Database - Bug #3647

incorrect hql and incorrect sql generated for nested CAN-FIND

07/03/2018 03:37 PM - Ovidiu Maxiniuc

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Ovidiu Maxiniuc	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 07/03/2018 03:54 PM - Ovidiu Maxiniuc

- Status changed from New to WIP

- Assignee set to Ovidiu Maxiniuc

I investigated an issue found by Constantin in client code and reported in #3567. I isolated is as the following testcase:

```
DEFINE TEMP-TABLE tt1
   FIELD f1 AS INT.

DEFINE TEMP-TABLE tt2
   FIELD f1 AS INT
   FIELD g1 AS INT.

DEFINE TEMP-TABLE tt3
   FIELD f1 AS INT
   FIELD g1 AS INT
   FIELD h1 AS INT.

FIND tt1
WHERE CAN-FIND(tt2
               WHERE tt2.f1 = tt1.f1 AND
                     CAN-FIND(FIRST tt3
                              WHERE tt3.f1 = tt1.f1 AND
                                    tt3.g1 = tt2.g1 AND
                                    tt3.h1 = 0))
NO-LOCK NO-ERROR.
MESSAGE "found 0".
```

This is incorrectly converted as

```
silent(() -> new FindQuery(tt1,
                           "(select count(tt2.id) from Tt2_1_1 as tt2 " +
                            "where tt2.f1 = tt1.f1 and " +
                                  "exists(from Tt3_1_1 as tt3 " +
                                         "where tt3.f1 = tt1.f1 and " +
                                               "tt3.g1 = ? and " +      /* X */
                                               "tt3.h1 = 0)" +
                           ") = 1",
                           null,
                           "tt1.id asc",
                           new Object[] { (P2JQuery.Parameter) () -> tt2.getG1() },  /* Y */
                           LockType.NONE
).setExternalBuffers(tt3, tt2).unique());
```

Normally, the line marked with X should read "tt3.g1 = tt2.g1 and " instead and the Y line should be dropped since the scope of buffer tt2 is only inside de CAN-FIND block.
However, there is a second issue here: if the code is correctly converted or manually fixed, the HQL Preprocessor (or the H2 driver - I am still investigating the culprit) does a bad job so the SQL sent to server looks like this:

```
from Tt1_1_1Impl as tt1
where (tt1._multiplex = ?0) and
      ((select count(tt2_1.id)
        from Tt2_1_1Impl as tt2_1
        where checkError(initError(false),
                         (tt2_1.f1 = tt1.f1 or tt2_1.f1 is null and tt1.f1 is null) and
                         exists(from Tt3_1_1Impl as tt3_1
                                where (tt3_1.f1 = tt1.f1 or tt3_1.f1 is null and tt1.f1 is null) and
                                      (tt3_1.g1 = tt2.g1 or tt3_1.g1 is null and tt2.g1 is null) and
                                      tt3_1.h1 = 0
                               )
                        )
       ) = 1
      )
```

Notice the from clause for tt2: from Tt2_1_1Impl as tt2_1. At the outer where level the buffer is correctly used tt2_1.f1, but in the inner where predicate, the field is used with an unknown buffer name: tt2.g1.

#2 - 07/03/2018 03:58 PM - Ovidiu Maxiniuc

The H2 driver is clean. The logged statement is HQL not SQL. So, most likely, the second problem is in HQLPreprocessor.

#3 - 07/04/2018 04:02 PM - Ovidiu Maxiniuc

I discovered that in the case of nesting CAN-FIND scopes the HQLPreprocessor was not looking in the outer scope for buffer names so the buffer name tt2 was not converted to correct alias (tt2_1).
Committed to 3600b as rev 11277.

I also investigated the relation to #3315. They are not directly related. In this case the substitution is incorrectly generated for buffers from the same database while in #3315 is about a combination of buffers from different databases. I checked the testcase from that tracker and it seems to be correctly converted at this moment.

#4 - 07/04/2018 05:23 PM - Ovidiu Maxiniuc

I think I am about to fix the generation of the nested CAN-FIND queries for same-database case. I noticed that the queryBufferOrder is used in index_selection.rules to detect the related buffers as a side effect of detecting the index. I think this is incorrect at least for the case of CAN-FIND nodes that have a record_phrase as ancestor. These nodes are ignored (not added to queryBufferOrder since they don't count for ORDER BY) so that they are not considered to be related buffers. As result, the field (tt2.g1 in the case from note-1) is extracted as a substitution when this is not the case (Y line).

