Database - Feature #3296

misc db features

04/27/2017 03:25 PM - Greg Shah

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

Description

Related issues:

Related to Database - Bug #3364: problems with TABLE-HANDLE and TABLE paramet...	**New**

History

#1 - 04/27/2017 03:26 PM - Greg Shah

Implement support for #3257-35, #3257-54, #3257-67.

#2 - 08/25/2017 01:37 PM - Greg Shah

There is a use of a table parameter with BY-REFERENCE. Is this handled already by conversion?

#3 - 09/29/2017 02:22 PM - Greg Shah

Eric reports that the BY-REFERENCE support for conversion and runtime does need to be added.

#4 - 09/29/2017 02:23 PM - Greg Shah

Both #3257-35 and #3257-54 have been found to be not needed. These can be ignored.

#5 - 09/29/2017 02:26 PM - Greg Shah

The following is needed:

- ~~#3257-67 possible table parameter issue documented in #2000-74 which would convert and compile but not work properly; it is not known if this is present or not, but it needs to be checked~~
- ~~#3257-19 DBPARAM single user mode; ECF will check with the customer to see if this is really needed; it has implications on connect and general multi-user server support~~
- #3257-33 triple-nested CAN-FINDs; this is an issue to resolve (these aren't converting properly)
- ~~#3257-36 NUM-RESULTS query attribute (used in 4 places), the runtime support is missing~~
- ~~BY-REFERENCE usage (conversion and runtime)~~

#6 - 10/09/2017 09:12 AM - Greg Shah

Eric checked the table parameter issue (#2000-74) and it no longer exists in the latest FWD project. That item is now closed.

#7 - 10/09/2017 09:14 AM - Greg Shah

Eric created branch 3296a.

That that branch he has completed the NUM-RESULTS implementation and put a minimal single-user mode implementation for DBPARAM. The

customer confirmed that actual single-user mode is not needed.

#8 - 10/30/2017 02:38 PM - Eric Faulhaber

Code review 3296a/11191:

Changes look good. I made a minor change to the TemporaryBuffer file header (the last two comments are part of the same merge).

#9 - 10/30/2017 03:11 PM - Constantin Asofiei

3296a 11193 fixes TABLE-HANDLE and TABLE parameters at function call. What remains is more complex to fix:

1. runtime for TABLE-HANDLE ... BY-REFERENCE - I don't have testcases yet, but code in TemporaryBuffer.createDynamicTable should be similar to TemporaryBuffer.associate
2. conversion for BY-REFERENCE and APPEND for TABLE-HANDLE and TABLE arguments at function calls: this is a parser issue too, as BY-REFERENCE and APPEND are emitted as a sibling to i.e. TABLE-HANDLE (as a direct child for the function call AST, i.e. FUNC_CHAR), and the conversion rules assume that BY-REFERENCE or APPEND are arguments for the function call, and not options. See uast/temp_table_by_ref_c.p which covers all the function/procedure cases for TABLE-HANDLE and TABLE, with BY-REFERENCE and/or APPEND. Just uncomment the commented code to see the error.

Other tests are these:

```
added uast/temp_table_by_ref.p
added uast/temp_table_by_ref2.p
added uast/temp_table_by_ref3.p
added uast/temp_table_by_ref_b.p
added uast/temp_table_by_ref_c.p
```

#10 - 10/31/2017 12:01 AM - Eric Faulhaber

- Status changed from New to WIP

Rebased task branch 3296a to trunk rev 11190. Current task branch revision is 11203.

#11 - 10/31/2017 08:42 AM - Constantin Asofiei

3296a rev 11204 (added a missing history entry) was merged to trunk rev 11191 and archived.

#12 - 11/01/2017 09:37 AM - Greg Shah

- Related to Bug #3364: problems with TABLE-HANDLE and TABLE parameters related to BY-REFERENCE added

#13 - 11/02/2017 10:38 AM - Eric Faulhaber

Task branch 3296b was created from trunk revision 11193. However, before adding any task-specific changes, that branch was used to fix a classpath issue and a data import regression, and was merged to trunk as revision 11194. The branch was archived, and the team was notified.

Task branch 3296c was created from trunk revision 11194.

