package com.twitter.simclusters\_v2.score

import com.twitter.simclusters\_v2.common.SimClustersEmbeddingId.\_

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

InternalId,

ScoreInternalId,

ScoringAlgorithm,

SimClustersEmbeddingId,

GenericPairScoreId => ThriftGenericPairScoreId,

ScoreId => ThriftScoreId,

SimClustersEmbeddingPairScoreId => ThriftSimClustersEmbeddingPairScoreId

}

/\*\*

\* A uniform Identifier type for all kinds of Calculation Score.

\*\*/

trait ScoreId {

def algorithm: ScoringAlgorithm

/\*\*

\* Convert to a Thrift object. Throw a exception if the operation is not override.

\*/

implicit def toThrift: ThriftScoreId =

throw new UnsupportedOperationException(s"ScoreId $this doesn't support Thrift format")

}

object ScoreId {

implicit val fromThriftScoreId: ThriftScoreId => ScoreId = {

case scoreId @ ThriftScoreId(\_, ScoreInternalId.GenericPairScoreId(\_)) =>

PairScoreId.fromThriftScoreId(scoreId)

case scoreId @ ThriftScoreId(\_, ScoreInternalId.SimClustersEmbeddingPairScoreId(\_)) =>

SimClustersEmbeddingPairScoreId.fromThriftScoreId(scoreId)

}

}

/\*\*

\* Generic Internal pairwise id. Support all the subtypes in InternalId, which includes TweetId,

\* UserId, EntityId and more combination ids.

\*\*/

trait PairScoreId extends ScoreId {

def id1: InternalId

def id2: InternalId

override implicit lazy val toThrift: ThriftScoreId = {

ThriftScoreId(

algorithm,

ScoreInternalId.GenericPairScoreId(ThriftGenericPairScoreId(id1, id2))

)

}

}

object PairScoreId {

// The default PairScoreId assume id1 <= id2. It used to increase the cache hit rate.

def apply(algorithm: ScoringAlgorithm, id1: InternalId, id2: InternalId): PairScoreId = {

if (internalIdOrdering.lteq(id1, id2)) {

DefaultPairScoreId(algorithm, id1, id2)

} else {

DefaultPairScoreId(algorithm, id2, id1)

}

}

private case class DefaultPairScoreId(

algorithm: ScoringAlgorithm,

id1: InternalId,

id2: InternalId)

extends PairScoreId

implicit val fromThriftScoreId: ThriftScoreId => PairScoreId = {

case ThriftScoreId(algorithm, ScoreInternalId.GenericPairScoreId(pairScoreId)) =>

DefaultPairScoreId(algorithm, pairScoreId.id1, pairScoreId.id2)

case ThriftScoreId(algorithm, ScoreInternalId.SimClustersEmbeddingPairScoreId(pairScoreId)) =>

SimClustersEmbeddingPairScoreId(algorithm, pairScoreId.id1, pairScoreId.id2)

}

}

/\*\*

\* ScoreId for a pair of SimClustersEmbedding.

\* Used for dot product, cosine similarity and other basic embedding operations.

\*/

trait SimClustersEmbeddingPairScoreId extends PairScoreId {

def embeddingId1: SimClustersEmbeddingId

def embeddingId2: SimClustersEmbeddingId

override def id1: InternalId = embeddingId1.internalId

override def id2: InternalId = embeddingId2.internalId

override implicit lazy val toThrift: ThriftScoreId = {

ThriftScoreId(

algorithm,

ScoreInternalId.SimClustersEmbeddingPairScoreId(

ThriftSimClustersEmbeddingPairScoreId(embeddingId1, embeddingId2))

)

}

}

object SimClustersEmbeddingPairScoreId {

// The default PairScoreId assume id1 <= id2. It used to increase the cache hit rate.

def apply(

algorithm: ScoringAlgorithm,

id1: SimClustersEmbeddingId,

id2: SimClustersEmbeddingId

): SimClustersEmbeddingPairScoreId = {

if (simClustersEmbeddingIdOrdering.lteq(id1, id2)) {

DefaultSimClustersEmbeddingPairScoreId(algorithm, id1, id2)

} else {

DefaultSimClustersEmbeddingPairScoreId(algorithm, id2, id1)

}

}

private case class DefaultSimClustersEmbeddingPairScoreId(

algorithm: ScoringAlgorithm,

embeddingId1: SimClustersEmbeddingId,

embeddingId2: SimClustersEmbeddingId)

extends SimClustersEmbeddingPairScoreId

implicit val fromThriftScoreId: ThriftScoreId => SimClustersEmbeddingPairScoreId = {

case ThriftScoreId(algorithm, ScoreInternalId.SimClustersEmbeddingPairScoreId(pairScoreId)) =>

SimClustersEmbeddingPairScoreId(algorithm, pairScoreId.id1, pairScoreId.id2)

}

}