package com.twitter.simclusters\_v2.summingbird.stores

import com.twitter.bijection.Injection

import com.twitter.bijection.scrooge.CompactScalaCodec

import com.twitter.simclusters\_v2.thriftscala.PersistedFullClusterId

import com.twitter.simclusters\_v2.thriftscala.TopProducersWithScore

import com.twitter.simclusters\_v2.thriftscala.TopSimClustersWithScore

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

import com.twitter.storehaus.ReadableStore

import com.twitter.storehaus\_internal.manhattan.Athena

import com.twitter.storehaus\_internal.manhattan.ManhattanRO

import com.twitter.storehaus\_internal.manhattan.ManhattanROConfig

import com.twitter.storehaus\_internal.util.ApplicationID

import com.twitter.storehaus\_internal.util.DatasetName

import com.twitter.storehaus\_internal.util.HDFSPath

object ProducerClusterEmbeddingReadableStores {

implicit val longInject: Injection[Long, Array[Byte]] = Injection.long2BigEndian

implicit val clusterInject: Injection[TopSimClustersWithScore, Array[Byte]] =

CompactScalaCodec(TopSimClustersWithScore)

implicit val producerInject: Injection[TopProducersWithScore, Array[Byte]] =

CompactScalaCodec(TopProducersWithScore)

implicit val clusterIdInject: Injection[PersistedFullClusterId, Array[Byte]] =

CompactScalaCodec(PersistedFullClusterId)

private val appId = "simclusters\_v2"

def getSimClusterEmbeddingTopKProducersStore(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[PersistedFullClusterId, TopProducersWithScore] = {

ManhattanRO.getReadableStoreWithMtls[PersistedFullClusterId, TopProducersWithScore](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("simcluster\_embedding\_top\_k\_producers\_by\_fav\_score\_20m\_145k\_updated"),

Athena

),

mhMtlsParams

)

}

def getProducerTopKSimClustersEmbeddingsStore(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[Long, TopSimClustersWithScore] = {

val datasetName = "producer\_top\_k\_simcluster\_embeddings\_by\_fav\_score\_20m\_145k\_updated"

ManhattanRO.getReadableStoreWithMtls[Long, TopSimClustersWithScore](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName(datasetName),

Athena

),

mhMtlsParams

)

}

def getProducerTopKSimClusters2020EmbeddingsStore(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[Long, TopSimClustersWithScore] = {

val datasetName = "producer\_top\_k\_simcluster\_embeddings\_by\_fav\_score\_20m\_145k\_2020"

ManhattanRO.getReadableStoreWithMtls[Long, TopSimClustersWithScore](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName(datasetName),

Athena

),

mhMtlsParams

)

}

def getSimClusterEmbeddingTopKProducersByFollowStore(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[PersistedFullClusterId, TopProducersWithScore] = {

ManhattanRO.getReadableStoreWithMtls[PersistedFullClusterId, TopProducersWithScore](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("simcluster\_embedding\_top\_k\_producers\_by\_follow\_score\_20m\_145k\_updated"),

Athena

),

mhMtlsParams

)

}

def getProducerTopKSimClustersEmbeddingsByFollowStore(

mhMtlsParams: ManhattanKVClientMtlsParams

): ReadableStore[Long, TopSimClustersWithScore] = {

ManhattanRO.getReadableStoreWithMtls[Long, TopSimClustersWithScore](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(appId),

DatasetName("producer\_top\_k\_simcluster\_embeddings\_by\_follow\_score\_20m\_145k\_2020"),

Athena

),

mhMtlsParams

)

}

}