#14 - 01/11/2018 05:40 PM - Eric Faulhaber

Test case for triple-nested CAN-FIND issue nested-canfind-3-deep.p:

```
define temp-table tt1
       field f1 as int
       field f2 as log.

define var flag as log.

FOR each person
   where person.emp-num > 0
     AND CAN-FIND(FIRST pers-addr WHERE
         pers-addr.emp-num = person.emp-num AND
         pers-addr.site-id = person.site-id AND
         (if flag then can-find(first tt1 where tt1.f1 eq pers-addr.site-id
          AND tt1.f2 no-lock) else true))
   USE-INDEX name
   no-lock:

end.

find first person no-lock.

if can-find(first pers-addr where
   pers-addr.emp-num = person.emp-num and
   pers-addr.site-id = person.site-id and
   (if flag then can-find(first tt1 where tt1.f1 eq pers-addr.site-id
    and tt1.f2 no-lock) else true))
then
   message "can-found".
```

Converts incorrectly to:

```
package com.goldencode.testcases;

import com.goldencode.p2j.util.*;
import com.goldencode.p2j.persist.*;
import com.goldencode.testcases.dmo._temp.*;
import com.goldencode.testcases.dmo.p2j_test.*;
import com.goldencode.p2j.ui.*;
import com.goldencode.p2j.persist.lock.*;

import static com.goldencode.p2j.util.BlockManager.*;
import static com.goldencode.p2j.util.CompareOps.*;
import static com.goldencode.p2j.util.logical.*;
import static com.goldencode.p2j.ui.LogicalTerminal.*;

/**
 * Business logic (converted to Java from the 4GL source code
 * in nested-canfind-3-deep.p).
 */
public class NestedCanfind3Deep
{
   Tt1_1_1.Buf tt1 = SharedVariableManager.addTempTable("tt1", Tt1_1_1.Buf.class, "tt1", "tt1");

   PersAddr.Buf persAddr = RecordBuffer.define(PersAddr.Buf.class, "p2j_test", "persAddr", "pers-addr");

   Person.Buf person = RecordBuffer.define(Person.Buf.class, "p2j_test", "person", "person");

   /**
    * External procedure (converted to Java from the 4GL source code
    * in nested-canfind-3-deep.p).
    */
   public void execute()
   {
      logical flag = UndoableFactory.logical();

      externalProcedure(NestedCanfind3Deep.this, new Block((Body) () ->
      {
         RecordBuffer.openScope(tt1, person, persAddr);
```

```
         AdaptiveQuery query0 = new AdaptiveQuery();

         forEach("loopLabel0", new Block((Init) () -> 
         {
            query0.initialize(person, "person.empNum > 0 and exists(from PersAddr as persAddr where persAddr.e
mpNum = person.empNum and persAddr.siteId = person.siteId and case when ? then  else true end)", () -> (flag).
booleanValue() ? new FindQuery(tt1, "tt1.f1 = ? and tt1.f2 = true", null, "tt1.id asc", new Object[]
            {
               persAddr.getSiteId()
            }, LockType.NONE).hasAny() : new logical(true), "person.lastName asc, person.firstName asc, person
.middleInit asc");
            query0.setExternalBuffers(persAddr);
         },
         (Body) () -> 
         {
            query0.next();
         }));

         new FindQuery(person, (String) null, null, "person.siteId asc, person.empNum asc", LockType.NONE).fir
st();

         if ((new FindQuery(persAddr, "persAddr.empNum = ? and persAddr.siteId = ? and case when ? then  else
true end", () -> and(isEqual(tt1.getF1(), persAddr.getSiteId()), () -> isEqual(tt1.isF2(), true)), "persAddr.s
iteId asc, persAddr.empNum asc", new Object[]
         {
            (P2JQuery.Parameter) () -> person.getEmpNum(),
            (P2JQuery.Parameter) () -> person.getSiteId(),
            flag
         }, LockType.NONE).hasAny()).booleanValue())
         {
            message("can-found");
         }
      }));
   }
}
```

#16 - 02/12/2018 08:19 AM - Ovidiu Maxiniuc

With 3444a/11222 the 2nd query is converted as:

```
if ((new FindQuery(persAddr,
                   "persAddr.empNum = ? and persAddr.siteId = ?",
                   () -> (flag).booleanValue() ?
                         new FindQuery(tt1,
                                       "tt1.f1 = ? and tt1.f2 = true",
                                       null,
                                       "tt1.id asc",
                                       new Object[] { persAddr.getSiteId() },
                                       LockType.NONE).hasAny() :
                        new logical(true),
                   "persAddr.id asc",
                   new Object[]
                   {
                      (P2JQuery.Parameter) () -> person.getEmpNum(),
                      (P2JQuery.Parameter) () -> person.getSiteId()
                   }).hasAny()).booleanValue())
{
  message("can-found");
}
```

The first is unchanged.

