namespace java com.twitter.simclusters\_v2.thriftjava

namespace py gen.twitter.simclusters\_v2.online\_store

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

#@namespace strato com.twitter.simclusters\_v2

include "entity.thrift"

include "com/twitter/algebird\_internal/algebird.thrift"

/\*\*

\* A SimClusters model version.

\*\*/

enum ModelVersion {

MODEL\_20M\_145K\_dec11 = 1, // DEPRECATED

MODEL\_20M\_145K\_updated = 2, // DEPRECATED

MODEL\_20M\_145K\_2020 = 3,

RESERVED\_4 = 4,

RESERVED\_5 = 5,

RESERVED\_6 = 6

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

/\*\*

\* Uniquely identifies a SimCluster. All fields are required as this is used as a memcache key.

\*\*/

struct FullClusterId {

1: required ModelVersion modelVersion

2: required i32 clusterId

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

/\*\*

\* Contains a set of scores per cluster.

\*\*/

struct Scores {

1: optional algebird.DecayedValue favClusterNormalized8HrHalfLifeScore

2: optional algebird.DecayedValue followClusterNormalized8HrHalfLifeScore

}(hasPersonalData = 'false')

/\*\*

\* A combination of entity and model. All fields are required as this is used as a memcache key.

\*\*/

struct EntityWithVersion {

1: required entity.SimClusterEntity entity

2: required ModelVersion version

}(hasPersonalData = 'true')

/\*\*

\* Contains top K clusters with corresponding scores. We're representing clusters purely using ints, and

\* omitting the modelVersion, since that is included in the memcache key.

\*\*/

struct TopKClustersWithScores {

1: optional map<i32, Scores> topClustersByFavClusterNormalizedScore(personalDataTypeKey = 'InferredInterests')

2: optional map<i32, Scores> topClustersByFollowClusterNormalizedScore(personalDataTypeKey = 'InferredInterests')

}(hasPersonalData = 'true')

/\*\*

\* Contains top K text entities with corresponding scores. We're omitting the modelVersion,

\* since that is included in the memcache key.

\*\*/

struct TopKEntitiesWithScores {

1: optional map<entity.TweetTextEntity, Scores> topEntitiesByFavClusterNormalizedScore

2: optional map<entity.TweetTextEntity, Scores> topEntitiesByFollowClusterNormalizedScore

}(hasPersonalData = 'true')

/\*\*

\* Contains top K tweets with corresponding scores. We're omitting the modelVersion,

\* since that is included in the memcache key.

\*\*/

struct TopKTweetsWithScores {

1: optional map<i64, Scores> topTweetsByFavClusterNormalizedScore(personalDataTypeKey='TweetId')

2: optional map<i64, Scores> topTweetsByFollowClusterNormalizedScore(personalDataTypeKey='TweetId')

}(hasPersonalData = 'true')

/\*\*

\* Contains FullClusterId and the corresponding top K tweets and scores.

\*\*/

struct ClusterIdToTopKTweetsWithScores {

1: required FullClusterId clusterId

2: required TopKTweetsWithScores topKTweetsWithScores

}(hasPersonalData = 'true')

/\*\*

\* Contains a map of Model Version to top K clusters with corresponding scores.

\*\*/

struct MultiModelTopKClustersWithScores {

1: optional map<ModelVersion, TopKClustersWithScores> multiModelTopKClustersWithScores

}(hasPersonalData = 'true')

/\*\*

\* Contains a map of Model Version top K tweets with corresponding scores.

\*\*/

struct MultiModelTopKTweetsWithScores {

1: optional map<ModelVersion, TopKTweetsWithScores> multiModelTopKTweetsWithScores

}(hasPersonalData = 'true')