package com.twitter.search.earlybird\_root.visitors;

import java.util.Collections;

import java.util.List;

import java.util.stream.Collectors;

import com.google.common.collect.ImmutableList;

import com.google.common.collect.Lists;

import com.twitter.search.common.partitioning.base.PartitionDataType;

import com.twitter.search.common.partitioning.base.PartitionMappingManager;

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

import com.twitter.search.queryparser.query.Conjunction;

import com.twitter.search.queryparser.query.Disjunction;

import com.twitter.search.queryparser.query.Query;

import com.twitter.search.queryparser.query.Query.Occur;

import com.twitter.search.queryparser.query.QueryParserException;

import com.twitter.search.queryparser.query.search.SearchOperator;

import com.twitter.search.queryparser.query.search.SearchQueryTransformer;

/\*\*

\* Truncate user id or id lists in [multi\_term\_disjunction from\_user\_id/id] queries.

\* Return null if query has incorrect operators or looked at wrong field.

\*/

public class MultiTermDisjunctionPerPartitionVisitor extends SearchQueryTransformer {

private final PartitionMappingManager partitionMappingManager;

private final int partitionId;

private final String targetFieldName;

public static final Conjunction NO\_MATCH\_CONJUNCTION =

new Conjunction(Occur.MUST\_NOT, Collections.emptyList(), Collections.emptyList());

public MultiTermDisjunctionPerPartitionVisitor(

PartitionMappingManager partitionMappingManager,

int partitionId) {

this.partitionMappingManager = partitionMappingManager;

this.partitionId = partitionId;

this.targetFieldName =

partitionMappingManager.getPartitionDataType() == PartitionDataType.USER\_ID

? EarlybirdFieldConstants.EarlybirdFieldConstant.FROM\_USER\_ID\_FIELD.getFieldName()

: EarlybirdFieldConstants.EarlybirdFieldConstant.ID\_FIELD.getFieldName();

}

private boolean isTargetedQuery(Query query) {

if (query instanceof SearchOperator) {

SearchOperator operator = (SearchOperator) query;

return operator.getOperatorType() == SearchOperator.Type.MULTI\_TERM\_DISJUNCTION

&& operator.getOperand().equals(targetFieldName);

} else {

return false;

}

}

@Override

public Query visit(Conjunction query) throws QueryParserException {

boolean modified = false;

ImmutableList.Builder<Query> children = ImmutableList.builder();

for (Query child : query.getChildren()) {

Query newChild = child.accept(this);

if (newChild != null) {

// For conjunction case, if any child is "multi\_term\_disjunction from\_user\_id" and returns

// Conjunction.NO\_MATCH\_CONJUNCTION, it should be considered same as match no docs. And

// caller should decide how to deal with it.

if (isTargetedQuery(child) && newChild == NO\_MATCH\_CONJUNCTION) {

return NO\_MATCH\_CONJUNCTION;

}

if (newChild != Conjunction.EMPTY\_CONJUNCTION

&& newChild != Disjunction.EMPTY\_DISJUNCTION) {

children.add(newChild);

}

}

if (newChild != child) {

modified = true;

}

}

return modified ? query.newBuilder().setChildren(children.build()).build() : query;

}

@Override

public Query visit(Disjunction disjunction) throws QueryParserException {

boolean modified = false;

ImmutableList.Builder<Query> children = ImmutableList.builder();

for (Query child : disjunction.getChildren()) {

Query newChild = child.accept(this);

if (newChild != null

&& newChild != Conjunction.EMPTY\_CONJUNCTION

&& newChild != Disjunction.EMPTY\_DISJUNCTION

&& newChild != NO\_MATCH\_CONJUNCTION) {

children.add(newChild);

}

if (newChild != child) {

modified = true;

}

}

return modified ? disjunction.newBuilder().setChildren(children.build()).build() : disjunction;

}

@Override

public Query visit(SearchOperator operator) throws QueryParserException {

if (isTargetedQuery(operator)) {

List<Long> ids = extractIds(operator);

if (ids.size() > 0) {

List<String> operands = Lists.newArrayList(targetFieldName);

for (long id : ids) {

operands.add(String.valueOf(id));

}

return operator.newBuilder().setOperands(operands).build();

} else {

// If the [multi\_term\_disjunction from\_user\_id] is a negation (i.e., occur == MUST\_NOT),

// and there is no user id left, the whole sub query node does not do anything; if it is

// NOT a negation, then sub query matches nothing.

if (operator.getOccur() == Query.Occur.MUST\_NOT) {

return Conjunction.EMPTY\_CONJUNCTION;

} else {

return NO\_MATCH\_CONJUNCTION;

}

}

}

return operator;

}

private List<Long> extractIds(SearchOperator operator) throws QueryParserException {

if (EarlybirdFieldConstants.EarlybirdFieldConstant.ID\_FIELD

.getFieldName().equals(targetFieldName)) {

return operator.getOperands().subList(1, operator.getNumOperands()).stream()

.map(Long::valueOf)

.filter(id -> partitionMappingManager.getPartitionIdForTweetId(id) == partitionId)

.collect(Collectors.toList());

} else {

return operator.getOperands().subList(1, operator.getNumOperands()).stream()

.map(Long::valueOf)

.filter(id -> partitionMappingManager.getPartitionIdForUserId(id) == partitionId)

.collect(Collectors.toList());

}

}

}