Database - Feature #2024

Feature # 1658 (Closed): improve database trigger support

implement runtime support for database triggers

02/21/2013 01:39 PM - Eric Faulhaber

Status:	Closed	**Start date:**	06/19/2013
Priority:	Normal	**Due date:**	07/01/2013
Assignee:	Ovidiu Maxiniuc	**% Done:**	100%
Category:		**Estimated time:**	72.00 hours
Target version:	Runtime Support for Server Features		
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Base Language - Feature #2108: event model rationalization	**Closed**	**06/08/2013**	**06/14/2013**
Related to Database - Bug #2222: WRITE event triggered too early	**Hold**		
Related to Database - Bug #2223: ASSIGN schema triggers without reference to ...	**New**		

History

#1 - 02/21/2013 01:41 PM - Eric Faulhaber

- Parent task set to #1658

#2 - 04/12/2013 11:20 AM - Eric Faulhaber

- Estimated time set to 72.00

#3 - 04/25/2013 11:00 AM - Eric Faulhaber

- Start date set to 06/17/2013

- Due date set to 06/27/2013

- Assignee set to Stanislav Lomany

#4 - 05/01/2013 06:15 PM - Eric Faulhaber

- Due date changed from 06/27/2013 to 07/01/2013

- Start date changed from 06/17/2013 to 06/19/2013

#5 - 09/17/2013 04:15 PM - Eric Faulhaber

- Assignee changed from Stanislav Lomany to Ovidiu Maxiniuc

#6 - 10/14/2013 02:52 PM - Ovidiu Maxiniuc

- Status changed from New to WIP

#7 - 11/05/2013 12:15 PM - Ovidiu Maxiniuc

Here are a few things about triggers that are not implemented or need to be adjusted:

- there are two kinds of triggers: **schema triggers** (declared in P4GL using the data dictionary) and **session triggers** declared at runtime using ON <event> OF.
- the session triggers are managed in layers (almost like the existing P2J scopes, except that only sub-routine blocks are taken into consideration):
  - the scope (where a session trigger is active) in a layer starts with it's declaration (with ON <event> OF) until another trigger is declared, trigger is REVERT ed, DISABLE d, another trigger overwrite it or when the routine in which it was declared ends.
  - a declared session trigger acts during the scope of the procedure/function it was declared and procedures/functions called by it. If a trigger is REVERT ed, the old trigger declared in calling subroutine enter in effect. REVERT only applies to the trigger in the current layer.
  - only one session trigger per object is active at a moment. When adding a second trigger for same event/object, the previous is dropped.

The REVERT will clean the current layer of the current trigger and the trigger from upper layer (if any) will take effect.
    - using DISABLE TRIGGER a trigger defined in an upper layer is prohibited to execute during the current subroutine and subroutines called by it. Once the subroutine exit the trigger is enabled again. If it was defined in the same layer with DISABLE TRIGGER, it will be active until that moment and will be dropped at the end of the layer.
    Because of these, I chosen to implement in TriggerManager a stack of 'Layers' that are automatically pushed/popped when execution of the program enter leave subroutines.
- the triggers are set on individual tables of a database. In the actual conversion, they are set on schema. Ex: ON FIND OF book is converted to registerDatabaseTrigger(DatabaseEventType.FIND, Book.Buf.class, TriggerBlock0.class ...). The table of the correct database cannot be directly identified using these parameters. I will use the interface querying to obtain the dmo interface class (Book.class) that will allow to extract the schema from DatabaseManager and then the primary database. I need to investigate how to handle non-local databases.
- The current triggers whose configuration are loaded from directory do not fit the schema triggers from P4GL:
    - current triggers have some extra properties (including the on_undo) that are not compatible with P4GL.
    - in P4GL, for each table, 7 triggers can be defined (CREATE//DELETE/FIND/WRITE/REPLICATION-CREATE/REPLICATION-DELETE and REPLICATION-WRITE). Each of them has a flag if it is overrideable or not. The code for them is saved to disk as individual files with TRIGGER PROCEDURE FOR <event> OF <table>. A similar trigger (ASSIGN) can be defined for each field of the database.
    - I think the best solution is to have these trigger procedure files converted with the rest of program (eventually in a separate folder/package) and configured in directory:

```
<node class="container" name="database">
   <node class="container" name="p2j_test">
      <node class="container" name="p2j">
         <node class="container" name="triggers">
            <node class="container" name="Book">
               <node class="container" name="FIND">
                  <node class="string" name="class">
                      <node-attribute name="value" value="com.goldencode.testcases.triggers.SchemaTrigg
erFindBook"/>
                  </node>
                  <node class="boolean" name="overrideable">
                      <node-attribute name="value" value="true"/>
                  </node>
               <node/>
               <node class="container" name="ASSIGN">
                  <node class="container" name="book-title">
                     <node class="string" name="class">
                        <node-attribute name="value" value="com.goldencode.testcases.triggers.SchemaTri
ggerAssignBookBookTitle"/>
                     </node>
                     <node class="boolean" name="overrideable">
                         <node-attribute name="value" value="true"/>
                     </node>
                  ...
```

When the persistence initializes for a database, the triggers will be loaded (much like current implementation, but the classes were hand-written instead of 'imported' from P4GL and converted with the rest of the application) and mapped to the specified tables from the database and executed when needed (when triggers are not disabled and when the active session trigger is not OVERRIDE -ing it). What should I do with current implementation. I understand from Constantin that it is something written specifically for Majic. Should I let the two implementations run in parallel or should I remove old implementation and eventually add support using the new implementation ?

- There is a collision of buffers when declaring trigger and calling can-find function for the same table. As a result the buffer is not initialized with RecordBuffer.define and not scoped with RecordBuffer.openScope. I will fix this.

#8 - 11/05/2013 12:50 PM - Greg Shah

> Because of these, I chosen to implement in TriggerManager a stack of 'Layers' that are automatically pushed/popped when execution of the program enter leave subroutines.

Consider that our current UI triggers work in this same way, and even have support for revert and so forth. Since these are both implemented using the ON statement, it is not surprising that they would the same way (or very similarly). It seems there may be an advantage to reusing the same implementation code rather than duplicating a very similar implementation. I prefer it to be a separate instance of that same implementation code, if possible. This would isolate the UI and database stuff. What do you think?

> I think the best solution is to have these trigger procedure files converted with the rest of program (eventually in a separate folder/package) and configured in directory:

I would be OK with this, however I do wonder if we can eliminate it completely. Can't we store the needed information in the generated classes using runtime annotations? Then we can scan the loaded application at server startup to build the "schema triggers" list. This has the major advantage of much less configuration. The only downside is that schema triggers can only be disabled with an application recompile, but essentially that was the case in the original 4GL implementation too (since it was hard coded in the schema).

