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

import java.io.IOException;

import java.util.Iterator;

import java.util.List;

import com.google.common.collect.ImmutableList;

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

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

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

import com.twitter.search.common.schema.earlybird.EarlybirdFieldConstants.EarlybirdFieldConstant;

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

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

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

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

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

/\*\*

\* Utility APIs for ThriftDocument used in Earlybird.

\*/

public final class EarlybirdThriftDocumentUtil {

private static final EarlybirdFieldConstants ID\_MAPPING = new EarlybirdFieldConstants();

private static final String FILTER\_FORMAT\_STRING = "\_\_filter\_%s";

/\*\*

\* Used to check whether a thrift document has filter nullcast internal field set.

\* @see #isNullcastFilterSet(ThriftDocument)

\*/

private static final String NULLCAST\_FILTER\_TERM =

formatFilter(EarlybirdFieldConstant.NULLCAST\_FILTER\_TERM);

private static final String SELF\_THREAD\_FILTER\_TERM =

formatFilter(EarlybirdFieldConstant.SELF\_THREAD\_FILTER\_TERM);

private static final String DIRECTED\_AT\_FILTER\_TERM =

formatFilter(EarlybirdFieldConstant.DIRECTED\_AT\_FILTER\_TERM);

private EarlybirdThriftDocumentUtil() {

// Cannot instantiate.

}

/\*\*

\* Formats a regular, simple filter term. The 'filter' argument should correspond to a constant

\* from the Operator class, matching the operand (filter:links -> "links").

\*/

public static final String formatFilter(String filter) {

return String.format(FILTER\_FORMAT\_STRING, filter);

}

/\*\*

\* Get status id.

\*/

public static long getID(ThriftDocument document) {

return ThriftDocumentUtil.getLongValue(

document, EarlybirdFieldConstant.ID\_FIELD.getFieldName(), ID\_MAPPING);

}

/\*\*

\* Get Card name.

\*/

public static String getCardName(ThriftDocument document) {

return ThriftDocumentUtil.getStringValue(

document, EarlybirdFieldConstant.CARD\_NAME\_FIELD.getFieldName(), ID\_MAPPING);

}

/\*\*

\* Get Card language.

\*/

public static String getCardLang(ThriftDocument document) {

return ThriftDocumentUtil.getStringValue(

document, EarlybirdFieldConstant.CARD\_LANG.getFieldName(), ID\_MAPPING);

}

/\*\*

\* Get Card language CSF.

\*

\* card language CSF is represented internally as an integer ID for a ThriftLanguage.

\*/

public static int getCardLangCSF(ThriftDocument document) {

return ThriftDocumentUtil.getIntValue(

document, EarlybirdFieldConstant.CARD\_LANG\_CSF.getFieldName(), ID\_MAPPING);

}

/\*\*

\* Get quoted tweet id.

\*/

public static long getQuotedTweetID(ThriftDocument document) {

return ThriftDocumentUtil.getLongValue(

document, EarlybirdFieldConstant.QUOTED\_TWEET\_ID\_FIELD.getFieldName(), ID\_MAPPING);

}

/\*\*

\* Get quoted tweet user id.

\*/

public static long getQuotedUserID(ThriftDocument document) {

return ThriftDocumentUtil.getLongValue(

document, EarlybirdFieldConstant.QUOTED\_USER\_ID\_FIELD.getFieldName(), ID\_MAPPING);

}

/\*\*

\* Get directed at user id.

\*/

public static long getDirectedAtUserId(ThriftDocument document) {

return ThriftDocumentUtil.getLongValue(

document, EarlybirdFieldConstant.DIRECTED\_AT\_USER\_ID\_FIELD.getFieldName(), ID\_MAPPING);

}

/\*\*

\* Get directed at user id CSF.

\*/

public static long getDirectedAtUserIdCSF(ThriftDocument document) {

return ThriftDocumentUtil.getLongValue(

document, EarlybirdFieldConstant.DIRECTED\_AT\_USER\_ID\_CSF.getFieldName(), ID\_MAPPING);

}

/\*\*

\* Get reference author id CSF.

\*/

public static long getReferenceAuthorIdCSF(ThriftDocument document) {

return ThriftDocumentUtil.getLongValue(

document, EarlybirdFieldConstant.REFERENCE\_AUTHOR\_ID\_CSF.getFieldName(), ID\_MAPPING);

}

/\*\*

\* Get links.

\*/

public static List<String> getLinks(ThriftDocument document) {

return getStringValues(document, EarlybirdFieldConstant.LINKS\_FIELD);

}

/\*\*

\* Get created at timestamp in sec.

