package com.twitter.usersignalservice.signals

import com.twitter.bijection.Codec

import com.twitter.bijection.scrooge.BinaryScalaCodec

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.onboarding.relevance.tweet\_engagement.thriftscala.EngagementIdentifier

import com.twitter.onboarding.relevance.tweet\_engagement.thriftscala.TweetEngagement

import com.twitter.onboarding.relevance.tweet\_engagement.thriftscala.TweetEngagements

import com.twitter.scalding\_internal.multiformat.format.keyval.KeyValInjection.Long2BigEndian

import com.twitter.simclusters\_v2.thriftscala.InternalId

import com.twitter.storage.client.manhattan.kv.ManhattanKVClientMtlsParams

import com.twitter.storehaus\_internal.manhattan.Apollo

import com.twitter.storehaus\_internal.manhattan.ManhattanCluster

import com.twitter.twistly.common.UserId

import com.twitter.usersignalservice.base.ManhattanSignalFetcher

import com.twitter.usersignalservice.base.Query

import com.twitter.usersignalservice.thriftscala.Signal

import com.twitter.usersignalservice.thriftscala.SignalType

import com.twitter.util.Future

import com.twitter.util.Timer

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

case class TweetSharesFetcher @Inject() (

manhattanKVClientMtlsParams: ManhattanKVClientMtlsParams,

timer: Timer,

stats: StatsReceiver)

extends ManhattanSignalFetcher[Long, TweetEngagements] {

import TweetSharesFetcher.\_

override type RawSignalType = TweetEngagement

override def name: String = this.getClass.getCanonicalName

override def statsReceiver: StatsReceiver = stats.scope(name)

override protected def manhattanAppId: String = MHAppId

override protected def manhattanDatasetName: String = MHDatasetName

override protected def manhattanClusterId: ManhattanCluster = Apollo

override protected def manhattanKeyCodec: Codec[Long] = Long2BigEndian

override protected def manhattanRawSignalCodec: Codec[TweetEngagements] = BinaryScalaCodec(

TweetEngagements)

override protected def toManhattanKey(userId: UserId): Long = userId

override protected def toRawSignals(

manhattanValue: TweetEngagements

): Seq[TweetEngagement] = manhattanValue.tweetEngagements

override def process(

query: Query,

rawSignals: Future[Option[Seq[TweetEngagement]]]

): Future[Option[Seq[Signal]]] = {

rawSignals.map {

\_.map {

\_.collect {

case tweetEngagement if (tweetEngagement.engagementType == EngagementIdentifier.Share) =>

Signal(

SignalType.TweetShareV1,

tweetEngagement.timestampMs,

Some(InternalId.TweetId(tweetEngagement.tweetId)))

}.sortBy(-\_.timestamp).take(query.maxResults.getOrElse(Int.MaxValue))

}

}

}

}

object TweetSharesFetcher {

private val MHAppId = "uss\_prod\_apollo"

private val MHDatasetName = "tweet\_share\_engagements"

}