#9 - 11/06/2013 01:38 PM - Ovidiu Maxiniuc

> Consider that our current UI triggers work in this same way, and even have support for revert and so forth. Since these are both implemented using the ON statement, it is not surprising that they would the same way (or very similarly). It seems there may be an advantage to reusing the same implementation code rather than duplicating a very similar implementation. I prefer it to be a separate instance of that same implementation code, if possible. This would isolate the UI and database stuff. What do you think?

I like the idea of unifying handling of UI and database triggers. Support for schema triggers and overriding flag can be added somehow.
I have investigated the possibility to have a common ancestor as an abstract class but there are many differences and not much than the Scopeable interface in common. In fact the Scopeable is implemented by the LogicalTerminal that delegates the calls to static methods of the TriggerManager with context-local data as parameters. The code of current TriggerManager is additionally complicated because each event is applied to a list of widgets and the support for persistent run triggers; the database triggers are applied to a single database-object and session triggers are they are certainly 'discarded' when leaving the top-level block.
If we extract the common implementation we get only the scopeStart() in a base class. The interface is also not too common as the TriggerManager has only static methods so the polimorphism cannot be applied.
There is also a difference in the implementation of visible by the following code.

```
DEFINE BUTTON b1 LABEL "Hit Me".
DEFINE BUTTON b2 LABEL "Exit".
DEFINE FRAME fr1 b1 b2.
```

```
ON CHOOSE OF b1 DISPLAY "t1 " + SELF:LABEL.
ON CHOOSE OF b1 DISPLAY "t2 " + SELF:LABEL.
ON CHOOSE OF b1 REVERT.

ENABLE b1 b2 WITH FRAME fr1.
WAIT-FOR CHOOSE OF b2.
```

When firing the 1st button the trigger t1 is fired and prints "t1 Hit M". With database triggers, none of those would be fired, only the schema trigger would act, if any.

> Can't we store the needed information in the generated classes using runtime annotations? Then we can scan the loaded application at server startup to build the "schema triggers" list. This has the major advantage of much less configuration.

Yes, you are right we can extract the table/field name on which the trigger applies from the TRIGGER PROCEDURE FOR statement from those files and add annotations, the only issue here is that the trigger source file taken from P4GL do not contain the Overrideable attribute so these should be manually add some additional hints for that.

#10 - 11/06/2013 02:16 PM - Greg Shah

> the only issue here is that the trigger source file taken from P4GL do not contain the Overrideable attribute so these should be manually add some additional hints for that

Where does the overridable flag come?

If it comes from the schema, let's read that flag at conversion time and add that flag to the annotation. The only problem here would be if the same trigger procedure is used for different schemas and one uses overridable and the other did not. I doubt that is even possible, since any parsed 4GL code will have its static database references hard coded to a single schema.

#11 - 11/07/2013 02:01 AM - Ovidiu Maxiniuc

> Where does the overridable flag come?

> If it comes from the schema, let's read that flag at conversion time and add that flag to the annotation. The only problem here would be if the same trigger procedure is used for different schemas and one uses overridable and the other did not. I doubt that is even possible, since any parsed 4GL code will have its static database references hard coded to a single schema.

Indeed, for each table in the schema administration of data dictionary there are two flags: one is the overridable attribute, the other is CRC. The latter is not important as it only enforce the compiler to check the code for errors.
On the other hand, taking into account that overriding an non-overrideable trigger is fatal error and the application performed without errors on p4gl, we can assume that all session triggers are defined on overrideable schema triggers so we could ignore this flag.

#12 - 11/07/2013 08:59 AM - Greg Shah

> Indeed, for each table in the schema administration of data dictionary there are two flags: one is the overridable attribute,

Can you use the same trigger procedure as the trigger for 2 different tables? If so, then we would have to store the overridable attribute in a way that can be checked conditionally based on the table being used.

> On the other hand, taking into account that overriding an non-overrideable trigger is fatal error and the application performed without errors on p4gl, we can assume that all session triggers are defined on overrideable schema triggers so we could ignore this flag.

We have seen similar behavior from validation expressions. They can have syntax errors and thus could never possibly work. But the code that references them will silently operate as if they returned true.

I wonder if the trigger actually executed or not. Please check. If it doesn't execute in this case, then we must duplicate that non-execution.

#13 - 11/07/2013 10:12 AM - Ovidiu Maxiniuc

Observations on schema triggers::

- checking the CRC box will force compile the trigger which mean the file must exist (or error 293 will be issued: *""test-trigger1.t" was not found."*)
- checking the CRC box will accept an empty trigger file but will complain at runtime.
- checking the CRC box will accept any valid progress code is accepted as long as it doesn't *"Missing FOR, FIND or CREATE for a table with name in current block. (232)"*
- at runtime, *"Cannot run test-trigger1.t for a FIND, DELETE, or CREATE trigger. (3248)"* will force an abend if compiled file is not a db trigger procedure

- at runtime , *"Trigger for table Book is executing test-trigger1.t, which is compiled against a different logical database name. (4461)"* will force an abend if compiled file is not a db trigger procedure for correct database.
- at runtime , *"Trigger for table Book executing test-trigger1.t which is FOR address table. (3240)"* will force an abend if compiled file declares any other db event trigger.
- the declared event type in the trigger procedure is actually ignored the procedure is executed for the event defined in data dictionary without ant errors or warnings. For example, in data dictionary for CREATE of table Book is associated a trigger procedure that specifies: TRIGGER PROCEDURE FOR Find OF book. it will be called on CREATE event and will not be called on FIND.
- even if the trigger file was compiled/checked in data dictionary, the result is not persisted in database schema so at runtime, file is re-compiled. If CRC is checked and content changed the program will end with following error: *"CRC for database trigger procedure test-trigger1.t does not match schema. (3242)"*. Looks like the CRC of the source trigger procedure when editing the database is stored and compared with the CRC of procedure file when it is recompiled at runtime.

As conclusion, a trigger procedure can be used within a database schema only for one table (the one declared inside the file) but the event they declare is ignored so the procedure can be used for any database event.

