package com.twitter.tweetypie

package repository

import com.twitter.flockdb.client.\_

import com.twitter.stitch.SeqGroup

import com.twitter.stitch.Stitch

import com.twitter.stitch.compat.LegacySeqGroup

sealed trait TweetCountKey {

// The flockdb Select used to calculate the count from TFlock

def toSelect: Select[StatusGraph]

// The Tweet id for this count

def tweetId: TweetId

// com.twitter.servo.cache.MemcacheCache calls toString to turn this key into a cache key

def toString: String

}

case class RetweetsKey(tweetId: TweetId) extends TweetCountKey {

lazy val toSelect: Select[StatusGraph] = RetweetsGraph.from(tweetId)

override lazy val toString: String = "cnts:rt:" + tweetId

}

case class RepliesKey(tweetId: TweetId) extends TweetCountKey {

lazy val toSelect: Select[StatusGraph] = RepliesToTweetsGraph.from(tweetId)

override lazy val toString: String = "cnts:re:" + tweetId

}

case class FavsKey(tweetId: TweetId) extends TweetCountKey {

lazy val toSelect: Select[StatusGraph] = FavoritesGraph.to(tweetId)

override lazy val toString: String = "cnts:fv:" + tweetId

}

case class QuotesKey(tweetId: TweetId) extends TweetCountKey {

lazy val toSelect: Select[StatusGraph] = QuotersGraph.from(tweetId)

override lazy val toString: String = "cnts:qt:" + tweetId

}

case class BookmarksKey(tweetId: TweetId) extends TweetCountKey {

lazy val toSelect: Select[StatusGraph] = BookmarksGraph.to(tweetId)

override lazy val toString: String = "cnts:bm:" + tweetId

}

object TweetCountsRepository {

type Type = TweetCountKey => Stitch[Count]

def apply(tflock: TFlockClient, maxRequestSize: Int): Type = {

object RequestGroup extends SeqGroup[TweetCountKey, Count] {

override def run(keys: Seq[TweetCountKey]): Future[Seq[Try[MediaId]]] = {

val selects = MultiSelect[StatusGraph]() ++= keys.map(\_.toSelect)

LegacySeqGroup.liftToSeqTry(tflock.multiCount(selects).map(counts => counts.map(\_.toLong)))

}

override val maxSize: Int = maxRequestSize

}

key => Stitch.call(key, RequestGroup)

}

}