#5 - 07/05/2018 02:47 PM - Ovidiu Maxiniuc

I tried to use a parallel structure for my discovery but in the end it seems to me that the semantic is the same: ie in the end, the same nodes are annotated with related_buffer=true as when I comment out the !ancestor(prog.record_phrase, -1)) (line 386) in index_selection.rules. However, this is not mandatory a bad thing, the generated code for my testcases is not affected. Yet, I saw that the line was added on purpose so I still have some doubts.

There is another issue I am investigating and still don't know how to tackle. Consider the following testcase:

```
FIND book
WHERE book.book-id NE 0 AND
      CAN-FIND(tt2
               WHERE tt2.f1 = book.book-id AND
                     CAN-FIND(FIRST tt3
                              WHERE tt3.g1 = tt2.g1))
NO-LOCK.
```

It should generate something like:

```
new FindQuery(book,
              "book.bookId != 0",
              () -> new FindQuery(tt2,
                                  "tt2.f1 = ? and exists(from Tt3_1_1 as tt3 where tt3.g1 = tt2.g1)",
                                  null,
                                  "tt2.id asc",
                                  new Object[] { book.getBookId() },
                                  LockType.NONE).setExternalBuffers(tt3).hasOne(),
              "book.bookId asc").unique();
```

This is hand written code so please let me know if you agree on its correctness. The generated code is really bad but for moment I interested in tt2.g1. I investigated the .ast and the inner HQL query looks like tt2.f1 = ? and exists(from Tt3_1_1 as tt3 where tt3.g1 = ?), tt2.g1 being incorrectly pulled out as a substitution as it is annotated with hql=false. The code that does the same for book.book-id (this time correctly) is in where_clause_prep3.rules, lines 307-317:

```
      <rule>evalLib("fields")                                                        and
            getNoteBoolean("hql")                                                    and
            ancestor(prog.kw_where, -1)                                              and
            copy.getAncestor(-1, prog.record_phrase, "client_can_find", true) != null  and
            !getNoteBoolean("current_buffer")                                        and
            getNoteBoolean("related_buffer")                                         and
            (!isNote("sub_expression") or !getNoteBoolean("sub_expression"))

         <action>putNote("sub_expression", true)</action>
         <action>putNote("hql", false)</action>
      </rule>
```

#6 - 07/05/2018 03:03 PM - Ovidiu Maxiniuc

The code from where_clause_prep3.rules, lines 307-317 tries to fix (or maybe better said to refine) the related_buffer relation. The book buffer is *related* to outer, server-side query but not for the inner query which is is a client-side where, but also executed on server-side.

OTOH, there should no restrictions for tt2 which is used in used the 2nd inner query and the related_buffer relation is not affected with tt3 that is its buffer.

#7 - 07/06/2018 07:03 PM - Ovidiu Maxiniuc

I continued investigations with following results:

1. In the note above, the TRPL code does not check the databases for the buffers. We should negate the hql annotation only if the databases do not match. If the database is the same then the can-find function will be encoded as a subquery which will be executed on in a single sql query.

2. The current trunk revision will convert the ABL find statement from note-5 into:

```
new FindQuery(book, "book.bookId != 0 and", () -> isEqual(tt2.getF1(), book.getBookId()), "book.bookId asc"
, LockType.NONE).setExternalBuffers(tt3).unique();
```

I was hoping that the change from point 1. will fix this too, but I was wrong. I debugged the TRPL starting from the extra and from HQL and it lead me to the fact that the tree is not properly annotated with client_branch. Going deeper I realized that only first child of the AND is the nested CAN-FIND is not because tt2 and tt3 belongs to same database (well, this is exactly the issue from 1. above). As result the outer CAN-FIND is not fully extracted as client node - the client_branch annotation goes up an AND node only if both siblings are deemed to be client. But since the innermost CAN-FIND (tt3) is detected that can be executed on "same server" as tt2, the marking process is halted. I am trying now to find some conditions to clarify this case.

#8 - 07/20/2018 03:51 AM - Eric Faulhaber

Task branch 3600b was merged to trunk revision 11273, which included a fix referencing this issue. Can this be closed now, or is there more work to do?

#9 - 07/20/2018 09:30 AM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Task branch 3600b was merged to trunk revision 11273, which included a fix referencing this issue. Can this be closed now, or is there more work to do?

Not yet.
I committed only the runtime part as I considered it a low-risk. The conversion, on the other hand, has not yet been committed to 3600b because it has some unwanted side-effects which may (don't know for sure) prevent compiling some queries that were compiling even if they were incorrectly generated.