Database - Feature #1949

Feature # 1658 (Closed): improve database trigger support

add conversion support for database triggers

01/14/2013 04:45 PM - Eric Faulhaber

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Stanislav Lomany	**% Done:**	100%
Category:		**Estimated time:**	254.00 hours
Target version:	Conversion Support for Server Features		
billable:	No	**vendor_id:**	GCD

Description

History

#1 - 01/14/2013 05:10 PM - Eric Faulhaber

We need to support conversion of trigger procedures and ON statements for the following database triggers:

- DELETE
- WRITE
- CREATE
- FIND

We will need a mechanism to register database triggers defined with ON blocks in a similar manner to what we do with UI triggers, with a database-trigger-specific manager class. Greg: please comment/correct if I am misstating the requirement.

In addition, the following methods and attributes must convert and compile.

- DISABLE-LOAD-TRIGGERS (KW_DIS_L_T)

Runtime support should be minimal, enough to compile only (i.e., stubbed out).

Although we ultimately will need a mechanism to extract trigger information from the schema and associate these bindings with the appropriate trigger procedures (i.e., in a jar resource or in the directory), this is not necessary currently to get the converted code to compile, so it should be deferred to the parent task.

#2 - 01/15/2013 04:53 PM - Stanislav Lomany

- Status changed from New to WIP

Eric, am I correct that I only need to cover session triggers, i.e.

```
ON event OF database-object
[ referencing-phrase ]
```

```
[ OVERRIDE ]
{ trigger-block | REVERT }
```

and leave schema triggers (external procedures) for now?

DISABLE-LOAD-TRIGGERS was introduced in Progress 9.1D. BTW, do we have documentation for this version?

Do I need to convert DISABLE TRIGGERS statement?

#3 - 01/15/2013 09:21 PM - Eric Faulhaber

We need to convert the session triggers (like your ON example above), but we also need support for schema triggers (i.e., trigger procedures). But for now, we only need the conversion of the external procedure. For example:

```
TRIGGER PROCEDURE FOR WRITE OF my-table.
...
```

The effort to manage the associations of triggers in the schema to trigger procedures should be deferred.

Yes, we also need to convert the DISABLE TRIGGER statement.

#4 - 01/16/2013 06:56 AM - Greg Shah

In regard to the TRIGGER PROCEDURE conversion, it seems like the external procedure can be converted to something with a very specific signature. In this case:

```
TRIGGER PROCEDURE FOR WRITE OF my-table.
...
```

instead of execute(), perhaps we emit something like this:

```
public void writeTrigger(MyTable myTable)
{
   ...
}
```

I don't think you can call these procedures using a RUN statement so I think it is safe to move to a custom signature for what is still the "external procedure".

In regards to the 4GL documentation, you can download 9.1E, the 10.x and even the 11.0 docs from Progress.

#5 - 01/18/2013 12:30 PM - Stanislav Lomany

About the ways of conversion:
1. The following statements:

```
TRIGGER PROCEDURE FOR CREATE OF table
TRIGGER PROCEDURE FOR DELETE OF table
TRIGGER PROCEDURE FOR FIND OF table
```

will be converted in this way:

```
public class SomeProc
{
   public void executeCreate(final Book book)
   {
      RecordBuffer.openScope(book);
      ...
   }
}
```

Note that code that normally goes to init() section will go to the beginning of the execute* function.

2.

```
TRIGGER PROCEDURE FOR WRITE OF table
[ NEW [ BUFFER ] buffer-name1 ]
[ OLD [ BUFFER ] buffer-name2 ]
```

will be converted to:

```
public class SomeProc
{
   public void executeWrite(final Book book1, final Book book2)
   {
      RecordBuffer.openScope(book1);
      RecordBuffer.openScope(book2);
      ...
   }
}
```

If new buffer name is specified, then it will be used as the first parameters, otherwise default buffer name will be used.
The old buffer should be read-only.

3.

```
DISABLE TRIGGERS FOR { DUMP | LOAD } OF table-name [ ALLOW-REPLICATION ]
```

will be converted using

```
com.goldencode.p2j.persist.trigger.DatabaseTriggerManager.disableDumpTriggers(DataModelObject buffer)
or
com.goldencode.p2j.persist.trigger.DatabaseTriggerManager.disableLoadTriggers(DataModelObject buffer, boolean
allowReplication)
```

4.

Note that at this point we do not support handles for record buffers.

```
DISABLE-DUMP-TRIGGERS
and
DISABLE-LOAD-TRIGGERS
```

will be converted to

```
((RecordBuffer) h.get()).disableLoadTriggers(true/false);
or
((RecordBuffer) h.get()).disableDumpTriggers;
```

Guys, is it OK?
ON statement conversion will follow up.

