Database - Bug #7667

Support OUTER-JOIN with PRESELECT at conversion

08/08/2023 04:06 AM - Alexandru Lungu

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Andrei Bălteanu	**% Done:**	30%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 08/08/2023 04:15 AM - Alexandru Lungu

- Status changed from New to WIP

- Subject changed from Support OUTER-JOIN with PRESELECT to Support OUTER-JOIN with PRESELECT at conversion

This is a continuation of the #6196 effort of optimizing CompoundQuery to merge components that have OUTER-JOIN. In 6196a there is a run-time support for OUTER-JOIN in PreselectQuery (more specifically in AdaptiveQuery). However, we can extend the solution to cover multi-table PreselectQuery with OUTER-JOIN right from the conversion side.

Goal: change the conversion rules such that preselect multi-table queries with outer-join will convert into PreselectQuery, not CompoundQuery.

Andrei, please address this issue after #6196 is completed. AFAIK, there is some attempt already in 6582a (which is really old, but can provide some insight) - refer to #6582.

#2 - 08/08/2023 05:32 AM - Igor Skornyakov

Please note that the conversion of the CompoundQuery to the PreselectQuery should not be applied if one of the components uses CONTAINS with non-constant search expression.
See #7086 (hopefully will be merged to the trunk soon).

#3 - 08/08/2023 07:03 AM - Alexandru Lungu

Andrei, please address #7667-2. Also, please make some tests with CAN-FIND for that matter.

#5 - 09/22/2023 09:35 AM - Andrei Bălteanu

I started by looking first at the part of this task related to CONTAINS and run-time optimization of multiple CompoundComponents into a PreselectQuery.
In order to block this optimization, I think we can add a condition in Optimizer.isServerJoinPossible() that will check if we have a CONTAINS in the where clause (if so, we automatically return false).
At the time of this method call, the FQL attached to a component is not yet computed.
As a trivial potential implementation for this, I added this:

```
=== modified file 'src/com/goldencode/p2j/persist/CompoundQuery.java'
--- old/src/com/goldencode/p2j/persist/CompoundQuery.java    2023-08-28 14:07:19 +0000
+++ new/src/com/goldencode/p2j/persist/CompoundQuery.java    2023-09-22 13:30:18 +0000
@@ -3528,8 +3528,17 @@
```

```
            return false;
         }

-        String where = query.getOriginalWhere();
-
+        String where = query.getOriginalWhere();
+        if (where.contains("contains"))
+        {
+           if (fine)
+           {
+              log.fine(" --- can't optimize a query with 'contains': ");
+           }
+
+           return false;
+        }
+
         if (fine)
         {
            StringBuilder buf = new StringBuilder(top ? "*" : " ");
```

The problem encountered by Roger in #7683-5 seems to be solved.
Of course, this is just a quick fix and a more elegant approach will be needed.

#6 - 09/22/2023 10:46 AM - Igor Skornyakov

In what branch can I find these changes?
Thank you.

#7 - 09/22/2023 11:18 AM - Igor Skornyakov

I've looked at the CompoundQuery in the trunk.

For me this check for CONTAINS looks oversimplified:

1. The where string can contain 'contains' substring as e.g. part of the string literal.
2. In theory is is possible to convert a CompoundQuery with CONTAINS to the PreselectQuery if the search expression of CONTAINS is a constant (does not depend on any other table used in the query).
3. Are you absolutely sure that if where string really uses CONTAINS operator it will contain 'contains' string in lower case?

I would rather perform the check for the CONTAINS operation at the HQL level. In fact the Compound query that cannot be converted to the PreselectQuery because of the CONTAINS will have "contains_db_expr" annotation added in the scope of #7086.

#8 - 09/25/2023 04:31 AM - Andrei Bălteanu

Igor, I think I led you into a small misunderstanding.
The comment from #7667-5 was related to a run-time optimization from a CompoundQuery to an Adaptive/PreselectQuery and it is **not** conversion related.
The case that I'm currently working on is a for each that is converted into a CompoundQuery and further optimized (at run-time) into an AdaptiveQuery.

#7667-5 was related to a regression caused by #6196, that appears when the CompoundQuery.Optimizer "succeeds" in doing it's job in cases where a CONTAINS keyword appears in the FQL. The presence of CONTAINS should always stop the optimization.

For that, I added the boolean hasContains in FQLPreprocessor and the method hasContainsKeyword to retrieve the value of it. The value of hasContains is changed only in FQLPreprocessor.fixEmptyContains.
The problem Roger encountered in #7683-5 seems to be solved.

Committed on 7667a revision 14752.