\*/

public static int getCreatedAtSec(ThriftDocument document) {

return ThriftDocumentUtil.getIntValue(

document, EarlybirdFieldConstant.CREATED\_AT\_FIELD.getFieldName(), ID\_MAPPING);

}

/\*\*

\* Get created at timestamp in ms.

\*/

public static long getCreatedAtMs(ThriftDocument document) {

long createdAtSec = (long) getCreatedAtSec(document);

return createdAtSec \* 1000L;

}

/\*\*

\* Get from user id.

\*/

public static long getFromUserID(ThriftDocument document) {

return ThriftDocumentUtil.getLongValue(

document, EarlybirdFieldConstant.FROM\_USER\_ID\_FIELD.getFieldName(), ID\_MAPPING);

}

/\*\*

\* Get from user.

\*/

public static String getFromUser(ThriftDocument document) {

return ThriftDocumentUtil.getStringValue(

document, EarlybirdFieldConstant.FROM\_USER\_FIELD.getFieldName(), ID\_MAPPING);

}

/\*\*

\* Get tokenized from user display name.

\*/

public static String getFromUserDisplayName(ThriftDocument document) {

return ThriftDocumentUtil.getStringValue(

document, EarlybirdFieldConstant.TOKENIZED\_USER\_NAME\_FIELD.getFieldName(), ID\_MAPPING);

}

/\*\*

\* Get tokenized from user.

\*/

public static String getTokenizedFromUser(ThriftDocument document) {

return ThriftDocumentUtil.getStringValue(

document, EarlybirdFieldConstant.TOKENIZED\_FROM\_USER\_FIELD.getFieldName(), ID\_MAPPING);

}

/\*\*

\* Get resolved links text.

\*/

public static String getResolvedLinksText(ThriftDocument document) {

return ThriftDocumentUtil.getStringValue(

document, EarlybirdFieldConstant.RESOLVED\_LINKS\_TEXT\_FIELD.getFieldName(), ID\_MAPPING);

}

/\*\*

\* Get iso language code.

\*/

public static List<String> getISOLanguage(ThriftDocument document) {

return ThriftDocumentUtil.getStringValues(

document, EarlybirdFieldConstant.ISO\_LANGUAGE\_FIELD.getFieldName(), ID\_MAPPING);

}

/\*\*

\* First remove the old timestamp if they exist.

\* Then add the created at and created at csf fields to the given thrift document.

\*/

public static void replaceCreatedAtAndCreatedAtCSF(ThriftDocument document, int value) {

removeField(document, EarlybirdFieldConstant.CREATED\_AT\_FIELD);

removeField(document, EarlybirdFieldConstant.CREATED\_AT\_CSF\_FIELD);

addIntField(document, EarlybirdFieldConstant.CREATED\_AT\_FIELD, value);

addIntField(document, EarlybirdFieldConstant.CREATED\_AT\_CSF\_FIELD, value);

}

/\*\*

\* Add the given int value as the given field into the given document.

\*/

public static ThriftDocument addIntField(

ThriftDocument document, EarlybirdFieldConstant fieldConstant, int value) {

ThriftFieldData fieldData = new ThriftFieldData().setIntValue(value);

ThriftField field =

new ThriftField().setFieldConfigId(fieldConstant.getFieldId()).setFieldData(fieldData);

document.addToFields(field);

return document;

}

private static EarlybirdFieldConstant getFeatureField(EarlybirdFieldConstant field) {

if (field.getFieldName().startsWith(

EarlybirdFieldConstant.ENCODED\_TWEET\_FEATURES\_FIELD.getFieldName())) {

return EarlybirdFieldConstant.ENCODED\_TWEET\_FEATURES\_FIELD;

} else if (field.getFieldName().startsWith(

EarlybirdFieldConstant.EXTENDED\_ENCODED\_TWEET\_FEATURES\_FIELD.getFieldName())) {

return EarlybirdFieldConstant.EXTENDED\_ENCODED\_TWEET\_FEATURES\_FIELD;

} else {

throw new IllegalArgumentException("Not a feature field: " + field);

}

}

/\*\*

\* Get the feature value of a field.

\*/

public static int getFeatureValue(

ImmutableSchemaInterface schema,

ThriftDocument document,

EarlybirdFieldConstant field) {

EarlybirdFieldConstant featureField = getFeatureField(field);

byte[] encodedFeaturesBytes =

ThriftDocumentUtil.getBytesValue(document, featureField.getFieldName(), ID\_MAPPING);

if (encodedFeaturesBytes == null) {

// Treat the feature value as 0 if there is no encoded feature field.

return 0;

} else {

EarlybirdEncodedFeatures encodedFeatures = EarlybirdEncodedFeaturesUtil.fromBytes(

schema, featureField, encodedFeaturesBytes, 0);

return encodedFeatures.getFeatureValue(field);

}

}

/\*\*

\* Check whether the feature flag is set.

