package com.twitter.search.earlybird.partition;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

public class EarlybirdIndex {

private final List<SegmentInfo> segmentInfoList;

public static final int MAX\_NUM\_OF\_NON\_OPTIMIZED\_SEGMENTS = 2;

// The Kafka offsets for the tweet create stream and the tweet update stream. Indexing should

// start from these offsets when it resumes.

private final long tweetOffset;

private final long updateOffset;

private final long maxIndexedTweetId;

public EarlybirdIndex(

List<SegmentInfo> segmentInfoList,

long tweetOffset,

long updateOffset,

long maxIndexedTweetId

) {

List<SegmentInfo> segmentInfos = new ArrayList<>(segmentInfoList);

Collections.sort(segmentInfos);

this.segmentInfoList = segmentInfos;

this.tweetOffset = tweetOffset;

this.updateOffset = updateOffset;

this.maxIndexedTweetId = maxIndexedTweetId;

}

public EarlybirdIndex(List<SegmentInfo> segmentInfoList, long tweetOffset, long updateOffset) {

this(segmentInfoList, tweetOffset, updateOffset, -1);

}

public List<SegmentInfo> getSegmentInfoList() {

return segmentInfoList;

}

public long getTweetOffset() {

return tweetOffset;

}

public long getUpdateOffset() {

return updateOffset;

}

public long getMaxIndexedTweetId() {

return maxIndexedTweetId;

}

/\*\*

\* Returns the number of non-optimized segments in this index.

\* @return the number of non-optimized segments in this index.

\*/

public int numOfNonOptimizedSegments() {

int numNonOptimized = 0;

for (SegmentInfo segmentInfo : segmentInfoList) {

if (!segmentInfo.isOptimized()) {

numNonOptimized++;

}

}

return numNonOptimized;

}

}