# Database - Bug #4492

## database trigger defined with abbreviation should create a new buffer

01/06/2020 06:17 AM - Adrian Lungu

| | | | |
|---|---|---|---|
| **Status:** | WIP | **Start date:** | |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | Adrian Lungu | **% Done:** | 20% |
| **Category:** | | **Estimated time:** | 0.00 hour |
| **Target version:** | | | |
| **billable:** | No | **case_num:** | |
| **vendor_id:** | GCD | **version:** | |

**Description**

### History

#### #1 - 01/06/2020 06:25 AM - Adrian Lungu

A database trigger defined in the following manner behaves differently on FWD:

```
TRIGGER PROCEDURE FOR CREATE OF dum.

find last dummy no-error.

if available dummy then do:
   dum.f2 = 1 + dummy.f2.
   message "Dummy: " + string(dummy.f2).
end.
else do:
   dum.f2 = 0.
   message "Dummy none".
end.

message "dum: " + string(dum.f2).
```

4GL makes a distinction between dum and dummy, even though dum is an abbreviation for dummy.

- dum is the buffer which fired the database trigger; a buffer which doesn't represent a record in the database yet
- dummy is the default buffer inside this procedure, one which can be involved in find statements

FWD considers that dum and dummy are the same unchangeable buffer, due to the fact that dum is an abbreviation for dummy, thus it is just a sugar syntax. The conversion time should be changed in order to create two buffer instances. To be taken in consideration that any other abbreviation refers to dummy, not to dum (i.e. d can be used to refer to dummy buffer).

#### #3 - 01/06/2020 08:50 AM - Greg Shah

*- Project changed from Bugs to Database*

*- Subject changed from Database trigger defined with abbreviation should create a new buffer to database trigger defined with abbreviation should create a new buffer*

*- Start date deleted (01/06/2020)*

#### #4 - 01/09/2020 12:04 PM - Adrian Lungu

*- Status changed from New to WIP*

#### #5 - 01/10/2020 08:52 AM - Adrian Lungu

There seems to be a sub-issue in #4492-1. The find last statement will find the newly created record. In 4GL, an error is thrown as no available record could be found.

```
do transaction:
   create dummy.
   create dummy.
   create dummy.
end.

for each dummy:
   delete dummy.
end.



TRIGGER PROCEDURE FOR CREATE OF dummy.

define buffer b for dummy.

find last b no-error.
if available b then do:
   message "Available.".
end.
else do:
   message "Not available.".
end.
```

4GL prints one Not available and two Available, while FWD prints 3 Available. This should be fixed. I will focus on this issue first, before changing the conversion to solve #4492-1.

#### #6 - 01/10/2020 10:46 AM - Adrian Lungu

Issue #4492-5 fixed. RecordBuffer.create() should have called the trigger before updating the dirtyContext, otherwise the newly created record (which was residing in the dirtyContext) could be found by find last. The fix was delivered on branch 4231b/rev. 11471.

#### #7 - 01/11/2020 09:56 AM - Adrian Lungu

Issue #4492-1 is implying multiple changes regarding conversion:

1. If the buffer used has the same name as the schema, then no changes are required
2. Otherwise an "abbreviation buffer" should be used as ArgumentBuffer, while the full length buffer should represent another instance

The first steps are to change the javaname generated for the buffer to represent the abbreviation (check for conflicts!). Afterwards, the generation of the full length buffer instance shouldn't be suppressed. Thus, the trigger procedure in #4492-1 change as following:

```
@SchemaTrigger(buffer = Dummy.Buf.class, event = DatabaseEventType.CREATE, table = "fwd.dummy", hasOldBuffer =
 false)
public class CreateTrg
extends DatabaseTrigger<Dummy.Buf>
{
   Dummy.Buf dummy;

   /**
    * External procedure (converted to Java from the 4GL source code
    * in custom/t4492/create-trg.p).
    */
   @Override
   public void create(final Dummy.Buf dummy)
   {
      externalProcedure(CreateTrg.this, TransactionType.FULL, new Block((Body) () ->
      {
         RecordBuffer.openScope(dummy);
         CreateTrg.this.dummy = dummy;
         silent(() -> new FindQuery(dummy, (String) null, null, "dummy.id asc").last());

         if (dummy._available())
         {
            dummy.setF2(plus(1, dummy.getF2()));
            message(concat(new character("Dummy: "), valueOf((integer) new FieldReference(dummy, "f2").getValu
e())));
         }
         else
         {
            dummy.setF2(new integer(0));
            message("Dummy none");
         }

         message(concat(new character("dum: "), valueOf((integer) new FieldReference(dummy, "f2").getValue()))
);
      }));
   }
}
```

```
@SchemaTrigger(buffer = Dummy.Buf.class, event = DatabaseEventType.CREATE, table = "fwd.dummy", hasOldBuffer =
 false)
public class CreateTrg
extends DatabaseTrigger<Dummy.Buf>
{
   Dummy.Buf dummy = RecordBuffer.define(Dummy.Buf.class, "fwd", "dummy");

   Dummy.Buf dum;

   /**
    * External procedure (converted to Java from the 4GL source code
    * in custom/t4492/create-trg.p).
    */
   @Override
   public void create(final Dummy.Buf _dum)
   {
      CreateTrg.this.dum = RecordBuffer.defineAlias(Dummy.Buf.class, _dum, "dum", "dum");

      externalProcedure(CreateTrg.this, TransactionType.FULL, new Block((Body) () ->
```

```
      {
         RecordBuffer.openScope(dum);
         silent(() -> new FindQuery(dummy, (String) null, null, "dummy.id asc").last());

         if (dummy._available())
         {
            dum.setF2(plus(1, dummy.getF2()));
            message(concat(new character("Dummy: "), valueOf((integer) new FieldReference(dummy, "f2").getValu
e()));
         }
         else
         {
            dum.setF2(new integer(0));
            message("Dummy none");
         }

         message(concat(new character("dum: "), valueOf((integer) new FieldReference(dum, "f2").getValue()));
      }));
   }
}
```

