package com.twitter.search.common.schema.earlybird;

import com.google.common.base.Preconditions;

import com.google.common.collect.ImmutableList;

import com.twitter.common.text.util.TokenStreamSerializer;

import com.twitter.search.common.schema.SchemaBuilder;

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

import com.twitter.search.common.schema.thriftjava.ThriftFieldConfiguration;

import com.twitter.search.common.schema.thriftjava.ThriftFieldSettings;

import com.twitter.search.common.schema.thriftjava.ThriftTokenStreamSerializer;

import com.twitter.search.common.util.analysis.CharTermAttributeSerializer;

import com.twitter.search.common.util.analysis.TermPayloadAttributeSerializer;

/\*\*

\* Build class used to build a ThriftSchema

\*/

public class EarlybirdSchemaBuilder extends SchemaBuilder {

private final EarlybirdCluster cluster;

public EarlybirdSchemaBuilder(FieldNameToIdMapping idMapping,

EarlybirdCluster cluster,

TokenStreamSerializer.Version tokenStreamSerializerVersion) {

super(idMapping, tokenStreamSerializerVersion);

this.cluster = cluster;

}

/\*\*

\* Configure the specified field to be Out-of-order.

\* In the realtime cluster, this causes Earlybird to used the skip list posting format.

\*/

public final EarlybirdSchemaBuilder withOutOfOrderEnabledForField(String fieldName) {

if (!shouldIncludeField(fieldName)) {

return this;

}

ThriftFieldSettings settings =

schema.getFieldConfigs().get(idMapping.getFieldID(fieldName)).getSettings();

Preconditions.checkState(settings.isSetIndexedFieldSettings(),

"Out of order field must be indexed");

settings.getIndexedFieldSettings().setSupportOutOfOrderAppends(true);

return this;

}

/\*\*

\* This turns on tweet specific normalizations. This turns on the following two token processors:

\* {@link com.twitter.search.common.util.text.splitter.HashtagMentionPunctuationSplitter}

\* {@link com.twitter.search.common.util.text.filter.NormalizedTokenFilter}

\* <p/>

\* HashtagMentionPunctuationSplitter would break a mention or hashtag like @ab\_cd or #ab\_cd into

\* tokens {ab, cd}.

\* NormalizedTokenFilter strips out the # @ $ from the tokens.

\*/

public final EarlybirdSchemaBuilder withTweetSpecificNormalization(String fieldName) {

if (!shouldIncludeField(fieldName)) {

return this;

}

ThriftFieldSettings settings =

schema.getFieldConfigs().get(idMapping.getFieldID(fieldName)).getSettings();

Preconditions.checkState(settings.isSetIndexedFieldSettings(),

"Tweet text field must be indexed.");

settings.getIndexedFieldSettings().setDeprecated\_performTweetSpecificNormalizations(true);

return this;

}

/\*\*

\* Add a twitter photo facet field.

\*/

public final EarlybirdSchemaBuilder withPhotoUrlFacetField(String fieldName) {

if (!shouldIncludeField(fieldName)) {

return this;

}

ThriftFieldSettings photoFieldSettings = getNoPositionNoFreqSettings();

ThriftTokenStreamSerializer tokenStreamSerializer =

new ThriftTokenStreamSerializer(tokenStreamSerializerVersion);

tokenStreamSerializer.setAttributeSerializerClassNames(

ImmutableList.<String>of(

CharTermAttributeSerializer.class.getName(),

TermPayloadAttributeSerializer.class.getName()));

photoFieldSettings

.getIndexedFieldSettings()

.setTokenStreamSerializer(tokenStreamSerializer)

.setTokenized(true);

putIntoFieldConfigs(idMapping.getFieldID(fieldName),

new ThriftFieldConfiguration(fieldName).setSettings(photoFieldSettings));

return this;

}

/\*\*

\* Returns whether the given field should be included or dropped.

\*/

@Override

protected boolean shouldIncludeField(String fieldName) {

return EarlybirdFieldConstants.getFieldConstant(fieldName).isValidFieldInCluster(cluster);

}

}