package com.twitter.recosinjector.edges

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.recos.util.Action

import com.twitter.recosinjector.util.TweetFavoriteEventDetails

import com.twitter.util.Future

class TimelineEventToUserTweetGraphBuilder(

userTweetEntityEdgeBuilder: UserTweetEntityEdgeBuilder

)(

override implicit val statsReceiver: StatsReceiver)

extends EventToMessageBuilder[TweetFavoriteEventDetails, UserTweetEntityEdge] {

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

Future(true)

}

override def buildEdges(details: TweetFavoriteEventDetails): Future[Seq[UserTweetEntityEdge]] = {

val engagement = details.userTweetEngagement

engagement.action match {

case Action.Favorite =>

val tweetDetails = engagement.tweetDetails

val entitiesMapFut = userTweetEntityEdgeBuilder.getEntitiesMapAndUpdateCache(

tweetId = engagement.tweetId,

tweetDetails = tweetDetails

)

entitiesMapFut

.map { entitiesMap =>

UserTweetEntityEdge(

sourceUser = engagement.engageUserId,

targetTweet = engagement.tweetId,

action = engagement.action,

metadata = engagement.engagementTimeMillis,

cardInfo = engagement.tweetDetails.map(\_.cardInfo.toByte),

entitiesMap = entitiesMap,

tweetDetails = tweetDetails

)

}

.map(Seq(\_))

case \_ => Future.Nil

}

}

override def filterEdges(

event: TweetFavoriteEventDetails,

edges: Seq[UserTweetEntityEdge]

): Future[Seq[UserTweetEntityEdge]] = {

Future(edges)

}

}