package com.twitter.home\_mixer.store

import com.twitter.bijection.Injection

import com.twitter.home\_mixer.store.ManhattanRealGraphKVDescriptor.\_

import com.twitter.stitch.Stitch

import com.twitter.storage.client.manhattan.bijections.Bijections

import com.twitter.storage.client.manhattan.bijections.Bijections.BinaryScalaInjection

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

import com.twitter.storage.client.manhattan.kv.impl.ReadOnlyKeyDescriptor

import com.twitter.storage.client.manhattan.kv.impl.ValueDescriptor

import com.twitter.storehaus.ReadableStore

import com.twitter.util.Future

import com.twitter.wtf.candidate.{thriftscala => wtf}

object ManhattanRealGraphKVDescriptor {

implicit val byteArray2Buf = Bijections.BytesBijection

val realGraphDatasetName = "real\_graph\_scores\_in"

val keyInjection = Injection.connect[Long, Array[Byte]].andThen(Bijections.BytesInjection)

val keyDesc = ReadOnlyKeyDescriptor(keyInjection)

val valueDesc = ValueDescriptor(BinaryScalaInjection(wtf.CandidateSeq))

val realGraphDatasetKey = keyDesc.withDataset(realGraphDatasetName)

}

/\*\*

\* Hydrates real graph in network scores for a viewer

\*/

class RealGraphInNetworkScoresStore(manhattanKVEndpoint: ManhattanKVEndpoint)

extends ReadableStore[Long, Seq[wtf.Candidate]] {

override def get(viewerId: Long): Future[Option[Seq[wtf.Candidate]]] = Stitch

.run(manhattanKVEndpoint.get(realGraphDatasetKey.withPkey(viewerId), valueDesc))

.map(\_.map(mhResponse => mhResponse.contents.candidates))

}