#17 - 02/13/2018 04:22 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> With 3444a/11222 the 2nd query is converted as:
> [...]

That looks correct to me. Do you see any issues with it?

#18 - 02/13/2018 04:52 PM - Eric Faulhaber

I think the first query should convert in a similar way; that is, the permanent tables as HQL (in this case with a sub-select) and the temp-table as a client-side FindQuery. I guess what's happening is that the inner-most CAN-FIND sub-expression is being annotated to convert to a client-side expression, but still has a conflicting annotation of hql = true, so it's emitting both ways, sort of. I think the correct way is something like:

```
forEach("loopLabel0", new Block((Init) () ->
{
   query0.initialize(person,
                     "person.empNum > 0 and exists(from PersAddr as persAddr where persAddr.empNum = person.em
pNum and persAddr.siteId = person.siteId)",
                     () -> (flag).booleanValue()
                           ? new FindQuery(tt1, "tt1.f1 = ? and tt1.f2 = true",
                                           null,
                                           "tt1.id asc",
                                           new Object[] { persAddr.getSiteId() },
                                           LockType.NONE).hasAny()
                           : new logical(true),
                     "person.lastName asc, person.firstName asc, person.middleInit asc");
   query0.setExternalBuffers(persAddr);
},
(Body) () ->
{
   query0.next();
}));
```

Please double check my logic.

Even this conversion is not ideal, because when flag = false, we would want to skip the client-side filtering altogether, since the match decision could be made entirely on the database server (the client-side expression would always return true in that case, accepting every record brought back for client-side filtering). However, it doesn't hurt us much in this case, precisely for the same reason; every record is accepted by the client-side filter, so we don't bring back more records than necessary. Anyway, for now it's not worth the extra complexity of refactoring it differently, so let's aim for this as a target.

At some point, we need to format the code like this, too -- a lot easier to read than the way it emits now.

#19 - 02/14/2018 03:56 AM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> That looks correct to me. Do you see any issues with it?

No. None.

#20 - 02/14/2018 04:31 AM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> I think the first query should convert in a similar way; that is, the permanent tables as HQL (in this case with a sub-select) and the temp-table as a client-side FindQuery. I guess what's happening is that the inner-most CAN-FIND sub-expression is being annotated to convert to a client-side expression, but still has a conflicting annotation of hql = true, so it's emitting both ways, sort of. I think the correct way is something like:
> [...]
> Please double check my logic.

Yes, apparently the and case ... must be dropped. I already investigated the hql=true annotation but without a clear result. I am now looking at some rules in annotations/where_clause.rules that read:

```
   ancestor(prog.kw_where, -1) and
   getNoteBoolean("hql")       and
   (!evalLib("is_client_branch") or evalLib("is_nested_can_find"))
```

The nodes that and case ... are annotated as client_branch but because of nested_can_find they are forced to be emitted in HQL. I was trying to adjust is_nested_can_find or refine the whole rule (occurs 3 times in same file).

Yet I think your suggestion for the code is NOT correct. The client-where will not be linked to proper point. If we rewrite it back we would have the equivalent of:

```
person.empNum > 0 and
exists(from PersAddr as persAddr where persAddr.empNum = person.empNum and persAddr.siteId = person.siteId) an
d
case when (flag) then (query) else true end.
```

which is not the original query. Instead, the if client-where should be part of inner exist. A better code should look like:

```
query0.initialize(person,
                  "person.empNum > 0 and " +
                  "exists(from PersAddr as persAddr where " +
                                          "persAddr.empNum = person.empNum and " +
                                          "persAddr.siteId = person.siteId and " +
                                          "case when ? then ? else true end))",
                  null,
                  "person.lastName asc, person.firstName asc, person.middleInit asc",
                  new Object[]
                  {
                     flag,
                     new FindQuery(tt1,
                                   "tt1.f1 = ? and tt1.f2 = true",
                                   null,
                                   "tt1.id asc",
                                   new Object[] { persAddr.getSiteId() },
                                   LockType.NONE).hasAny()
                  });
```

