Database - Bug #7355

regression in row structure calculation for DATETIME-TZ when join is used on temp-tables

05/15/2023 12:37 PM - Constantin Asofiei

Status:	Test	**Start date:**	
Priority:	Urgent	**Due date:**	
Assignee:	Alexandru Lungu	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 05/15/2023 12:40 PM - Constantin Asofiei

- Priority changed from Normal to Urgent

I'm not sure when this was introduced (I suspect when tt.* or something like this). In FWD, DATETIME-TZ type uses 2 SQL columns. The problem can be seen when a join is used:

```
define temp-table tt1 field f0 as datetime-tz field f1 as character index ix1 is unique f1.
define temp-table tt2 field f1 as character index ix1 f1.

create tt1.
tt1.f0 = now.
tt1.f1 = "abc".
create tt2.
tt2.f1 = "abc".

for each tt1, each tt2 where tt1.f1 = tt2.f1:
  message tt1.f1 tt1.f0.
end.
```

The SQL is this:

```
select
    tt1_1_1__i0_.*, tt2_1_1__i1_.*
from
    tt1 tt1_1_1__i0_
cross join
    tt2 tt2_1_1__i1_
where
    tt1_1_1__i0_._multiplex = ? and tt2_1_1__i1_._multiplex = ? and ((tt1_1_1__i0_.f1 = tt2_1_1__i1_.f1 or tt1
_1_1__i0_.f1 is null and tt2_1_1__i1_.f1 is null))
order by
    tt1_1_1__i0_._multiplex asc, tt1_1_1__i0_.f1 asc nulls last, tt1_1_1__i0_._multiplex asc, tt2_1_1__i1_.rec
id asc
lazy
```

and for some reason the RowStructure for tt1 is of 'count' 10 instead of 11. This is from the ResultSet.expressions:

```
TT1_1_1__I0_.RECID
TT1_1_1__I0_._MULTIPLEX
TT1_1_1__I0_._ERRORFLAG
```

```
TT1_1_1__I0_._ORIGINROWID
TT1_1_1__I0_._DATASOURCEROWID
TT1_1_1__I0_._ERRORSTRING
TT1_1_1__I0_._PEERROWID
TT1_1_1__I0_._ROWSTATE
TT1_1_1__I0_.F0
TT1_1_1__I0_.F0_OFFSET
TT1_1_1__I0_.F1
TT2_1_1__I1_.RECID
TT2_1_1__I1_._MULTIPLEX
TT2_1_1__I1_._ERRORFLAG
TT2_1_1__I1_._ORIGINROWID
TT2_1_1__I1_._DATASOURCEROWID
TT2_1_1__I1_._ERRORSTRING
TT2_1_1__I1_._PEERROWID
TT2_1_1__I1_._ROWSTATE
TT2_1_1__I1_.F1
```

#2 - 05/15/2023 12:43 PM - Constantin Asofiei

Denormalized datetime-tz extent fields need to be checked, too.

#4 - 05/15/2023 08:32 PM - Ovidiu Maxiniuc

- Status changed from New to WIP

- % Done changed from 0 to 100

Your analysis is correct. Here is the patch for fixing the issue:

```
--- a/src/com/goldencode/p2j/persist/orm/FqlToSqlConverter.java
+++ b/src/com/goldencode/p2j/persist/orm/FqlToSqlConverter.java
@@ -56,8 +56,9 @@
 **                  dialect doesn't require computed columns (refs: #7108).
 **                  Fixed second pass query generation for queries with CONTAINS operator.
 ** 007 OM  20230428 Made sure [aliases] is balanced. Avoid double-emit or skipping of FQLAst tokens.
-*                  Fixed handling of nested SUBSELECTs. Local optimizations. Bit of code cleanup.
+**                  Fixed handling of nested SUBSELECTs. Local optimizations. Bit of code cleanup.
 ** 008 GBB 20230512 Logging methods replaced by CentralLogger/ConversionStatus.
+** 009 OM  20230516 Fixed RowStructure for datetimetz properties (they use two columns per property).
 */

 /*
@@ -1631,6 +1632,7 @@
    {
       int fieldCounter = 0;
       String sqlTableAliasName = getSqlTableAliasName(alias);
+      Set<Map.Entry<String, Property>> fieldEntries = dmoMeta.propsByName.entrySet();
       if (dialect.allowSelectWildcard() &&
           canUseWildcard(dmoMeta)       &&
           !NO_ALIAS.equals(sqlTableAliasName))
@@ -1641,6 +1643,13 @@
          }
          sb.append(sqlTableAliasName).append(".*");
          fieldCounter = dmoMeta.propsByName.size();
```

```
+          for (Map.Entry<String, Property> field : fieldEntries)
+          {
+             if (field.getValue()._isDatetimeTz)
+             {
+                fieldCounter++; // datetimetz use two columns per property
+             }
+          }

           if (rowStructure != null)
           {
@@ -1650,7 +1659,6 @@
           return;
        }

-       Set<Map.Entry<String, Property>> fieldEntries = dmoMeta.propsByName.entrySet();
        for (Map.Entry<String, Property> field : fieldEntries)
        {
           Property property = field.getValue();
```

