package com.twitter.tsp.columns

import com.twitter.stitch.SeqGroup

import com.twitter.stitch.Stitch

import com.twitter.strato.catalog.Fetch

import com.twitter.strato.catalog.OpMetadata

import com.twitter.strato.config.\_

import com.twitter.strato.config.AllowAll

import com.twitter.strato.config.ContactInfo

import com.twitter.strato.config.Policy

import com.twitter.strato.data.Conv

import com.twitter.strato.data.Description.PlainText

import com.twitter.strato.data.Lifecycle.Production

import com.twitter.strato.fed.StratoFed

import com.twitter.strato.thrift.ScroogeConv

import com.twitter.tsp.thriftscala.TopicSocialProofRequest

import com.twitter.tsp.thriftscala.TopicSocialProofOptions

import com.twitter.tsp.service.TopicSocialProofService

import com.twitter.tsp.thriftscala.TopicWithScore

import com.twitter.util.Future

import com.twitter.util.Try

import javax.inject.Inject

class TopicSocialProofBatchColumn @Inject() (

topicSocialProofService: TopicSocialProofService)

extends StratoFed.Column(TopicSocialProofBatchColumn.Path)

with StratoFed.Fetch.Stitch {

override val policy: Policy =

ReadWritePolicy(

readPolicy = AllowAll,

writePolicy = AllowKeyAuthenticatedTwitterUserId

)

override type Key = Long

override type View = TopicSocialProofOptions

override type Value = Seq[TopicWithScore]

override val keyConv: Conv[Key] = Conv.ofType

override val viewConv: Conv[View] = ScroogeConv.fromStruct[TopicSocialProofOptions]

override val valueConv: Conv[Value] = Conv.seq(ScroogeConv.fromStruct[TopicWithScore])

override val metadata: OpMetadata =

OpMetadata(

lifecycle = Some(Production),

Some(PlainText("Topic Social Proof Batched Federated Column")))

case class TspsGroup(view: View) extends SeqGroup[Long, Fetch.Result[Value]] {

override protected def run(keys: Seq[Long]): Future[Seq[Try[Result[Seq[TopicWithScore]]]]] = {

val request = TopicSocialProofRequest(

userId = view.userId,

tweetIds = keys.toSet,

displayLocation = view.displayLocation,

topicListingSetting = view.topicListingSetting,

context = view.context,

bypassModes = view.bypassModes,

tags = view.tags

)

val response = topicSocialProofService

.topicSocialProofHandlerStoreStitch(request)

.map(\_.socialProofs)

Stitch

.run(response).map(r =>

keys.map(key => {

Try {

val v = r.get(key)

if (v.nonEmpty && v.get.nonEmpty) {

found(v.get)

} else {

missing

}

}

}))

}

}

override def fetch(key: Key, view: View): Stitch[Result[Value]] = {

Stitch.call(key, TspsGroup(view))

}

}

object TopicSocialProofBatchColumn {

val Path = "topic-signals/tsp/topic-social-proof-batched"

}