#9 - 09/25/2023 08:51 AM - Andrei Bălteanu

While trying to create a testcase that would help me solve this initial purpose of the task (changing the conversion rules such that multi-table queries with outer-join will convert into PreselectQuery instead of not CompoundQuery), I discovered a regression caused by #6196. This regression can be observed when we look at this testcase:

```
output to "resultFWD.txt".
define temp-table t1 field f1 as integer index index1 is primary unique f1.
define temp-table t2 field f2 as integer index index2 is primary unique f2.

create t1. t1.f1 = 10.
create t1. t1.f1 = -15.

create t2. t2.f2 = -15.
create t2. t2.f2 = 20.

define query q0 for t1, t2.
open query q0
    preselect
        each t1,
        each t2 outer-join where t1.f1 eq t2.f2.

get next q0.
message t1.f1.
message t2.f2.
message "--------".

get next q0.
message t1.f1.
message t2.f2.
message "--------".
```

The result in 4GL is:

```
-15
-15
--------
10
?
--------
```

The result in FWD is:

```
-15
-15
--------
-15
-15
--------
```

I think the problem is in PreselectQuery.load().
Inside this method, we iterate trough the row ids of the results. For the second next, the temp-table t2 has no value that matches the join clause resulting in the row id to be null. That will cause this if to be hit:

```
if (id == null)
{
    isNullId = true;
    break;
}
```

Because isNullId is TRUE, a call to PreselectQuery.setEmptyBuffersUnknown() will be done. Inside this method, we iterate trough the original components (which in this case are 2) and check if the buffer associated is null.
Since that is not the case, because each component already has a buffer that contains the previous result (-15 for both), nothing will be done, resulting in the buffer not to change.

#10 - 09/25/2023 09:04 AM - Alexandru Lungu

Review of 7667a changes:

- Please limit this only to outer-join cases. I think it is right to join two tables if there is an contains clause as long as they are not **outer**. With your changes, the optimization will always be ruled out.
- Please rework the hasContains flag into a hasNonConstantContains to handle only the cases of contains with dynamic parameter. You should do some extra steps in fixEmptyContains to set this flag **only** if the one of its parameter is non-constant. For this matter, you need eventually to build an auxiliary static method FQLPreprocessor.isConstant that is able to detect if a FQL AST is a constant expression (a.k.a doesn't have a database reference). Basically, it should return false if it ever encounters PROPERTY type.

#11 - 09/26/2023 04:46 AM - Andrei Bălteanu

Thanks for the review!
Modified accordingly and committed to 7667a revision 14753.

#12 - 09/26/2023 06:07 PM - Roger Borrello

Branch 7667a has been rebased to the latest trunk (14754).

#13 - 09/28/2023 04:52 AM - Andrei Bălteanu

Regarding the problem from #7667-9, I was thinking about the following changes:

```
=== modified file 'src/com/goldencode/p2j/persist/PreselectQuery.java'
--- old/src/com/goldencode/p2j/persist/PreselectQuery.java    2023-08-30 14:49:03 +0000
+++ new/src/com/goldencode/p2j/persist/PreselectQuery.java    2023-09-28 08:31:41 +0000
@@ -2994,12 +2994,22 @@
       try
       {
          boolean isNullId = false;
-         for (Long id : ids)
+         for (int i = 0; i < len; i++)
          {
-            if (id == null)
+            if (ids[i] == null)
             {
-               isNullId = true;
-               break;
+               QueryComponent comp = components.get(i);
+
+               if (comp.isOuter() && i > 0)
+               {
+                  RecordBuffer buffer = comp.getBuffer();
+                  buffer.setUnknownMode();
+               }
+               else
+               {
+                  isNullId = true;
+                  break;
+               }
             }
          }
```

Len is a variable that stores the number of original components before optimization. We iterate through rowIds of the results.
If any are null, we check if the component is **outer** and **not first**. For those components we set the buffer to the unknownMode.
Retested the simple testcase from #7667-9 and the output in FWD matches the one in 4GL.

#14 - 09/28/2023 05:44 AM - Alexandru Lungu

Review of 7667a:

- Fix history entry of CompoundQuery (AB is on a different line than the entry description)
- Recheck spacing (a return has align at 4 spaces instead of 3) - no need for extra history entries

I am functionally OK with the changes, so once these points are fixed, I will wait for Igor's review co continue with the merge.

#15 - 09/28/2023 06:30 AM - Andrei Bălteanu

- % Done changed from 0 to 20

Modified accordingly.
There was also a parameter out of line and i fixed that too (cheers to Radu for the heads up).
Committed on 7667a revision 14757.

#16 - 09/28/2023 08:27 AM - Alexandru Lungu

Note that this task is about conversion support, so 7667a is not going to address the task, but only fix a run-time regression required by this task later on.

I am OK with the changes now.
Igor, please review.
Greg, let me know if we can merge this after the review.

#17 - 09/28/2023 08:40 AM - Igor Skornyakov

Alexandru Lungu wrote:

> Note that this task is about conversion support, so 7667a is not going to address the task, but only fix a run-time regression required by this task later on.
>
> I am OK with the changes now.
> Igor, please review.

Alexandru,
I do not think that I'm a right person to review this change since I do not undersrand the details of the PreselectQuery logic good enough.
I believe that Ovidiu or Eric can do it better.

#18 - 09/28/2023 09:26 AM - Eric Faulhaber

- Status changed from WIP to Review

I'm having a look...