> Even this conversion is not ideal, because when flag = false, we would want to skip the client-side filtering altogether, since the match decision could be made entirely on the database server (the client-side expression would always return true in that case, accepting every record brought back for client-side filtering). However, it doesn't hurt us much in this case, precisely for the same reason; every record is accepted by the client-side filter, so we don't bring back more records than necessary. Anyway, for now it's not worth the extra complexity of refactoring it differently, so let's aim for this as a target.

I agree.

> At some point, we need to format the code like this, too -- a lot easier to read than the way it emits now.

True, this is the reason I failed to see the failure yesterday.
Offtopic: An intelligent chop-down feature for all long lines would be welcomed (chop at , in argument lists, split (hql) string after "and", yet keep on one line if anonymous array c'tor has only one element).

#21 - 02/14/2018 12:13 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> Yet I think your suggestion for the code is NOT correct. The client-where will not be linked to proper point. If we rewrite it back we would have the equivalent of:
> [...]

I agree that it is not correct, but not because the client-side query is not linked to the proper point. It won't work because there is no persAddr buffer in the business logic on which the client-side FindQuery can operate. The pers_addr table is only touched on the database server in a sub-select; there is no proxy that would be populated in business logic for the FindQuery to reference for its query substitution parameter referenced by persAddr.getSiteId().

> which is not the original query. Instead, the if client-where should be part of inner exist. A better code should look like:
> [...]

No, this won't work either. The FindQuery can't be treated as a query substitution parameter of the person query. By their nature, query substitution parameters are resolved once, *before* the query is executed. We need to get the pers_addr.site_id value for every candidate pers_addr record evaluated in the sub-select, which of course changes during query execution on the server. Plus, we have the same lack of persAddr buffer in the business logic.

We need to think on this some more...

#22 - 03/16/2018 04:29 PM - Eric Faulhaber

I think what has to happen here is that any nested CAN-FIND which has to be executed on the client (i.e., application server) and which references a related buffer in a containing CAN-FIND needs to pull all such containing CAN-FINDs with related buffers into the client-side expression. Those containing CAN-FINDs that might otherwise be converted to subselects must be part of the expression that converts to the client-side lambda, because the client-side CAN-FIND where clause referencing those related buffers must access the buffers to evaluate its where clause expression.

While logically correct, this will result in a pretty inefficient query in this case. We should consider how we might refactor the original 4GL (even if it needs to be done manually) to get a better result.

If, OTOH, nothing forces any part of the query to be evaluated as a client-side expression (i.e., all tables reside in the same database) OR the client-side, nested CAN-FIND does not reference any related buffer in an enclosing CAN-FIND, I think the entire where clause can be converted to a server-side expression with nested subselects.

#23 - 04/05/2018 09:32 PM - Eric Faulhaber

- Assignee changed from Eric Faulhaber to Ovidiu Maxiniuc

#25 - 04/06/2018 01:15 PM - Ovidiu Maxiniuc

Last week I added some small fix that should correctly generate the nested CAN-FINDs in a single nested query if possible (the buffers belong to same database) but I did not full-scale test it and I don't know if there are unwanted side-effects.

At the same time, an idea crossed my mind. The big problem is when the buffers involved in the nested CAN-FIND constructs belongs to different databases. What if we get them together? Maybe you already thought about it but I wonder whether is it a not such bad idea to keep the temp-tables in same database as the permanent tables. I know that this only works for applications with only a single permanent database, but I estimate that 3 / 4 of the applications do not use more than that. There are also some downsides:

- performance is better with in-process H2 database;
- FWD conversion needs to know a-priori about this case and generate different HQL for these projects;
- possible extra effort for db maintenance (OTOH, it may be a plus to have the temp-table already created in database instead of creating them each time they are needed and dropped when nobody uses them anymore).

Yet, in the end, this is only an optimization for a subset of the applications since the solution for general case must be working correctly, regardless of the performance of constrained by the multiple database application architecture.