\*/

public static boolean isFeatureBitSet(

ImmutableSchemaInterface schema,

ThriftDocument document,

EarlybirdFieldConstant field) {

EarlybirdFieldConstant featureField = getFeatureField(field);

byte[] encodedFeaturesBytes =

ThriftDocumentUtil.getBytesValue(document, featureField.getFieldName(), ID\_MAPPING);

if (encodedFeaturesBytes == null) {

// Treat the bit as not set if there is no encoded feature field.

return false;

} else {

EarlybirdEncodedFeatures encodedFeatures = EarlybirdEncodedFeaturesUtil.fromBytes(

schema, featureField, encodedFeaturesBytes, 0);

return encodedFeatures.isFlagSet(field);

}

}

/\*\*

\* Check whether nullcast flag is set in the encoded features field.

\*/

public static boolean isNullcastBitSet(ImmutableSchemaInterface schema, ThriftDocument document) {

return isFeatureBitSet(schema, document, EarlybirdFieldConstant.IS\_NULLCAST\_FLAG);

}

/\*\*

\* Remove all fields with the given field constant in a document.

\*/

public static void removeField(ThriftDocument document, EarlybirdFieldConstant fieldConstant) {

List<ThriftField> fields = document.getFields();

if (fields != null) {

Iterator<ThriftField> fieldsIterator = fields.iterator();

while (fieldsIterator.hasNext()) {

if (fieldsIterator.next().getFieldConfigId() == fieldConstant.getFieldId()) {

fieldsIterator.remove();

}

}

}

}

/\*\*

\* Remove a string field with given fieldConstant and value.

\*/

public static void removeStringField(

ThriftDocument document, EarlybirdFieldConstant fieldConstant, String value) {

List<ThriftField> fields = document.getFields();

if (fields != null) {

for (ThriftField field : fields) {

if (field.getFieldConfigId() == fieldConstant.getFieldId()

&& field.getFieldData().getStringValue().equals(value)) {

fields.remove(field);

return;

}

}

}

}

/\*\*

\* Adds a new TokenStream field for each engagement counter if normalizedNumEngagements >= 1.

\*/

public static void addNormalizedMinEngagementField(

ThriftDocument doc,

String fieldName,

int normalizedNumEngagements) throws IOException {

if (normalizedNumEngagements < 1) {

return;

}

TokenStreamSerializer serializer =

new TokenStreamSerializer(ImmutableList.of(new IntTermAttributeSerializer()));

TwitterNormalizedMinEngagementTokenStream stream = new

TwitterNormalizedMinEngagementTokenStream(normalizedNumEngagements);

byte[] serializedStream = serializer.serialize(stream);

ThriftFieldData fieldData = new ThriftFieldData().setTokenStreamValue(serializedStream);

ThriftField field = new ThriftField().setFieldConfigId(ID\_MAPPING.getFieldID(fieldName))

.setFieldData(fieldData);

doc.addToFields(field);

}

public static List<String> getStringValues(

ThriftDocument document, EarlybirdFieldConstant field) {

return ThriftDocumentUtil.getStringValues(document, field.getFieldName(), ID\_MAPPING);

}

public static boolean isNullcastFilterSet(ThriftDocument document) {

return isFilterSet(document, NULLCAST\_FILTER\_TERM);

}

public static boolean isSelfThreadFilterSet(ThriftDocument document) {

return isFilterSet(document, SELF\_THREAD\_FILTER\_TERM);

}

public static String getSelfThreadFilterTerm() {

return SELF\_THREAD\_FILTER\_TERM;

}

public static String getDirectedAtFilterTerm() {

return DIRECTED\_AT\_FILTER\_TERM;

}

public static boolean isDirectedAtFilterSet(ThriftDocument document) {

return isFilterSet(document, DIRECTED\_AT\_FILTER\_TERM);

}

/\*\*

\* Check whether given filter is set in the internal field.

\*/

private static boolean isFilterSet(ThriftDocument document, String filter) {

List<String> terms = ThriftDocumentUtil.getStringValues(

document, EarlybirdFieldConstant.INTERNAL\_FIELD.getFieldName(), ID\_MAPPING);

for (String term : terms) {

if (filter.equals(term)) {

return true;

}

}

return false;

}

}