#19 - 09/28/2023 10:49 AM - Eric Faulhaber

Code review 7667a/14757:

The logic looks good. A few minor issues:

- Please avoid using the for-each syntax with an ArrayList (or array) in potentially performance-sensitive code. Use int len = list.size(); for (int i = 0; i < len; i++) { Foo foo = list.get(i); ... } instead. The former creates unnecessary objects under the covers, and is slightly slower.
- In CompoundQuery$Optimizer.isServerJoinPossible in the comments starting at line 3667, please insert a space between the double-slash and the start of each comment. Please correct "non-contans" to "non-constant" in the comments and logging messages.
- The javadoc for FQLPreprocessor.hasContains is a bit confusing. It references the contains keyword, but that is not checked here; the fact that the method (currently) is only called when processing a contains operator is only known by the caller. Please use the term "node" instead of "root". Root has a very specific meaning: it is the single ancestor node of the entire AST, not the parent of a particular branch.

BTW, I know this next item is not part of this update, but the edit in FQLP.fixEmptyContains made me aware of it: we really should not be doing a separate, dedicated walk of the entire AST for fixEmptyContains (unless it is unavoidable, due to the change it makes to the tree). There is a cost to the iteration of nodes, and the contains keyword is relatively rarely used. A missing operand for contains is even more rare. So, we are doing an extra walk of the entire tree for every where clause, to detect/correct a missing contains operand, when this is likely to not occur in 99.999...% of cases. If this adjustment can be integrated into mainWalk instead, it would be more efficient. I do not want to hold up the merge for this, since it is pre-existing, but please queue this for attention afterward.

#20 - 09/28/2023 11:05 AM - Eric Faulhaber

Andrei Bălteanu wrote:

> Regarding the problem from #7667-9, I was thinking about the following changes:
> [...]
> Len is a variable that stores the number of original components before optimization. We iterate through rowIds of the results.
> If any are null, we check if the component is **outer** and **not first**. For those components we set the buffer to the unknownMode.
> Retested the simple testcase from #7667-9 and the output in FWD matches the one in 4GL.

This change looks ok to me, but this is code which is executed in a lot of different scenarios, so regression testing is warranted.

#21 - 09/28/2023 05:12 PM - Roger Borrello

Branch 7667 has been rebased to trunk_14756.

#22 - 09/29/2023 02:35 AM - Alexandru Lungu

Andrei, please address #7667-19 asap (including the move of your code in mainWalk to avoid extra AST traversals). As these are (theoretically)

minor, I will do a quick review after and merge.

#23 - 09/29/2023 04:57 AM - Andrei Bălteanu

Sure thing!
Committed the changes on 7667a revision 14760.

#24 - 09/29/2023 06:59 AM - Alexandru Lungu

Andrei, I am OK with the changes. Still, isConstant is not "integrated" into the mainWalk - there are still two walks done in parallel. This should be fixed, but I don't have quite an idea how you can isolate.

> There is a cost to the iteration of nodes, and the contains keyword is relatively rarely used. A missing operand for contains is even more rare. So, we are doing an extra walk of the entire tree for every where clause, to detect/correct a missing contains operand, when this is likely to not occur in 99.999...% of cases.

Creating a separate topic #7855.

I am ready for merge of 7667a.

#25 - 09/29/2023 09:46 AM - Greg Shah

You can merge 7667a now.

#26 - 09/29/2023 09:51 AM - Alexandru Lungu

Branch 7667a was merged to trunk rev. 14757 and archived.

#27 - 09/29/2023 09:58 AM - Greg Shah

What is the % Done on this task?

#28 - 09/29/2023 10:06 AM - Andrei Bălteanu

- Status changed from Review to WIP

- % Done changed from 20 to 30

The branch 7667a was created to solve a problem related to a **run-time** optimization. This was not the initial purpose of this task, it was a regression that was discovered in the process.
The purpose of this task is to change the conversion rules such that preselect multi-table queries with outer-join will convert into PreselectQuery, not CompoundQuery, at **conversion-time**.
I think we can move it at 30%, because there is still work to do.

#29 - 10/02/2023 04:40 AM - Andrei Bălteanu

I retested the testcase from #6196-47, but instead of for each I used preselect each.
Without the changes from #7667-13 there are multiple places where the output differs from the one in 4GL.
Using the patch, the output in FWD is the same one as the one from 4GL.
Committed the changes to 7667b revision 14752.

#30 - 10/26/2023 10:15 AM - Alexandru Lungu

I just checked-out 7667b and changes are OK (#7667-13).

I suggest getting this into trunk when possible; this was an obvious regression that got fixed in 7667b. The sooner we merge this, the better. I will queue it in my testing routine. Keeping you updated with the testing.

#31 - 10/30/2023 08:51 AM - Greg Shah

Has enough testing been done to confirm that 7667b is safe?

Are any other code reviews needed?

#33 - 01/11/2024 05:51 AM - Alexandru Lungu

Branch 7667b was merged to trunk revision 14916 and archived.