Database - Bug #6301

incorrect initialization of date-related fields with today and now literals

04/27/2022 01:18 PM - Ovidiu Maxiniuc

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

Description

Related issues:

Related to Database - Feature #6453: temp-table features	**Closed**

History

#1 - 04/27/2022 01:29 PM - Ovidiu Maxiniuc

The issue can be observed at runtime: the "now" moment is not evaluated for each record at the moment the record is created (as 4GL does) but when the table is created. The problem is somewhere in RecordMeta's c'tor(line 277), PropertyMeta's c'tor (line 179) and BaseRecord.initialize() (line 332).

This affects all date related data-types (date, datetime and datetime-tz) fields when they are defined with now and today initial values.

Here is a testcase:

```
define temp-table tt1
    field f1 as datetime-tz init now.

message "now:" now.
create tt1.
message f1.

pause message "Wait a minute!...".

message "now:" now.
create tt1.
message f1.
```

After launching the procedure it will print the current timestamp twice and prompt for a keystroke to continue. The message lines will be replaced with with similar ones but:

- in 4GL there will be different timestamps, reflecting the moment when the now function/literal was evaluated;
- in FWD only the first line will be updated, the second line (the field) will be the same because the value was evaluated when the temp-table metadata was processed.

The today is more difficult to observe but this is doable if the procedure is launched just before midnight and the second part is executed after passing 12 o'clock.

#2 - 04/27/2022 01:47 PM - Greg Shah

- Project changed from Runtime Infrastructure to Database

#3 - 10/08/2022 06:06 PM - Eric Faulhaber

- % Done changed from 0 to 100

- Status changed from New to Review

- Assignee set to Eric Faulhaber

A fix is committed to 3821c/14279. I have tested with the above test case.

Ovidiu, please review.

Alexandru, please take a look at FastCopyHelper.getInitialValue. This method is affected by the update, in that it relies upon PropertyMeta.getInitialValue, and the return value from the latter method is now no longer deterministic for a particular PropertyMeta instance. That is, for initial values which represent the built-in functions now and today, you will get a different result every time PropertyMeta.getInitialValue is invoked. FastCopyHelper is the only class dependent upon PropertyMeta.getInitialValue, so I just wanted you to consider whether this change will cause a problem for that case.

#4 - 10/09/2022 08:16 AM - Constantin Asofiei

Eric, 3821c/14279 regresses some automated tests with this:

```
java.lang.ClassCastException: java.lang.Long cannot be cast to java.util.Date
        at java.util.Date.compareTo(Date.java:131)
        at com.goldencode.p2j.persist.orm.BaseRecord.setDatum(BaseRecord.java:812)
        at com.goldencode.p2j.persist.Record._setDate(Record.java:481)
```

#5 - 10/09/2022 04:26 PM - Eric Faulhaber

Constantin Asofiei wrote:

> Eric, 3821c/14279 regresses some automated tests with this:
> [...]

Thanks for finding this. Fixed in 3821c/14280.

#6 - 10/10/2022 03:16 AM - Alexandru Lungu

Eric Faulhaber wrote:

> Alexandru, please take a look at FastCopyHelper.getInitialValue. This method is affected by the update, in that it relies upon

PropertyMeta.getInitialValue, and the return value from the latter method is now no longer deterministic for a particular PropertyMeta instance. That is, for initial values which represent the built-in functions now and today, you will get a different result every time PropertyMeta.getInitialValue is invoked. FastCopyHelper is the only class dependent upon PropertyMeta.getInitialValue, so I just wanted you to consider whether this change will cause a problem for that case.

The change also fixes a regression for copy-temp-table. When doing such copy in loose-mode (using default values for unmatched fields), the date fields should be initialized with the now of the copy operation (not table creation). The non-deterministic approach is the right way indeed.

