package com.twitter.frigate.pushservice.store

import com.twitter.conversions.DurationOps.\_

import com.twitter.frigate.common.history.History

import com.twitter.frigate.common.store.RealTimeClientEventStore

import com.twitter.frigate.data\_pipeline.common.HistoryJoin

import com.twitter.frigate.data\_pipeline.thriftscala.Event

import com.twitter.frigate.data\_pipeline.thriftscala.EventUnion

import com.twitter.frigate.data\_pipeline.thriftscala.PushRecSendEvent

import com.twitter.frigate.data\_pipeline.thriftscala.UserHistoryValue

import com.twitter.storehaus.ReadableStore

import com.twitter.util.Duration

import com.twitter.util.Future

import com.twitter.util.Time

case class OnlineUserHistoryKey(

userId: Long,

offlineUserHistory: Option[UserHistoryValue],

history: Option[History])

case class OnlineUserHistoryStore(

realTimeClientEventStore: RealTimeClientEventStore,

duration: Duration = 3.days)

extends ReadableStore[OnlineUserHistoryKey, UserHistoryValue] {

override def get(key: OnlineUserHistoryKey): Future[Option[UserHistoryValue]] = {

val now = Time.now

val pushRecSends = key.history

.getOrElse(History(Nil.toMap))

.sortedPushDmHistory

.filter(\_.\_1 > now - (duration + 1.day))

.map {

case (time, frigateNotification) =>

val pushRecSendEvent = PushRecSendEvent(

frigateNotification = Some(frigateNotification),

impressionId = frigateNotification.impressionId

)

pushRecSendEvent -> time

}

realTimeClientEventStore

.get(key.userId, now - duration, now)

.map { attributedEventHistory =>

val attributedClientEvents = attributedEventHistory.sortedHistory.flatMap {

case (time, event) =>

event.eventUnion match {

case Some(eventUnion: EventUnion.AttributedPushRecClientEvent) =>

Some((eventUnion.attributedPushRecClientEvent, event.eventType, time))

case \_ => None

}

}

val realtimeLabeledSends: Seq[Event] = HistoryJoin.getLabeledPushRecSends(

pushRecSends,

attributedClientEvents,

Seq(),

Seq(),

Seq(),

now

)

key.offlineUserHistory.map { offlineUserHistory =>

val combinedEvents = offlineUserHistory.events.map { offlineEvents =>

(offlineEvents ++ realtimeLabeledSends)

.map { event =>

event.timestampMillis -> event

}

.toMap

.values

.toSeq

.sortBy { event =>

-1 \* event.timestampMillis.getOrElse(0L)

}

}

offlineUserHistory.copy(events = combinedEvents)

}

}

}

}