package com.twitter.tweetypie.storage

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.stitch.Stitch

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

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

import com.twitter.util.Time

/\*\*

\* Deletes data for the scrubbed field and writes a metadata record.

\* Provides scrub functionality. Right now, we only allow the scrubbing of the geo field.

\* It should be simple to add more fields to the allowlist if needed.

\*/

object ScrubHandler {

val scrubFieldsAllowlist: Set[Field] = Set(Field.Geo)

def apply(

insert: ManhattanOperations.Insert,

delete: ManhattanOperations.Delete,

scribe: Scribe,

stats: StatsReceiver

): TweetStorageClient.Scrub =

(unfilteredTweetIds: Seq[TweetId], columns: Seq[Field]) => {

val tweetIds = unfilteredTweetIds.filter(\_ > 0)

require(columns.nonEmpty, "Must specify fields to scrub")

require(

columns.toSet.size == columns.size,

s"Duplicate fields to scrub specified: $columns"

)

require(

columns.forall(scrubFieldsAllowlist.contains(\_)),

s"Cannot scrub $columns; scrubbable fields are restricted to $scrubFieldsAllowlist"

)

Stats.addWidthStat("scrub", "ids", tweetIds.size, stats)

val mhTimestamp = Time.now

val stitches = tweetIds.map { tweetId =>

val deletionStitches = columns.map { field =>

val mhKeyToDelete = TweetKey.fieldKey(tweetId, field.id)

delete(mhKeyToDelete, Some(mhTimestamp)).liftToTry

}

val collectedStitch =

Stitch.collect(deletionStitches).map(collectWithRateLimitCheck).lowerFromTry

collectedStitch

.flatMap { \_ =>

val scrubbedStitches = columns.map { column =>

val scrubbedKey = TweetKey.scrubbedFieldKey(tweetId, column.id)

val record =

TweetManhattanRecord(

scrubbedKey,

ManhattanValue(StringCodec.toByteBuffer(""), Some(mhTimestamp))

)

insert(record).liftToTry

}

Stitch.collect(scrubbedStitches)

}

.map(collectWithRateLimitCheck)

}

Stitch.collect(stitches).map(collectWithRateLimitCheck).lowerFromTry.onSuccess { \_ =>

tweetIds.foreach { id => scribe.logScrubbed(id, columns.map(\_.id.toInt), mhTimestamp) }

}

}

}