#5 - 05/16/2023 02:23 AM - Alexandru Lungu

- File datetimetz_count.patch added

Nice catch, this makes sense. At some point canUseWildcard was denying the use of wildcard for datetime-tz fields. I removed this because there were lots of tables with such fields, but didn't handle the field count properly.

Ovidiu's approach is right, but I think we shall avoid iterating all fields when computing such query. I had bad performance before with tables that have ~300 fields (especially in for-each loops). I suggest computing the number of datetime-tz in DmoMeta only once.

Consider the attached patch instead.

#6 - 05/16/2023 03:44 AM - Constantin Asofiei

Alexandru, please create a 7355a branch and cache the total number of SQL fields in a table at DmoMeta (so not just datetime-tz field count). This will make things clear at FqlToSqlConverter and DmoMeta will be responsible of computing this value only once.

#7 - 05/16/2023 09:07 AM - Alexandru Lungu

- Status changed from WIP to Review

Created 7355a and committed rev. 14572. DmoMeta computes the total number of SQL fields only once and uses it in FqlToSqlConverter.

#8 - 05/16/2023 09:15 AM - Greg Shah

Ovidiu: Please review.

#9 - 05/16/2023 02:04 PM - Constantin Asofiei

Alexandru, there is PropertyReader.propertySize() which gives the number of SQL columns for a DMO property. Is it possible to use that?

#11 - 05/16/2023 03:38 PM - Ovidiu Maxiniuc

I am OK with the code from rev. 14572. It will clearly bring more performance than my initial solution. An alternative would be to compute this number in c'tor, when the properties are added to the map (line 406).

There is another RowStructure c'tor which must also be called with the right arguments:

```
--- src/com/goldencode/p2j/persist/orm/DirectAccessHelper.java
+++ src/com/goldencode/p2j/persist/orm/DirectAccessHelper.java
@@ -277,7 +277,7 @@
       ResultSet rs = response.getResultSet();
       if (rs.next())
       {
-         RowStructure rowStructure = new RowStructure(meta.dmoImplInterface, meta.getSqlFieldCount());
+         RowStructure rowStructure = new RowStructure(meta.dmoImplInterface, meta.propsByName.size());
          return (Record) SQLQuery.hydrateRecord(rs, rowStructure, 1, session);
       }
       else
```

Constantin, indeed using a feature like PropertyReader.propertySize() would make code more robust/generic allowing future expanding or other data types to 2 columns or even more (‡). But at this moment the value is not present in Property class and PropertyReader is only implemented in c.g.p.p.orm.types.*Types classes.

(‡) This gave me a possible idea for an implementation of the expansion of the extent fields, although things could get messy, with the accent in case of datetime-tz extents fields.

#12 - 05/17/2023 05:25 AM - Alexandru Lungu

If I am going to use this is the c'tor and use propertySize, I can do:

```
boolean normalized = property.extent > 0 && property.index == 0;
if (!normalized)
{
   Class<?> fwdType = property._fwdType;
   if (BaseDataType.class.isAssignableFrom(fwdType))
   {
      fieldsCount += TypeManager.getDataHandler((Class<? extends BaseDataType>) fwdType).propertySize();
   }
   else
   {
      // is this a good fall-back?
      fieldsCount++;
   }
}
```

In fact, I committed this to 7355a/rev. 14573 (including some javadoc changes). Tested this and didn't see any obvious issues.