#14 - 11/12/2013 02:02 PM - Ovidiu Maxiniuc

Today I had some questions about the WRITE trigger and its implementation so I had the following summarized mail discussion with Constantin:

OM: I have the following issue: in a WRITE trigger there should be two buffer parameters with buffer but with different values old and new. I wonder is this is possible.
At this moment in RecordBuffer the 'snapshot' is available as DMO, but I would need a special Buffer object that won't conflict with the Buffer that contains the current value that awaits to be persisted.
Theoretically a new buffer can be built using RecordBuffer.define(), but I think that a collision is possible because both buffers will contain instances of record (same Id from primary index)

CA: I don't think you can send plain DMO instances as parametri (even if they would be accepted), because you will lose access to RecordBuffer.
Please test the following resources newBuffer and oldBuffer in trigger:
1. what ID/UNIQUE-ID resource has fiecareeach of the two?
2. oldBuffer is read-only. What happens if DEFINE BUFFER bb FOR oldBuffer and then use the buffer to alter something? Any error? What if you directly use the oldBuffer to change something?
3. can you do RELEASE on oldBuffer? what about newBuffer?
4. what happens if you change a field from newBuffer? does it get to db (and caller code)?
5. what happens if in the trigger you define another explicit BUFFER for same table and change something? Will it enter recursivity (calling WRITE trigger again)? any eroare?

OM:
1. The primary index is unchanged and same for all triggers. The unique-id is a little different. The new buffer is same with main program, consistent with all triggers,
Instead, the OLD buffer is new for each called trigger.
2. The NEW buffer is modifiable, the OLD is read-only
3. One cannot release any of those buffers. Cannot RELEASE record from buffer . Trigger is executing. (6026) I will add this to to my TODO list.
4. (from docs) *The AVM automatically validates a record when releasing it. You can also use the VALIDATE statement to explicitly validate a record. In either case, WRITE triggers execute before the validation occurs (so WRITE triggers can correct values and do more sophisticated validation).*
5. It's an error to change the current cursor, but you can change cursors that are not "parameters" of any trigger. Probably a kind of flag block circular references.

CA:
6. by resource ID I meant string(buffer newBuffer:handle) and string(buffer oldBuffer:handle). But as long as UNIQUE-ID are different that they are too.
7. the conclusion is that oldBuffer must be a "read-only" buffer. I think a new proxy must be created, using something like RecordBuffer.defineReadOnlyBuffer. Package-private API.
8. BTW, does triggers work on temp tables? I mean user-defined, not schema.

OM:
6. yes, they are different, too.
7. in the generated code, it already writes:
oldBuff.openReadOnlyScope();
in the implementation I can take care of R/O issue, However, when entering the trigger block the oldBuff is not yet R/O. I do't think this is an issue, so the old builder should work. I am concerned about the collision of the record ID (as these are basically two snapshots the same record) in the two different buffers.
8. you can define the triggers but they will never fire. That's logical if you think that you are in full control of the data you enter in temp-tables.

CA:
Re 7: I don't think there will be collisions regarding record IDs, as already we allow two (or more) different buffers to reference different records (this is actually how 4GL works). Just make sure the record referenced in the oldBuffer is a copy, and not the same instance.

#15 - 11/13/2013 01:19 PM - Ovidiu Maxiniuc

Here is a 100 points question.
What's the best solution for notifying ASSIGN triggers ?

In P4GL, the ASSIGN trigger is fired just when the assign statements occur. I believe, in P2J we should adjust the DMO implementation like this:

```
public class BookImpl
...
   /**
    * Setter: Title
    *
    * @param   bookTitle
    *          Title
    */
   public void setBookTitle(Text bookTitle)
   {
      Text bookTitleOld = new Text();
      bookTitleOld.assign(this.bookTitle);
      this.bookTitle.assign(bookTitle, true);
      DatabaseTriggerManager.triggerAssign(this.class, "book-title", this.bookTitle, bookTitleOld)
   }
```

The generated code stays the same except for dmos which became more complicated and code is multiplied. This is probably the fast from execution point of view.

A second solution would be to write a special assign static method for database fields only: instead of generating book.setBookTitle(new character("abc"));:

```
    assign(book, "book-title", new character("abc"));
```

The assign will contain the code more-or-less code from 1st solution. The advantage of this is that code dmo remains the same, code is readable, but the downside is that the new assign method will be rather slow because reflection will be needed.

I'm thinking of a 3rd solution, to do all that inline, but that's really messy, nobody will be able to read the simple assign code.

#16 - 11/13/2013 01:30 PM - Eric Faulhaber

Wouldn't it be easier/cleaner to invoke the trigger manager from the record buffer proxy? It's already an architected intercept mechanism for all calls to set a buffer value from business logic. Also, I think we don't want to invoke the trigger manager when Hibernate is calling the setters, right?

Is there any case this wouldn't cover?

#17 - 11/27/2013 03:53 PM - Ovidiu Maxiniuc

I think that parts of the progress.g should be re-written to allow the correct syntax to be parsed. It here are the parts I worked on:

```
def_parm_stmt returns [String varname = null]
   :
      p:KW_PARM!
      (
         options { generateAmbigWarnings = false; }
         :
           varname=table_parm[true]
         | def_buf_stmt[false, false]
         | varname=regular_parm[#p, null]
      )
   ;
```

The second argument for regular_parm was not used at all. I passed instead a reference to a field that will be used as for LIKE clause if not null and LIKE clause do not occur normally in token stream:

```
regular_parm [Aast param, Aast reference] returns [String varname = null]
   :
      {
         int vtype = -1;
         int size  = -1;
         boolean as_or_like_processed = false;
      }
      s:symbol  { varname = #s.getText(); }
      (
         options { generateAmbigWarnings = false; }
         :

           a:as_clause[varname, true]
           {
              vtype = #a.getFirstChild().getType();
              as_or_like_processed = true;
           }

         | l:like_clause[varname, null, false]
           {
              vtype = #l.getFirstChild().getType();
              as_or_like_processed = true;
           }

         | initializer[vtype]
         | size = extent
           {
              if ((size == -1 || size > 0) && param != null)
                 param.putAnnotation("extent", new Long(size));
           }
         | label
         | format_string[false]
         | column_label
         | decimals_clause
         | case_sensitive
         | KW_NO_UNDO
      )*
      {
         if (reference != null && !as_or_like_processed)
         {
             sym.addLocalVariableLike(varname, reference);
         }
       }
```

```
   ;


trigger_procedure_stmt
   :
      t:KW_TRIGGER^ { #t.setType(TRIGGER_PROCEDURE); }
      KW_PROC! KW_FOR!
      db_trigger_clause
   ;
```

