package com.twitter.recos.user\_tweet\_entity\_graph

import com.twitter.graphjet.bipartite.api.EdgeTypeMask

import com.twitter.recos.recos\_common.thriftscala.SocialProofType

import com.twitter.recos.util.Action

/\*\*

\* The bit mask is used to encode edge types in the top bits of an integer,

\* e.g. favorite, retweet, reply and click. Under current segment configuration, each segment

\* stores up to 128M edges. Assuming that each node on one side is unique, each segment

\* stores up to 128M unique nodes on one side, which occupies the lower 27 bits of an integer.

\* This leaves five bits to encode the edge types, which at max can store 32 edge types.

\* The following implementation utilizes the top four bits and leaves one free bit out.

\*/

class UserTweetEdgeTypeMask extends EdgeTypeMask {

import UserTweetEdgeTypeMask.\_

override def encode(node: Int, edgeType: Byte): Int = {

if (edgeType < 0 || edgeType > SIZE || edgeType == Click.id.toByte) {

throw new IllegalArgumentException("encode: Illegal edge type argument " + edgeType)

} else {

node | (edgeType << 28)

}

}

override def edgeType(node: Int): Byte = {

(node >>> 28).toByte

}

override def restore(node: Int): Int = {

node & MASK

}

}

object UserTweetEdgeTypeMask extends Enumeration {

type UserTweetEdgeTypeMask = Value

/\*\*

\* Byte values corresponding to the action taken on a tweet, which will be encoded in the

\* top 4 bits in a tweet Id

\* NOTE: THERE CAN ONLY BE UP TO 16 TYPES

\*/

val Click: UserTweetEdgeTypeMask = Value(0)

val Favorite: UserTweetEdgeTypeMask = Value(1)

val Retweet: UserTweetEdgeTypeMask = Value(2)

val Reply: UserTweetEdgeTypeMask = Value(3)

val Tweet: UserTweetEdgeTypeMask = Value(4)

val IsMentioned: UserTweetEdgeTypeMask = Value(5)

val IsMediatagged: UserTweetEdgeTypeMask = Value(6)

val Quote: UserTweetEdgeTypeMask = Value(7)

val Unfavorite: UserTweetEdgeTypeMask = Value(8)

/\*\*

\* Reserve the top four bits of each integer to encode the edge type information.

\*/

val MASK: Int = Integer.parseInt("00001111111111111111111111111111", 2)

val SIZE: Int = this.values.size

/\*\*

\* Map valid social proof types specified by clients to an array of bytes. If clients do not

\* specify any social proof types in thrift, it will return all available social types by

\* default.

\*

\* @param socialProofTypes are the valid socialProofTypes specified by clients

\* @return an array of bytes representing valid social proof types

\*/

def getUserTweetGraphSocialProofTypes(

socialProofTypes: Option[Seq[SocialProofType]]

): Array[Byte] = {

socialProofTypes

.map { \_.map { \_.getValue }.toArray }

.getOrElse((0 until SIZE).toArray)

.map { \_.toByte }

}

/\*\*

\* Converts the action byte in the RecosHoseMessage into GraphJet internal byte mapping

\*/

def actionTypeToEdgeType(actionByte: Byte): Byte = {

val edgeType = Action(actionByte) match {

case Action.Favorite => Favorite.id

case Action.Retweet => Retweet.id

case Action.Reply => Reply.id

case Action.Tweet => Tweet.id

case Action.IsMentioned => IsMentioned.id

case Action.IsMediaTagged => IsMediatagged.id

case Action.Quote => Quote.id

case Action.Unfavorite => Unfavorite.id

case \_ =>

throw new IllegalArgumentException("getEdgeType: Illegal edge type argument " + actionByte)

}

edgeType.toByte

}

}