#### #8 - 01/12/2020 10:13 AM - Greg Shah

Didn't you previously find that some abbreviations were treated as the same buffer as dummy but dum was somehow special (and treated as a different buffer)?  If I understood that correctly, please explain what makes dum special (how do we detect that it is different).

#### #9 - 01/13/2020 05:55 AM - Adrian Lungu

Greg Shah wrote:

> Didn't you previously find that some abbreviations were treated as the same buffer as dummy but dum was somehow special (and treated as a different buffer)?  If I understood that correctly, please explain what makes dum special (how do we detect that it is different).

dum is special because it is used inside the TRIGGER PROCEDURE FOR CREATE OF dum statement.

Inside any external procedure, dummy and its abbreviations are all referencing the same dummy default buffer. TRIGGER PROCEDURE FOR CREATE OF dum. is similar to DEFINE PARAMETER BUFFER dum FOR dummy, thus it **overrides** the value of dum with the newly created buffer. The other abbreviations are still referencing the default dummy buffer, while dum is different.

The conversion changes should reflect that TRIGGER PROCEDURE FOR CREATE OF dum is actually some sort of DEFINE PARAMETER BUFFER dum FOR dummy.

#### #10 - 01/13/2020 01:32 PM - Adrian Lungu

Finally, I've got to the responsible area in the conversion. Mainly it looks like a new BUFFER-SCORE Progress Ast should be created to reflect the new buffer reference. By now, record_scoping_prep.rules seems to be the file which require changes. I will try to make TRIGGER PROCEDURE FOR CREATE OF dum behave as close as possible as DEFINE PARAMETER BUFFER dum FOR dummy; firstly by generating a new BUFFER-SCOPE and afterwards see how this will affect the conversion.

#### #11 - 01/14/2020 04:24 PM - Adrian Lungu

*- % Done changed from 0 to 20*

The root of the problem is clearly at a lower level than anticipated. After some code analysis, the parser is the one that should handle the changes. It is nearly impossible to solve this scenario, unless the parser changes the annotations.

Regarding TRIGGER PROCEDURE FOR CREATE OF dum, dum is seen as a TABLE token, thus a reference to a table/defined buffer already in the SchemaDictionary. As far as I could catch up with the ProgressParser, there is a slim number of places where a buffer is actually defined (i.e. def_buf_stmt()). TRIGGER PROCEDURE FOR CREATE OF dum is creating problems as there is no mechanism for a TABLE token to define a buffer or to identify that is related to a TRIGGER PROCEDURE FOR CREATE.

I didn't have many encounters with ProgressParser by now, thus this fix may be quite time consuming. Finally, this bug was found while creating smoke tests; it was not encountered yet in real scenarios. Maybe set it on a lower priority for now?

#### #12 - 01/14/2020 04:32 PM - Greg Shah

I agree, let's pause work on this.

#### #13 - 02/14/2020 04:57 AM - Adrian Lungu

Adrian Lungu wrote:

> Issue #4492-5 fixed. RecordBuffer.create() should have called the trigger before updating the dirtyContext, otherwise the newly created record (which was residing in the dirtyContext) could be found by find last. The fix was delivered on branch 4231b/rev. 11471.

This causes a severe regression. I think this should be solved in another way. By now, 4231b/rev. 11356 reverts the changes.

#### #14 - 06/20/2020 12:10 PM - Greg Shah

Task branch 4231b has been merged to trunk as revision 11347.