The common part from both forms session and schema:

```
db_trigger_clause
   :
      // we have not confirmed that the force flag of record[] should be false
      (KW_CREATE | KW_DELETE | KW_FIND) KW_OF! record[true, false, false]

      // we have not confirmed that the force flag of record[] should be false
      | KW_WRITE KW_OF! r:record[true, false, false]
         (
           options { generateAmbigWarnings = false; }
           :
           write_referencing_phrase[#r, true]
         )?
         (
           options { generateAmbigWarnings = false; }
           :
           write_referencing_phrase[#r, false]
         )?

      | KW_ASSIGN KW_OF! f:lvalue
        (
           options { generateAmbigWarnings = false; }
           :
           assign_trigger_var_def[#f]
        )?
   ;
```

There is no NEW in ASSIGN trigger syntax. It also passed the ref parameter to regular_parm.

```
assign_trigger_var_def [Aast ref]
   :
      {
         String symname = null;
      }

      (KW_OLD^) (KW_VALUE!)?
      symname=regular_parm[null, ref]

      {
         ##.putAnnotation("define_buffer", new Boolean(false));

         // pass the AST with all options to the symbol resolver to set
         // those variable options accordingly
         sym.setVariableOptions(symname, ##);

         // write any non-standard options into annotations
         sym.annotateVariableOptions(symname, ##, true);
      }
   ;
```

on_stmt is simplified, the common parts with schema triggers are taken over by db_trigger_clause

```
on_stmt
   :
      {
         boolean bypass = false;
```

```
         boolean buf    = false;
         Aast    ref    = null;

         boolean oldBufScope = setupTriggerScope();
      }

      KW_ON^

      // here is the event specification for what should cause the trigger to fire
      (
         options { generateAmbigWarnings = false; }
         :
           // database trigger case
           db_trigger_clause
           (
              options { generateAmbigWarnings = false; }
              :
              KW_OVERRIDE
           )?

           // this is the key_label key_function form
         | event key_function DOT!    { bypass = true; }
...
```

write_referencing_phrase only handles references in WRITE event buffers:

```
write_referencing_phrase [Aast ref, boolean is_new]
   :
      {
         String txt = null;
      }
      (KW_NEW^ | KW_OLD^)
      (
         options { generateAmbigWarnings = false; }
         :
         KW_BUFFER!
      )?
      b:symbol
      {
         ##.putAnnotation("define_buffer", new Boolean(true));
         defineBufferFromSymbol(#b, ref.getText());
      }
   ;
```

The main problem is that the db_event is not generated any more in session triggers (the events will be children of KW_OF, like they are for TRIGGER_PROCEDURE in the case of schema triggers) which means a few more changes in TRPL code.

I think this is the correct way to handle some correct syntax forms like:

```
on assign of t1.f1 <trigger-body>
on assign of t1.f1 old value f2 <trigger-body>
on assign of t1.f1 old value f2 as int <trigger-body>
on assign of t1.f1 old value f2 like f1 <trigger-body>
on assign of t1.f1 old value f2 like f1 format "a_format" no-undo <trigger-body>
trigger procedure for assign of t1.f1 old f2 like t1.f3
```

#18 - 11/28/2013 11:23 AM - Greg Shah

Sadly, I independently found the same "implicit like" problem with the assign_trigger_var_def/regular_parm processing. It was found back in October when I was parsing a potential customer's code for analysis and I hit this issue in assign triggers. My solution was this:

```
@@ -8943,13 +8948,9 @@
       {
          if (varname != null)
          {
-            boolean setOpts = true;
-
             // pass the AST with all options to the symbol resolver to set
-            // those variable options accordingly (as long as this isn't
-            // a table-handle definition which has no options)
-            if (setOpts)
-               sym.setVariableOptions(varname, ##);
+            // those variable options accordingly
+            sym.setVariableOptions(varname, ##);

             // write any non-standard options into annotations
             sym.annotateVariableOptions(varname, ##, true);
@@ -9060,7 +9061,7 @@
          :
            varname=table_parm[true]
          | def_buf_stmt[false, false]
-         | varname=regular_parm[#p, false]
+         | varname=regular_parm[#p]
       )
    ;

@@ -11647,7 +11648,7 @@
  * these), however this code does not check for a repeated definition of
  * the options. Note that as an undocumented feature, Progress also allows
  * that the AS and LIKE clauses can be provided after other options. This is
- * supported.
+ * supported.  In fact, these clauses can be optional in certain circumstances.
  * <p>
  * These option rules are all implemented as separate rules to force a clean
  * tree structure for each option rather than accepting a flat structure
@@ -11658,11 +11659,10 @@
  *
  * @param    param
  *           The parameter AST where to save the extent annotation, if set. May be null.
- * @param    optional
- *           true to allow the AS and LIKE clauses to be
- *           optional.
+ *
+ * @return   The new variable's name.
  */
-regular_parm [Aast param, boolean optional] returns [String varname = null]
+regular_parm [Aast param] returns [String varname = null]
    :
       {
          int vtype = -1;
@@ -14932,6 +14932,9 @@
  */
 trigger_procedure_stmt
    :
+      {
+         Aast likeRef = null;
+      }
       t:KW_TRIGGER^ { #t.setType(TRIGGER_PROCEDURE); }
       KW_PROC! KW_FOR!
       (
@@ -14955,13 +14958,18 @@

          | KW_ASSIGN
            (
-                KW_OF! lvalue
-              | assign_trigger_var_def
+                KW_OF! lv:lvalue
+              | vd:assign_trigger_var_def[null]
            )
+           {
+              // the vd case is not exactly the "variable definition node", but it will
```

```
+               // suffice for our downstream needs
+               likeRef = (#lv != null) ? #lv : #vd.getImmediateChild(SYMBOL, null);
+            }
             (
                options { generateAmbigWarnings = false; }
                :
-               assign_trigger_var_def
+               assign_trigger_var_def[likeRef]
             )?
        )
    ;
@ -14988,25 +14996,54 @@
  * VALUE keyword (dropped) and then the variable definition
  * which is handled by {@link #regular_parm}.
  * <p>
+ * If the likeRef parameter is non-null AND this is a keyword OLD
+ * where there is no explicit AS or LIKE, then this code will
+ * implement an implicit LIKE clause.  This is an undocumented feature that
+ * was found in customer code.
+ * <p>
  * Called by {@link #trigger_procedure_stmt}.
+ *
+ * @param    likeRef
+ *           The variable or field definition to use for an implicit LIKE clause. This may be
+ *           null.
  */
-assign_trigger_var_def
+assign_trigger_var_def [Aast likeRef]
    :
       {
          String symname = null;
       }


-      (KW_NEW^ | KW_OLD^) (KW_VALUE)? symname=regular_parm[null, true]
+      (KW_NEW^ | KW_OLD^) (KW_VALUE!)? symname=regular_parm[null]


{
          ##.putAnnotation("define_buffer", new Boolean(false));
+
+         // the OLD [VALUE] varname case can have an IMPLICIT LIKE clause so we have to
+         // explicitly add the variable if it wasn't already added
+         if (##.getType() == KW_OLD)
+         {
+            Aast asChild   = ##.getImmediateChild(KW_AS, null);
+            Aast likeChild = ##.getImmediateChild(KW_LIKE, null);
+
+            // if either of these are present, no implicit LIKE is needed; the likeRef check
+            // is just for safety purposes
+            if (asChild  null && likeChild  null && likeRef != null)
+            {
+               // here is where we enable the variable lookup for all subsequent
+               // code in this scope, which should always be the external proc
+               sym.addLocalVariableLike(symname, likeRef);
+            }
+         }
+
+         // find the right node to annotate
+         Aast varDef = ##.getChildAt(0);


// pass the AST with all options to the symbol resolver to set
          // those variable options accordingly
-         sym.setVariableOptions(symname, ##);
+         sym.setVariableOptions(symname, varDef);


// write any non-standard options into annotations
-         sym.annotateVariableOptions(symname, ##, true);
+         sym.annotateVariableOptions(symname, varDef, true);
```

```
        }
    ;
```