#13 - 05/17/2023 06:00 AM - Constantin Asofiei

We need some tests with datetime-tz extent fields (I think is enough to have tt1.f0 as extent in the example). Test with both normalized and denormalized extents. If this works, then the fix is OK.

#14 - 05/17/2023 07:43 AM - Alexandru Lungu

I tested with a complex example using extents and datetime-tz, but only normalized (which was ok). I am trying to test the denormalized extents. AFAIK, there is a "denorm-extents" option in p2j.cfg.xml, but it is only about the generated DDL for persistent databases. For the DMO's part, these are the same for normalized and denormalized. However, I don't know exactly the option to trigger temporary tables to use denormalized extents. Can you help me on this? I thought a directory.xml options is used, but there is nothing with "denorm" or "extent".

#15 - 05/17/2023 05:09 PM - Ovidiu Maxiniuc

The denormalization of dynamic temp-tables is governed by persistence/denormalize-dynamic-temp-tables directory key (default is false).

The denormalization of static extent tables was disabled as part of #6561. To force it here you need to use hints. Create a <same-name-as-4gl-source>.hints file with the following content:

```
<?xml version="1.0"?>

<hints>
   <schema>
      <table name="[your-table-which-contains-extent-fields]">
         <custom-extent/>
      </table>
   </schema>
</hints>
```

Then just reconvert and you'll get the DMO using expanded fields.

#16 - 05/18/2023 09:02 PM - Ovidiu Maxiniuc

Alexandru,
I mistaken the 7355a with another branch and I have just rebased it to latest trunk. Since I am doing the operation rather blindly on devsrv01 I realized only when my intended branch refused to update. Sorry for any inconvenience.

#17 - 05/19/2023 02:21 AM - Alexandru Lungu

Ovidiu Maxiniuc wrote:

> Then just reconvert and you'll get the DMO using expanded fields.

Got it! The changes are OK. Tested with a 4-table join: first with datetime-tz extent, second with int extent, third with datetime-tz extent and last with no extents. The sql field count was computed correctly for each table. The result was as expected.

> I mistaken the 7355a with another branch and I have just rebased it to latest trunk. Since I am doing the operation rather blindly on devsrv01 I realized only when my intended branch refused to update. Sorry for any inconvenience.

This is good; I don't have to rebase it myself now :) As the tests are OK and customer applications are fine with the change, should I commit this to trunk?

#18 - 05/19/2023 02:22 AM - Constantin Asofiei

Alexandru Lungu wrote:

> This is good; I don't have to rebase it myself now :) As the tests are OK and customer applications are fine with the change, should I commit this to trunk?

Push it to 7156a for now, as a patch.

#19 - 05/19/2023 03:00 AM - Alexandru Lungu

- Status changed from Review to Test

Done. Committed as 7156a/rev. 14567.

#20 - 05/19/2023 06:22 AM - Greg Shah

Do we need any additional review or testing before merging to trunk?

#21 - 05/19/2023 06:35 AM - Alexandru Lungu

I've run the changes across customer applications and the individual set of tests (without issues). The core of the changes is in #7355-12.

Constantin, can you confirm that your scenario which caused #7355-1 is fixed now?

#22 - 05/19/2023 06:59 AM - Constantin Asofiei

Alexandru Lungu wrote:

> Constantin, can you confirm that your scenario which caused #7355-1 is fixed now?

Confirmed, the #7229 test is fixed.

#23 - 05/19/2023 02:02 PM - Greg Shah

Can we merge this to trunk?

#24 - 05/19/2023 02:28 PM - Constantin Asofiei

Greg Shah wrote:

> Can we merge this to trunk?

Is OK from my part.

#25 - 05/19/2023 02:33 PM - Greg Shah

OK, it can go after 7241a.

#26 - 05/22/2023 06:12 AM - Alexandru Lungu

Doing the rebase now. Planning to merge asap.

#27 - 05/22/2023 06:21 AM - Alexandru Lungu

Merged 7355a to trunk as rev. 14577. Archived 7355a.

#28 - 05/30/2023 04:49 AM - Alexandru Lungu

- Assignee set to Alexandru Lungu

This can be closed.

Files

datetimetz_count.patch	3.53 KB	05/16/2023	Alexandru Lungu