#7 - 10/14/2022 08:05 AM - Greg Shah

- Related to Feature #6453: temp-table features added

#8 - 10/17/2022 02:01 PM - Eric Faulhaber

- Status changed from Review to Test

Igor reported a regression for a table with two fields of type datetime and datetime-tz, respectively, both with an initial value of now:

```
[10/17/2022 15:47:31 GMT+03:00] (com.goldencode.p2j.util.TransactionManager:SEVERE) Abnormal end; original err
or:
java.lang.ClassCastException: java.sql.Timestamp cannot be cast to java.time.OffsetDateTime
       at com.goldencode.p2j.persist.orm.types.TypeManager.setOffsetDateTimeParameter(TypeManager.java:674)
       at com.goldencode.p2j.persist.orm.types.DatetimetzType.setParameter(DatetimetzType.java:143)
       at com.goldencode.p2j.persist.orm.Persister.insert(Persister.java:847)
       at com.goldencode.p2j.persist.orm.Persister.insert(Persister.java:400)
       at com.goldencode.p2j.persist.orm.Session.save(Session.java:683)
       at com.goldencode.p2j.persist.orm.Validation.flush(Validation.java:569)
       at com.goldencode.p2j.persist.orm.Validation.validateUniqueIndices(Validation.java:436)
       at com.goldencode.p2j.persist.orm.Validation.validateMaybeFlush(Validation.java:329)
       at com.goldencode.p2j.persist.RecordBuffer.validateMaybeFlush(RecordBuffer.java:10513)
       at com.goldencode.p2j.persist.RecordBuffer.access$2200(RecordBuffer.java:1461)
       at com.goldencode.p2j.persist.RecordBuffer$Handler.invoke(RecordBuffer.java:12037)
       at com.goldencode.p2j.persist.$__Proxy6.setFint(Unknown Source)
       at com.goldencode.testcases.dmo._temp.Tt_1MethodAccess.invoke(Unknown Source)
       at com.goldencode.p2j.util.Utils.invoke(Utils.java:1635)
       at com.goldencode.p2j.persist.TemporaryBuffer$ReferenceProxy.invoke(TemporaryBuffer.java:9027)
       at com.goldencode.p2j.persist.$__Proxy7.setFint(Unknown Source)
       at com.goldencode.testcases.ias.TempTable.lambda$execute$7(TempTable.java:66)
       at com.goldencode.p2j.util.Block.body(Block.java:636)
```

He tracked it back to the fact that we use the same Supplier object to implement both fields' initial value, when they need to be separate.

#9 - 10/17/2022 02:17 PM - Eric Faulhaber

Fixed in 6129b/14302. Tested with this simple test case:

```
def temp-table tt1
   field f1 as datetime    init now
   field f2 as datetime-tz init now.

create tt1.

display tt1.
```

Igor, please confirm this fixes your more complex test.

#10 - 10/17/2022 09:16 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> A fix is committed to 3821c/14279. I have tested with the above test case.
>
> Ovidiu, please review.

Interesting idea, different than the one I envisioned. I understand that it is quite open for future dynamic initial values (currently only 3 dynamic expressions). Probably one of the most documented commit. The regressions which were already fixed were so subtle - probably wouldn't spot them. I am OK with the implementation.

Here is an additional case to test:

```
define temp-table tt0
  field f1 as date init today
  field f2 as date extent 2 init today
  field f3 as datetime init now
  field f4 as datetime extent 2 init now
  field f5 as datetime-tz init now
  field f6 as datetime-tz extent 2 init now.

create tt0.
    message f1 f3 f5.
    message f2[1] f4[1] f6[1].
    message f2[2] f4[2] f6[2].

define variable bth as handle.
create temp-table bth.
bth:create-like(temp-table tt0:handle).
bth:temp-table-prepare("tt9").

define variable bbh as handle.
bbh = bth:default-buffer-handle.
bbh:buffer-create().
    message bbh::f1 bbh::f3 bbh::f5.
    message bbh:buffer-field("f2"):buffer-value(1)
            bbh:buffer-field("f4"):buffer-value(1)
            bbh:buffer-field("f6"):buffer-value(1).
    message bbh:buffer-field("f2"):buffer-value(2)
            bbh:buffer-field("f4"):buffer-value(2)
```

```
            bbh:buffer-field("f6"):buffer-value(2).
```

Note that there is the LegacyFieldInfo.initial in TableMapper which is still computed incorrectly (locally, at the moment the table is processed by this class). This value should also be a accesses of this member should be Supplier / DynamicInitializer or the getter calls should be replaced with different equivalent APIs.