I did take some care to make sure that the resulting variable defs were annotated correctly. I like that your approach has a bit less code, but my approach keeps the addLocalVariableLike() call in the same place as all the rest of the var options/annotations settings, so that seems easier to read/understand (to me at least). I also put some good comments in place to explain things since it is an undocumented behavior.

In regard to your other changes, I should note that the 4GL docs state that NEW is a possible keyword used in the assign trigger def. Have you tested that syntax as documented and found it does not work?

I am a bit nervous about your changes to move to a common db_trigger_clause approach. I did try that same thing when I first wrote this code, but if I remember correctly I found some deviations that forced me to have 2 slightly different approaches to the ON stmt versus the trigger procedures. It is possible that I didn't test this as well as you have, so you certainly may have found something that I did not. But I also know that I have run this code on several other customer apps and have had to make changes over time to get those other cases to parse right. Because all of this is so hazy, it will be very important for you to write enough testcases to have confidence that the changes you are making are safe. I hope that my previous split approach was unnecessary.

#19 - 11/28/2013 11:25 AM - Greg Shah

- File progress_parser_ges_upd20131126.zip added

This is the intermediate version of my progress.g changes which I plan to include with my work on #1634.

#20 - 11/28/2013 12:28 PM - Ovidiu Maxiniuc

> In regard to your other changes, I should note that the 4GL docs state that NEW is a possible keyword used in the assign trigger def. Have you tested that syntax as documented and found it does not work?

Yes, I was not able to compose any statement that contains assign trigger and new and the p4gl compiler to validate it. The Assign is a very particular case of trigger by the fact that it receives only the affected field and not the entire record (however, the whole record is accessible for both schema and session triggers). I was not able to find any reference about NEW in my docs. The OE 10.2B only mention the following syntax for assign triggers:

```
OLD [ VALUE ] old-field-name
```

While the ABL Database Triggers and Indexes chapter of Progress OpenEdge v11.0 Web Papers don't mention anything about syntax.

On the other hand, the old reference behaves exactly like a variable definition, clauses like no-undo, format and label were tested and proved to be taken in consideration. I could not prove the init clause as at the moment then the trigger code executes, the init value is probably already overwritten by the initialization of the variable with the old value of the field processed.

```
on assign of book.book-title old oldbook no-undo format "x(3)" label "GotIt" init "9999" do:
..
```

```
    oldbook = "1234567890".
    display oldbook with side-label.
end.
```

Will display: GotIt: 123 instead of ISBN: 1234567890 that is obtained without the extra "refinements".
So I believe, my solution to allow a regular_parm in OLD clause is correct in this case. I wonder if it both solutions can be merged ?

#21 - 11/28/2013 12:57 PM - Greg Shah

From the v11.0 documention for the TRIGGER PROCEDURE statement:

```
TRIGGER PROCEDURE FOR ASSIGN
   {    { OF table .field }
      | { NEW [ VALUE ] value1 { AS data-type | LIKE db-field } }
   }
   [ COLUMN-LABEL label ]
   [ FORMAT format-string ]
   [ INITIAL constant ]
   [ LABEL label-string ]
   [ NO-UNDO ]
   [ OLD [ VALUE ] value2 { AS data-type | LIKE db-field }
      [ COLUMN-LABEL label ]
      [ FORMAT format-string ]
      [ INITIAL constant ]
      [ LABEL label-string ]
      [ NO-UNDO ]
   ]
```

#22 - 11/28/2013 01:18 PM - Ovidiu Maxiniuc

Thanks, I see now.
I was blinded by the WRITE trigger that also allows new/old values. If no field is specified, then I assume the trigger is "generic" and can be fired by any field that matches the as / like datatype.
The only way connect this trigger to a database field is via data-dictionary so this syntax is not compatible with the session form.

#23 - 11/29/2013 08:25 AM - Ovidiu Maxiniuc

- File om_upd20131129a.zip added

Update for review.
I know it's not up-to date (to rev 10416), I will merge the changes probably on Monday.
Known issues:
Also the updates misses handling of NEW value in ASSIGN schema triggers and there is a wrong promotion of the OLD value in the session assign trigger. The latter is best visible in constructs like:

```
on assign of book.isbn old oldisbn do:
    on assign of book.publisher old oldpublisher do:
        message "assign in assign" oldisbn oldpublisher.
    end.
end.
```

#24 - 12/02/2013 02:17 PM - Ovidiu Maxiniuc

After more testing/investigations on the schema ASSIGN trigger syntax I came to following form:

```
TRIGGER PROCEDURE FOR ASSIGN
   {    { OF table.field }
      | { NEW [ VALUE ] value1
            { AS data-type | LIKE db-field }
            [ COLUMN-LABEL label ]
            [ FORMAT format-string ]
            [ INITIAL constant ]
            [ LABEL label-string ]
            [ NO-UNDO ]
         }
   }
   [ OLD [ VALUE ] value2
      [ AS data-type | LIKE db-field ]
      [ COLUMN-LABEL label ]
      [ FORMAT format-string ]
      [ INITIAL constant ]
      [ LABEL label-string ]
      [ NO-UNDO ]
   ]
```

The are two changes:
1. AS data-type | LIKE db-field is mandatory for NEW value1 and optional for value2 (if missing the type of table.field or value1 is used).
2. formatting clauses (FORMAT / INITIAL / NO-UNDO etc) are only valid for NEW value1, and not for table.field (that is already using the formatting from db schema).

