namespace java com.twitter.simclusters\_v2.thriftjava

namespace py gen.twitter.simclusters\_v2.multi\_type\_graph

#@namespace scala com.twitter.simclusters\_v2.thriftscala

#@namespace strato com.twitter.simclusters\_v2

include "entity.thrift"

union LeftNode {

1: i64 userId(personalDataType = 'UserId')

}(persisted = 'true', hasPersonalData = 'true')

struct RightNode {

1: required RightNodeType rightNodeType(personalDataType = 'EngagementsPublic')

2: required Noun noun

}(persisted = 'true', hasPersonalData = 'true')

struct RightNodeWithEdgeWeight {

1: required RightNode rightNode

2: required double weight(personalDataType = 'EngagementScore')

}(persisted = 'true', hasPersonalData = 'true')

enum RightNodeType {

FollowUser = 1,

FavUser = 2,

BlockUser = 3,

AbuseReportUser = 4,

SpamReportUser = 5,

FollowTopic = 6,

SignUpCountry = 7,

ConsumedLanguage = 8,

FavTweet = 9,

ReplyTweet = 10,

RetweetTweet = 11,

NotifOpenOrClickTweet = 12,

SearchQuery = 13

}(persisted = 'true')

union Noun {

// Note: Each of the following needs to have an ordering defined in Ordering[Noun]

// in file: multi\_type\_graph/assemble\_multi\_type\_graph/AssembleMultiTypeGraph.scala

// Please take note to make changes to Ordering[Noun] when modifying/adding new noun type here

1: i64 userId(personalDataType = 'UserId')

2: string country(personalDataType = 'InferredCountry')

3: string language(personalDataType = 'InferredLanguage')

4: i64 topicId(personalDataType = 'TopicFollow')

5: i64 tweetId(personalDataType = 'TweetId')

6: string query(personalDataType = 'SearchQuery')

}(persisted = 'true', hasPersonalData = 'true')

struct RightNodeWithEdgeWeightList {

1: required list<RightNodeWithEdgeWeight> rightNodeWithEdgeWeightList

}(persisted = 'true', hasPersonalData = 'true')

struct NounWithFrequency {

1: required Noun noun

2: required double frequency (personalDataType = 'EngagementScore')

}(persisted = 'true', hasPersonalData = 'true')

struct NounWithFrequencyList {

1: required list<NounWithFrequency> nounWithFrequencyList

}(persisted = 'true', hasPersonalData = 'true')

struct RightNodeTypeStruct {

1: required RightNodeType rightNodeType

}(persisted = 'true', hasPersonalData = 'false')

struct MultiTypeGraphEdge{

1: required LeftNode leftNode

2: required RightNodeWithEdgeWeight rightNodeWithEdgeWeight

}(persisted = 'true', hasPersonalData = 'true')

struct LeftNodeToRightNodeWithEdgeWeightList{

1: required LeftNode leftNode

2: required RightNodeWithEdgeWeightList rightNodeWithEdgeWeightList

}(persisted = 'true', hasPersonalData = 'true')

struct RightNodeSimHashSketch {

1: required RightNode rightNode

2: required list<byte> simHashOfEngagers

3: optional double normalizer

}(persisted='true', hasPersonalData = 'false')

struct SimilarRightNode {

1: required RightNode rightNode

2: required double score (personalDataType = 'EngagementScore')

}(persisted='true', hasPersonalData = 'true')

struct SimilarRightNodes {

1: required list<SimilarRightNode> rightNodesWithScores

}(persisted='true', hasPersonalData = 'true')

struct RightNodeWithScore {

1: required RightNode rightNode

2: required double clusterScore (personalDataType = 'EngagementScore')

}(persisted='true', hasPersonalData = 'true')

struct RightNodeWithScoreList {

1: required list<RightNodeWithScore> rightNodeWithScoreList

}(persisted='true', hasPersonalData = 'true')

struct RightNodeWithClusters {

1: required RightNode rightNode

2: required string modelVersion (personalDataType = 'EngagementId')

3: required map<i32, double> clusterIdToScores (personalDataTypeKey = 'EngagementId', personalDataTypeValue = 'EngagementScore')

}(persisted="true", hasPersonalData = 'true')

struct ModelVersionWithClusterScores {

1: required string modelVersion (personalDataType = 'EngagementId')

2: required map<i32, double> clusterIdToScores (personalDataTypeKey = 'EngagementId', personalDataTypeValue = 'EngagementScore')

}(persisted = 'true', hasPersonalData = 'true')