package com.twitter.tweetypie

package hydrator

import com.twitter.conversions.DurationOps.\_

import com.twitter.snowflake.id.SnowflakeId

object CreatedAtRepairer {

// no createdAt value should be less than this

val jan\_01\_2006 = 1136073600000L

// no non-snowflake createdAt value should be greater than this

val jan\_01\_2011 = 1293840000000L

// allow createdAt timestamp to be up to this amount off from the snowflake id

// before applying the correction.

val varianceThreshold: MediaId = 10.minutes.inMilliseconds

}

/\*\*

\* Detects tweets with bad createdAt timestamps and attempts to fix, if possible

\* using the snowflake id. pre-snowflake tweets are left unmodified.

\*/

class CreatedAtRepairer(scribe: FutureEffect[String]) extends Mutation[Tweet] {

import CreatedAtRepairer.\_

def apply(tweet: Tweet): Option[Tweet] = {

assert(tweet.coreData.nonEmpty, "tweet core data is missing")

val createdAtMillis = getCreatedAt(tweet) \* 1000

if (SnowflakeId.isSnowflakeId(tweet.id)) {

val snowflakeMillis = SnowflakeId(tweet.id).unixTimeMillis.asLong

val diff = (snowflakeMillis - createdAtMillis).abs

if (diff >= varianceThreshold) {

scribe(tweet.id + "\t" + createdAtMillis)

val snowflakeSeconds = snowflakeMillis / 1000

Some(TweetLenses.createdAt.set(tweet, snowflakeSeconds))

} else {

None

}

} else {

// not a snowflake id, hard to repair, so just log it

if (createdAtMillis < jan\_01\_2006 || createdAtMillis > jan\_01\_2011) {

scribe(tweet.id + "\t" + createdAtMillis)

}

None

}

}

}