The common part with session triggers is the whole OLD branch:

```
ON ASSIGN OF table.field
   [ OLD [ VALUE ] value2
      [ AS data-type | LIKE db-field ]
      [ COLUMN-LABEL label ]
      [ FORMAT format-string ]
      [ INITIAL constant ]
      [ LABEL label-string ]
      [ NO-UNDO ]
   ]
```

#25 - 12/06/2013 02:48 PM - Ovidiu Maxiniuc

- File om_upd20131206a.zip added

This update contains the merged progress.g from Greg with support for parsing TRIGGER PROCEDURES as described in note 24.
Known issue:
for:

```
on assign of book.isbn old oldisbn2 as char:
   DISPLAY "Triggered session ASSIGN book.isbn hidden".
end.
```

the initialization block

```
public void init()
{
    TestRoot.this.oldisbn2 = (character) oldisbn2;
}
```

is generated in wrong place (externalProcedure' Block) instead of inner class trigger Block. This happens only if there are more then one assign trigger declared so I suspect that the issue is the depot technique for building the init subtree before the trigger class is grafted.

#26 - 12/09/2013 06:35 PM - Greg Shah

Code Review 1206a

This is just a review of the progress.g changes. I am OK with the changes so far. Please make sure to add the missing error processing and add method javadoc where missing.

#27 - 12/09/2013 06:45 PM - Eric Faulhaber

Stanislav: please see the following excerpt from an email discussion I had with Ovidiu today:

> ECF: The update also includes a "src/com/goldencode/p2j/persist/trigger_old/" directory with a number of files; is this intentional?

> OM: The 'trigger_old' is the old implementation of database triggers. There is a 'NO_RECORD_AVAILABLE' trigger type defined that does not fit in Progress specifications. I didn't want to just drop it so I renamed the old package. Will this type of trigger be supported from now on ?

Wasn't this put in for a specific production problem we were trying to debug for Majic? Is it still in production? Is it still needed?

#28 - 12/09/2013 06:49 PM - Eric Faulhaber

I've looked at om_upd20131206a.zip. There is a lot there, but it's too difficult for me to understand in its current state. I'm going to wait until you have commented the headers, merged up to the latest revision, and removed/commented the debug code, then I'll try again.

One thing I noticed that was a bit confusing, perhaps you can help me understand the purpose: you generate a schema-triggers.xml file in schema/p2o.xml containing information about schema triggers. You later read that file, in convert/base_structure.xml and construct a hash map from its contents. When you process a trigger procedure, you pull information from that map.

Is this XML file needed at runtime for any reason, or is just an intermediate, conversion artifact? If the latter, why the separate file? Was there no (convenient) way to use existing TRPL features (e.g., annotations, artificial nodes, etc.) to achieve the same purpose?

Please upload your test cases (and any special instructions needed to convert them) with the next update. Since there is so much conversion-side change in this update, I'd like to convert your test cases and see what comes out, to help me understand the rule-set changes.

#29 - 12/09/2013 06:56 PM - Greg Shah

Please check in all test cases as you go.

#30 - 12/10/2013 04:38 PM - Ovidiu Maxiniuc

- File trigger_testcases_om_upd20131210.zip added

Attached large part of my test-cases for database triggers.

#31 - 12/11/2013 03:58 PM - Ovidiu Maxiniuc

- File om_upd20131211a.zip added

Uploaded update package. This version is merged with latest bzr revision.
Previously mentioned issues have been fixed.
It have been tested against the testcases from trigger_testcases_om_upd20131210.zip.

The schema-triggers.xml generated in the schema/p2o.xml is just an intermediary file needed for temporarily storage of attributes found in db.schema until the trigger-procedure files are processed. It is not needed at runtime because in convert/base_structure.xml the generated classes are annotated with needed information for loading an configuring them at runtime.
The intermediary file was the solution preferred to loading and scanning all databases p2o-s when processing the schema trigger procedures. My first intention was to annotate directly the .ast of the triggers when processing .schema, but at that very early moment they are not yet processed/created.

#32 - 12/12/2013 01:14 AM - Eric Faulhaber

Code review 20131211a (non-parser-related portions):

Wow, massive change. Good work! I haven't had time to integrate this into a development environment and run the test cases, but the changes look OK, AFAICT. A few notes:

- A general note: I appreciate that you are cleaning up the formatting of source code, correcting typos, etc. as you make other, functional updates in a source file. While it makes reviews a little harder because of the increased volume of change, it has a good long term impact on the code base.
- convert/java_templates.tpl is missing changes from bzr rev 10418 (from 2013-12-06).
- In p2o.xml, why is there one variable (xml_dbname) which does not follow the camelCase convention of all the other variables in the file? Not a huge deal, but the inconsistency is somewhat distracting.

- In variable_definitions.rules, line 890, you are removing quotes from the format string for all variable definitions with the "define_constant" annotation. Is this because the format string in your test case had an extra set of escape quotes? Are you sure this is the best place to remove them? I'm concerned the extra quotes may be specific to your conversion code path, but that this change may regress other cases that don't have extra quotes.
- buffer-definitions.rules: looks like some debug code left behind at line 259. Also, why add the implicit variable (line 181), if you only needed it once?
- database_general.rules: minor format change: please align parameters starting at line 422 and 432 with the open parentheses in the lines above.
- Are you sure snapshot is the appropriate variable in RecordBuffer to use for the oldDmoValue in trigger execution? Have you looked carefully at all the places snapshot is touched? I'm not saying it's wrong, just that it was created for a different use (as a placeholder between stateful query calls, like looping FINDs).
- RecordBuffer.getDMOImplementationClass: you added a @throws tag in the javadoc for PersistenceException, but the method doesn't throw it (it can throw the unchecked IllegalStateException from the checkActive call, however).
- What happens during undo processing of changes which caused triggers to fire? BTW, I noticed you removed the trigger firing from RecordBuffer$Reversible{Create|Delete}.rollbackWorker (which seem to be correct changes), but you left the comments about it behind.

#33 - 12/12/2013 06:36 AM - Stanislav Lomany

> Wasn't this put in for a specific production problem we were trying to debug for Majic? Is it still in production? Is it still needed?

