictionlessFollowCounter =

statsReceiver.counter("num\_not\_follow\_or\_frictionless")

private val followEdgeCounter = statsReceiver.counter("num\_follow\_edge")

/\*\*

\* For now, we are only interested in Follow events

\*/

override def shouldProcessEvent(event: WriteEvent): Future[Boolean] = {

event.action match {

case SocialGraphAction.Follow | SocialGraphAction.FrictionlessFollow =>

followOrFrictionlessFollowCounter.incr()

Future(true)

case \_ =>

notFollowOrFrictionlessFollowCounter.incr()

Future(false)

}

}

/\*\*

\* Determine whether a Follow event is valid/error free.

\*/

private def isValidFollowEvent(followEvent: FollowGraphEvent): Boolean = {

followEvent.followType match {

case Some(FollowType.NormalFollow) | Some(FollowType.FrictionlessFollow) =>

followEvent.result.validationError.isEmpty

case \_ =>

false

}

}

override def buildEdges(event: WriteEvent): Future[Seq[UserUserEdge]] = {

val userUserEdges = event.follow

.map(\_.collect {

case followEvent if isValidFollowEvent(followEvent) =>

val sourceUserId = followEvent.result.request.source

val targetUserId = followEvent.result.request.target

followEdgeCounter.incr()

UserUserEdge(

sourceUserId,

targetUserId,

Action.Follow,

Some(System.currentTimeMillis())

)

}).getOrElse(Nil)

Future(userUserEdges)

}

override def filterEdges(

event: WriteEvent,

edges: Seq[UserUserEdge]

): Future[Seq[UserUserEdge]] = {

Future(edges)

}

}