package com.twitter.tweetypie.storage

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.stitch.Stitch

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

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

import com.twitter.tweetypie.storage.TweetUtils.\_

import com.twitter.tweetypie.thriftscala.Tweet

import com.twitter.util.Throw

import com.twitter.util.Time

object UpdateTweetHandler {

def apply(

insert: ManhattanOperations.Insert,

stats: StatsReceiver

): TweetStorageClient.UpdateTweet = { (tpTweet: Tweet, fields: Seq[Field]) =>

require(

fields.forall(!TweetFields.coreFieldIds.contains(\_)),

"Core fields cannot be modified by calling updateTweet; use addTweet instead."

)

require(

areAllFieldsDefined(tpTweet, fields),

s"Input tweet $tpTweet does not have specified fields $fields set"

)

val now = Time.now

val storedTweet = StorageConversions.toStoredTweetForFields(tpTweet, fields.toSet)

val tweetId = storedTweet.id

Stats.updatePerFieldQpsCounters("updateTweet", fields.map(\_.id), 1, stats)

val (fieldIds, stitchesPerTweet) =

fields.map { field =>

val fieldId = field.id

val tweetKey = TweetKey.fieldKey(tweetId, fieldId)

val blob = storedTweet.getFieldBlob(fieldId).get

val value = ManhattanValue(TFieldBlobCodec.toByteBuffer(blob), Some(now))

val record = TweetManhattanRecord(tweetKey, value)

(fieldId, insert(record).liftToTry)

}.unzip

Stitch.collect(stitchesPerTweet).map { seqOfTries =>

val fieldkeyAndMhResults = fieldIds.zip(seqOfTries).toMap

// If even a single field was rate limited, we will send an overall OverCapacity TweetResponse

val wasRateLimited = fieldkeyAndMhResults.exists { keyAndResult =>

keyAndResult.\_2 match {

case Throw(e: DeniedManhattanException) => true

case \_ => false

}

}

if (wasRateLimited) {

buildTweetOverCapacityResponse("updateTweets", tweetId, fieldkeyAndMhResults)

} else {

buildTweetResponse("updateTweets", tweetId, fieldkeyAndMhResults)

}

}

}

private def areAllFieldsDefined(tpTweet: Tweet, fields: Seq[Field]) = {

val storedTweet = StorageConversions.toStoredTweetForFields(tpTweet, fields.toSet)

fields.map(\_.id).forall(storedTweet.getFieldBlob(\_).isDefined)

}

}