Eric, I don't remember exactly why NO_RECORD_AVAILABLE was added, but most likely, yes, as far as I remember, it was related to specific Majic problem (I couldn't find traces of that problem).

#34 - 12/12/2013 12:56 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Code review 20131211a (non-parser-related portions):
>
> - Are you sure snapshot is the appropriate variable in RecordBuffer to use for the oldDmoValue in trigger execution? Have you looked carefully at all the places snapshot is touched? I'm not saying it's wrong, just that it was created for a different use (as a placeholder between stateful query calls, like looping FINDs).

You probably are right here. I am investigating this right now. I first looked if there is nothing similar to what I needed and this variable looked like the one. If I prove that this is not correct I will add a new one and the management of it.

> - What happens during undo processing of changes which caused triggers to fire? BTW, I noticed you removed the trigger firing from RecordBuffer$Reversible{Create|Delete}.rollbackWorker (which seem to be correct changes), but you left the comments about it behind.

Just double-checked. Nothing happens. The action the triggers have done cannot be undone by running the 'undo' triggers.

Regarding the NO_RECORD_AVAILABLE issue: I also scanned both projects (p2j and majic) and I could not find any trace of NO_RECORD_AVAILABLE nor of NoRecordAvailableTrigger implementation.

#35 - 12/12/2013 02:04 PM - Constantin Asofiei

> Regarding the NO_RECORD_AVAILABLE issue: I also scanned both projects (p2j and majic) and I could not find any trace of NO_RECORD_AVAILABLE nor of NoRecordAvailableTrigger implementation.

Ovidiu, the NoRecordAvailableTrigger was added some time ago to help debug some problems seen in MAJIC. Don't recall if this was helpful in the end or not. I think we should either keep our custom trigger types (which are configured in the directory) or make them configurable as the legacy ones.

#36 - 12/13/2013 04:42 AM - Ovidiu Maxiniuc

Ok, so I will merge back the custom trigger implementation and I will fire them with lower priority, that is, the 'legacy' triggers will be called before all other load triggers. There are no dump legacy triggers (on FIND).
The procedures for registration/unregistration and disabling them were never implemented so I guess when these triggers run, they didn't need such operations so I will not implement them.
I also could not find any rules for converting such kind of triggers so I assume they were written manually and also manually configured into registry, this is similar to a schema trigger.

#37 - 12/16/2013 03:21 PM - Ovidiu Maxiniuc

- File write-trigger-test5.p added

I attached a testcase. The expected output (from 4GL) is commented at the end of the file. I only can fire the WRITE trigger just before the P2J persistence.save(..);, and this will output:

```
1          Outer transaction start
2            Inside procedure start
3                Inner transaction start
4        Session trigger CREATE for
5                Inner transaction end
6        Session trigger WRITE for                  old
7            Inside procedure end
8          Outer transaction continue
9        Session trigger ASSIGN for 111              old
10       Session trigger CREATE for
11       Session trigger ASSIGN for 222              old
12       Session trigger WRITE for 222              old
13         Outer transaction end
14       Top scope continue
15       Session trigger FIND for 222
16       Session trigger DELETE for 222
17       Session trigger FIND for 111
18       Session trigger DELETE for 111
19       Top scope end
```

It looks like the 4GL flush to disk (the WRITE db event) happens very late, when the buffer must be re-populated with other record (brabd new from CREATE from line 10 or loaded from db in the case of FIND from line 15).

The P2J is too eager to persist the currently stored data in the buffer so it will flush each time a transaction block ends (line 6 for DO TRANSACTION and line 12 for procedure scope end).

#38 - 12/17/2013 12:18 PM - Eric Faulhaber

It does appear you are correct in your conclusion that we are validating too early. Did you confirm that P2J is flushing in response to the transaction block end, or is this an educated guess, based on the 4GL code? I ask because I would expect the scope of your outer transaction to be expanded to the scope of the external procedure (because of the book buffer access outside of the do block). You can confirm/refute this with a compile listing.

It wouldn't make sense if the write trigger were being fired outside the transaction scope, because you are supposed to be able to validate and edit a buffer in a write trigger, and then have those changes persisted. So, the test case output may be a bit misleading in this sense.

The question is: are we misinterpreting the transaction scope expansion during conversion, or do we have a runtime error that is causing the flush to happen too early?

#39 - 12/18/2013 01:46 AM - Ovidiu Maxiniuc

- File write-trigger-test5.p.lst added

I believe this is the listing of the previously attached file. It's the 1st time I have done this, I didn't know this was possible! Looks interesting but I'm not sure I understand the buffer scopes.

P2J, on the other hand, when a block ends and isTransaction(), BlockManager will call processValidate() that will eventually flush() the current record from all Commitable of the 'peek' block.

#40 - 01/07/2014 01:19 AM - Eric Faulhaber

The following trigger-related methods are present but not implemented in BufferImpl:

```
public void openReadOnlyScope()
public void disableDumpTriggers()
public void disableLoadTriggers(boolean allowReplication)
```

What is the intention for these?

#41 - 01/07/2014 01:04 PM - Eric Faulhaber

From Ovidiu's email response:

The openReadOnlyScope() is not needed any more. The OLD buffer is built in DatabaseTriggerManager directly as readonly. The other two, disableDumpTriggers() and disableLoadTriggers(boolean allowReplication) the implementation will be provided with today's update.

[ECF: update will be posted to #2217]

#42 - 01/10/2014 01:05 PM - Eric Faulhaber

- % Done changed from 0 to 100

- Status changed from WIP to Closed

Base trigger support is finished. Several bugs have been found and are being worked as separate tasks.

#43 - 01/30/2014 04:09 PM - Eric Faulhaber

- File om_upd20140130b.zip added

- Status changed from Closed to WIP

Temporarily re-opening this issue to post Ovidiu's update to fix some problems with trigger conversion. Ovidiu, please update this task with a brief summary of the problems you addressed. Once this update has passed conversion regression test (i.e., should have no conversion impact on the regression test environment), and has been checked into bzr, I will re-close this issue.

I've reviewed the update and it looks good.

