package com.twitter.frigate.pushservice.store

import com.twitter.gizmoduck.thriftscala.User

import com.twitter.gizmoduck.thriftscala.UserType

import com.twitter.stitch.Stitch

import com.twitter.storehaus.ReadableStore

import com.twitter.strato.client.Client

import com.twitter.strato.client.UserId

import com.twitter.strato.config.FlockCursors.BySource.Begin

import com.twitter.strato.config.FlockCursors.Continue

import com.twitter.strato.config.FlockCursors.End

import com.twitter.strato.config.FlockPage

import com.twitter.strato.generated.client.socialgraph.service.soft\_users.softUserFollows.EdgeBySourceClientColumn

import com.twitter.util.Future

object SoftUserFollowingStore {

type ViewerFollowingCursor = EdgeBySourceClientColumn.Cursor

val MaxPagesToFetch = 2

val PageLimit = 50

}

class SoftUserFollowingStore(stratoClient: Client) extends ReadableStore[User, Seq[Long]] {

import SoftUserFollowingStore.\_

private val softUserFollowingEdgesPaginator = new EdgeBySourceClientColumn(stratoClient).paginator

private def accumulateIds(cursor: ViewerFollowingCursor, pagesToFetch: Int): Stitch[Seq[Long]] =

softUserFollowingEdgesPaginator.paginate(cursor).flatMap {

case FlockPage(data, next, \_) =>

next match {

case cont: Continue if pagesToFetch > 1 =>

Stitch

.join(

Stitch.value(data.map(\_.to).map(\_.value)),

accumulateIds(cont, pagesToFetch - 1))

.map {

case (a, b) => a ++ b

}

case \_: End | \_: Continue =>

// end pagination if last page has been fetched or [[MaxPagesToFetch]] have been fetched

Stitch.value(data.map(\_.to).map(\_.value))

}

}

private def softFollowingFromStrato(

sourceId: Long,

pageLimit: Int,

pagesToFetch: Int

): Stitch[Seq[Long]] = {

val begin = Begin[UserId, UserId](UserId(sourceId), pageLimit)

accumulateIds(begin, pagesToFetch)

}

override def get(user: User): Future[Option[Seq[Long]]] = {

user.userType match {

case UserType.Soft =>

Stitch.run(softFollowingFromStrato(user.id, PageLimit, MaxPagesToFetch)).map(Option(\_))

case \_ => Future.None

}

}

}