package com.twitter.search.ingester.pipeline.twitter;

import java.util.List;

import javax.naming.NamingException;

import com.google.common.base.Preconditions;

import com.google.common.collect.Lists;

import org.apache.commons.pipeline.StageException;

import org.apache.commons.pipeline.validation.ConsumedTypes;

import org.apache.commons.pipeline.validation.ProducedTypes;

import com.twitter.common\_internal.text.version.PenguinVersion;

import com.twitter.search.common.converter.earlybird.DelayedIndexingConverter;

import com.twitter.search.common.indexing.thriftjava.ThriftVersionedEvents;

import com.twitter.search.common.relevance.entities.TwitterMessage;

import com.twitter.search.common.schema.base.Schema;

import com.twitter.search.common.schema.earlybird.EarlybirdSchemaCreateTool;

import com.twitter.search.ingester.model.IngesterThriftVersionedEvents;

import com.twitter.search.ingester.model.IngesterTwitterMessage;

@ConsumedTypes(IngesterTwitterMessage.class)

@ProducedTypes(IngesterThriftVersionedEvents.class)

public class ConvertDelayedMessageToThriftStage extends TwitterBaseStage

<TwitterMessage, IngesterThriftVersionedEvents> {

private List<PenguinVersion> penguinVersionList;

private FieldStatExporter fieldStatExporter;

private DelayedIndexingConverter messageConverter;

@Override

protected void doInnerPreprocess() throws StageException, NamingException {

Schema schema;

try {

schema = EarlybirdSchemaCreateTool.buildSchema(Preconditions.checkNotNull(earlybirdCluster));

} catch (Schema.SchemaValidationException e) {

throw new StageException(this, e);

}

penguinVersionList = wireModule.getPenguinVersions();

messageConverter = new DelayedIndexingConverter(schema, decider);

fieldStatExporter = new FieldStatExporter("unsorted\_urls", schema, penguinVersionList);

}

@Override

public void innerProcess(Object obj) throws StageException {

if (!(obj instanceof IngesterTwitterMessage)) {

throw new StageException(this, "Object is not an IngesterTwitterMessage instance: " + obj);

}

penguinVersionList = wireModule.getCurrentlyEnabledPenguinVersions();

fieldStatExporter.updatePenguinVersions(penguinVersionList);

IngesterTwitterMessage message = IngesterTwitterMessage.class.cast(obj);

for (IngesterThriftVersionedEvents events : buildVersionedEvents(message)) {

fieldStatExporter.addFieldStats(events);

emitAndCount(events);

}

}

/\*\*

\* Method that converts all URL and card related fields and features of a TwitterMessage to a

\* ThriftVersionedEvents instance.

\*

\* @param twitterMessage An IngesterThriftVersionedEvents instance to be converted.

\* @return The corresponding ThriftVersionedEvents instance.

\*/

private List<IngesterThriftVersionedEvents> buildVersionedEvents(

IngesterTwitterMessage twitterMessage) {

List<ThriftVersionedEvents> versionedEvents =

messageConverter.convertMessageToOutOfOrderAppendAndFeatureUpdate(

twitterMessage, penguinVersionList);

Preconditions.checkArgument(

versionedEvents.size() == 2,

"DelayedIndexingConverter produced an incorrect number of ThriftVersionedEvents.");

return Lists.newArrayList(

toIngesterThriftVersionedEvents(versionedEvents.get(0), twitterMessage),

toIngesterThriftVersionedEvents(versionedEvents.get(1), twitterMessage));

}

private IngesterThriftVersionedEvents toIngesterThriftVersionedEvents(

ThriftVersionedEvents versionedEvents, IngesterTwitterMessage twitterMessage) {

// We don't want to propagate the same DebugEvents instance to multiple

// IngesterThriftVersionedEvents instances, because future stages might want to add new events

// to this list for multiple events at the same time, which would result in a

// ConcurrentModificationException. So we need to create a DebugEvents deep copy.

IngesterThriftVersionedEvents ingesterThriftVersionedEvents =

new IngesterThriftVersionedEvents(twitterMessage.getUserId());

ingesterThriftVersionedEvents.setDarkWrite(false);

ingesterThriftVersionedEvents.setId(twitterMessage.getTweetId());

ingesterThriftVersionedEvents.setVersionedEvents(versionedEvents.getVersionedEvents());

ingesterThriftVersionedEvents.setDebugEvents(twitterMessage.getDebugEvents().deepCopy());

return ingesterThriftVersionedEvents;

}

}