#44 - 02/03/2014 08:13 AM - Ovidiu Maxiniuc

I wasn't expecting special issues as the update only affects schema triggers and majic does not use them.
Surprisingly, I had to run the conversion 3 times for this update.
First time there was a JVM crash. Unfortunately, I lost the log file.
The second time, the conversion ended with the following:

```
     [java] ./src/syman/adp/adpout0.p
     [java] Elapsed job time:  00:00:01.802
     [java] ERROR:
     [java] java.lang.RuntimeException: ERROR!  Active Rule:
     [java] ------------------------
     [java]       RULE REPORT
     [java] ------------------------
     [java] Rule Type :   WALK
     [java] Source AST:  [ next ] BLOCK/INNER_BLOCK/KW_REPEAT/KW_ON/KW_NEXT/ @551:24 {17179873192}
     [java] Copy AST  :  [ next ] BLOCK/INNER_BLOCK/KW_REPEAT/KW_ON/KW_NEXT/ @551:24 {17179873192}
     [java] Condition :  evalLib("classNameConflict", basePkg, "Action")
     [java] Loop      :  false
     [java] --- END RULE REPORT ---
     [java]
     [java]     at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:863)
     [java]     at com.goldencode.p2j.convert.ConversionDriver.processTrees(ConversionDriver.java:931)
     [java]     at com.goldencode.p2j.convert.ConversionDriver.back(ConversionDriver.java:835)
     [java]     at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1756)
     [java] Caused by: java.lang.NoClassDefFoundError: com/goldencode/p2j/library/NativeAPIEntry
     [java]     at com.goldencode.p2j.convert.NameMappingWorker$Library.classNameExists(NameMappingWorker.java
:291)
     [java]     at com.goldencode.expr.CE14226.execute(Unknown Source)
     [java]     at com.goldencode.expr.Expression.execute(Expression.java:341)
     [java]     at com.goldencode.p2j.pattern.NamedFunction.execute(NamedFunction.java:387)
     [java]     at com.goldencode.p2j.pattern.AstSymbolResolver.execute(AstSymbolResolver.java:646)
     [java]     at com.goldencode.p2j.pattern.CommonAstSupport$Library.execLib(CommonAstSupport.java:1097)
     [java]     at com.goldencode.p2j.pattern.CommonAstSupport$Library.evalLib(CommonAstSupport.java:1073)
     [java]     at com.goldencode.expr.CE14225.execute(Unknown Source)
     [java]     at com.goldencode.expr.Expression.execute(Expression.java:341)
     [java]     at com.goldencode.p2j.pattern.Rule.apply(Rule.java:401)
     [java]     at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:640)
     [java]     at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:609)
     [java]     at com.goldencode.p2j.pattern.Rule.apply(Rule.java:440)
     [java]     at com.goldencode.p2j.pattern.RuleContainer.apply(RuleContainer.java:531)
     [java]     at com.goldencode.p2j.pattern.RuleSet.apply(RuleSet.java:50)
     [java]     at com.goldencode.p2j.pattern.RuleContainer.apply(RuleContainer.java:531)
     [java]     at com.goldencode.p2j.pattern.RuleSet.apply(RuleSet.java:50)
     [java]     at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:213)
     [java]     at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:160)
     [java]     at com.goldencode.p2j.pattern.PatternEngine.apply(PatternEngine.java:1350)
     [java]     at com.goldencode.p2j.pattern.PatternEngine.processAst(PatternEngine.java:1248)
```

```
     [java]     at com.goldencode.p2j.pattern.PatternEngine.processAst(PatternEngine.java:1196)
     [java]     at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:832)
     [java]     ... 3 more
     [java] Caused by: java.lang.ClassNotFoundException: com.goldencode.p2j.library.NativeAPIEntry
     [java]     at java.net.URLClassLoader$1.run(URLClassLoader.java:366)
     [java]     at java.net.URLClassLoader$1.run(URLClassLoader.java:355)
     [java]     at java.security.AccessController.doPrivileged(Native Method)
     [java]     at java.net.URLClassLoader.findClass(URLClassLoader.java:354)
     [java]     at java.lang.ClassLoader.loadClass(ClassLoader.java:425)
     [java]     at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
     [java]     at java.lang.ClassLoader.loadClass(ClassLoader.java:358)
     [java]     ... 26 more
```

The 3rd time it went fine and there no conversion issues, generated sources being unchanged.

#45 - 02/03/2014 10:12 AM - Ovidiu Maxiniuc

The om_upd20140130b.zip update fixes a schema-trigger conversion bug when the project contains access to temp-tables.
Normally, p2o.xml extracts/resolves the references to schema triggers found in all databases of the project in a file named schema-triggers.xml, file that is used for annotations later, when generating the trigger classes. The file collects all triggers from all processed trees/databases and is saved to disk in global post-rules.
Because temp-tables are processed in a second run of p2o.xml, the original file was overwritten with an empty version (because temp-tables do not support triggers at all), so when building classes in base_structure.xml, the references could not be found in the overwritten file, so the trigger classes were generated without correct trigger annotations.
The solution was to identify the second run based on the name of the tree (only _temp is processed) and skip the persistence rule. So the file is always freshened at each 1st step of the conversion and its content is not trashed by _temp database.

I knew that there are 4243 schema triggers defined in <db_name>.df and 4215 trigger procedures. During this investigation I could identify 4212 correct associations, leading to 3 trigger procedures not associated with any DB object and 31 triggers declared without actual implementation.

#46 - 02/17/2014 10:05 AM - Ovidiu Maxiniuc

- File om_upd20140217a.zip added

I have merged the update with latest bzr version. The conversion testing passed (as expected, no changes in generated code). Only conversion rules are modified, the runtime is untouched.
I am going to release this update to bzr.

#47 - 02/19/2014 02:35 PM - Ovidiu Maxiniuc

Committed to bzr as revision 10472 and distributed by mail.

#48 - 02/20/2014 12:59 PM - Eric Faulhaber

- Status changed from WIP to Closed

#49 - 11/16/2016 11:42 AM - Greg Shah

- Target version changed from Milestone 7 to Runtime Support for Server Features

Files

progress_parser_ges_upd20131126.zip	268 KB	11/28/2013	Greg Shah
om_upd20131129a.zip	400 KB	11/29/2013	Ovidiu Maxiniuc
om_upd20131206a.zip	726 KB	12/06/2013	Ovidiu Maxiniuc
trigger_testcases_om_upd20131210.zip	17.7 KB	12/10/2013	Ovidiu Maxiniuc
om_upd20131211a.zip	668 KB	12/11/2013	Ovidiu Maxiniuc
write-trigger-test5.p	2.36 KB	12/16/2013	Ovidiu Maxiniuc
write-trigger-test5.p.lst	3.76 KB	12/18/2013	Ovidiu Maxiniuc
om_upd20140130b.zip	39.2 KB	01/30/2014	Eric Faulhaber
om_upd20140217a.zip	39.5 KB	02/17/2014	Ovidiu Maxiniuc