package com.twitter.timelineranker.core

import com.twitter.timelines.model.UserId

import com.twitter.util.Future

/\*\*

\* Similar to FollowGraphData but makes available its fields as separate futures.

\*/

case class FollowGraphDataFuture(

userId: UserId,

followedUserIdsFuture: Future[Seq[UserId]],

mutuallyFollowingUserIdsFuture: Future[Set[UserId]],

mutedUserIdsFuture: Future[Set[UserId]],

retweetsMutedUserIdsFuture: Future[Set[UserId]]) {

def inNetworkUserIdsFuture: Future[Seq[UserId]] = followedUserIdsFuture.map(\_ :+ userId)

def get(): Future[FollowGraphData] = {

Future

.join(

followedUserIdsFuture,

mutuallyFollowingUserIdsFuture,

mutedUserIdsFuture,

retweetsMutedUserIdsFuture

)

.map {

case (followedUserIds, mutuallyFollowingUserIds, mutedUserIds, retweetsMutedUserIds) =>

FollowGraphData(

userId,

followedUserIds,

mutuallyFollowingUserIds,

mutedUserIds,

retweetsMutedUserIds

)

}

}

}

object FollowGraphDataFuture {

private def mkEmptyFuture(field: String) = {

Future.exception(

new IllegalStateException(s"FollowGraphDataFuture field $field accessed without being set")

)

}

val EmptyFollowGraphDataFuture: FollowGraphDataFuture = FollowGraphDataFuture(

userId = 0L,

followedUserIdsFuture = mkEmptyFuture("followedUserIdsFuture"),

mutuallyFollowingUserIdsFuture = mkEmptyFuture("mutuallyFollowingUserIdsFuture"),

mutedUserIdsFuture = mkEmptyFuture("mutedUserIdsFuture"),

retweetsMutedUserIdsFuture = mkEmptyFuture("retweetsMutedUserIdsFuture")

)

}