#6 - 01/18/2013 01:58 PM - Greg Shah

In regard to the TRIGGER PROCEDURE conversion:

```
public class SomeProc
{
   public void executeCreate(final Book book)
   {
      RecordBuffer.openScope(book);
      ...
   }
}
```

I prefer a different name that makes it more obvious this is a trigger, like triggerCreate() instead of executeCreate().

Also, it seems to me that we shoujld treat this like an external procedure. It should call the BlockManager.externalProcedure() and pass the Block instance like normal. As far as I know there are few limits in what can be done in a trigger procedure. You are discouraged from doing interactive stuff, but it is not prohibited:

```
Some 3GL applications execute schema triggers. Triggers might also be executed in batch
mode. Therefore, you should avoid any user-interface interactions within schema trigger
procedures.
```

In fact, the 4GL docs have an example that shows a MESSAGE statement in a TRIGGER PROCEDURE. If you can do interactive stuff, you can do anything, including stream I/O, redirected terminal... these are all things that could block, that need full scope processing, that could have transaction processing. I think we need to implement the full block manager stuff as normal.

Eric will respond on the buffer handle parts.

#7 - 01/21/2013 03:47 PM - Stanislav Lomany

Eric, for the write trigger we need to have read-only buffer parameter which represents the old state of the buffer. This buffer can be updated, but no changes will be committed. How can we implement it in conversion part? Should we call some dmo.openScope(true) function (true parameter indicates that it is a read-only scope) in the beginning of the target scope (i.e. external procedure or a trigger block)?

#8 - 01/21/2013 03:57 PM - Eric Faulhaber

Why don't we go with dmo.openReadOnlyScope()? I can add this to the new Buffer interface and BufferImpl class (see #1967).

I have not added a "regular" openScope() method to the Buffer interface, since we typically open multiple scopes at the same time, using RecordBuffer.openScope(DataModelObject...), and I probably will leave it that way for now. If I understand triggers, each operates on a single buffer at a time, right? So, there would be no reason to add a RecordBuffer.openReadOnlyScope(DataModelObject...) method, right?

#9 - 01/21/2013 04:09 PM - Stanislav Lomany

> Why don't we go with dmo.openReadOnlyScope()? I can add this to the new Buffer interface and BufferImpl class (see #1967).
>
> I have not added a "regular" openScope() method to the Buffer interface, since we typically open multiple scopes at the same time, using RecordBuffer.openScope(DataModelObject...), and I probably will leave it that way for now. If I understand triggers, each operates on a single buffer at a time, right? So, there would be no reason to add a RecordBuffer.openReadOnlyScope(DataModelObject...) method, right?

Yes, you a correct. Let's go with openReadOnlyScope.

#10 - 01/21/2013 05:26 PM - Stanislav Lomany

Is this conversion correct (part 1)?

1.

```
TRIGGER PROCEDURE FOR CREATE OF table
TRIGGER PROCEDURE FOR DELETE OF table
TRIGGER PROCEDURE FOR FIND OF table
```

to:

```
public class ExternalProc
{
   Book.Buf buf;

   public void triggerCreate(final Book.Buf buf)
   {
      externalProcedure(new Block()
      {
         public void body()
         {
            RecordBuffer.openScope(buf);
            ExternalProc.this.buf = buf;
```

2.

```
TRIGGER PROCEDURE FOR WRITE OF table
[ NEW [ BUFFER ] buffer-name1 ]
[ OLD [ BUFFER ] buffer-name2 ]
```

to:

```
public class ExternalProc
{
   Book.Buf buf1;

   Book.Buf buf2;

   public void triggerWrite(final Book.Buf buf1, final Book.Buf buf2)
   {
      externalProcedure(new Block()
      {
         public void body()
         {
            RecordBuffer.openScope(buf1);
            buf2.openReadOnlyScope();
            ExternalProc.this.buf1 = buf1;
            ExternalProc.this.buf2 = buf2;
```

3.

```
DISABLE TRIGGERS FOR { DUMP | LOAD } OF table-name [ ALLOW-REPLICATION ]
```

to:

```
dmo.disableDumpTriggers()
or
dmo.disableLoadTriggers(boolean allowReplication)
```

4.

```
DISABLE-DUMP-TRIGGERS
and
DISABLE-LOAD-TRIGGERS
```

to:

```
((Buffer) h.get()).disableLoadTriggers(true/false);
or
((Buffer) h.get()).disableDumpTriggers;
```

#11 - 01/21/2013 05:38 PM - Eric Faulhaber

Stanislav Lomany wrote:

> Is this conversion correct (part 1)?

In cases 1 and 2, the openScope/openReadOnlyScope calls and the assignments to the external procedure's Buf* instance variables should be in the Block's init() method, rather than in the body() method. These calls must only happen once for the block.

Cases 3 and 4 look correct.

#12 - 01/21/2013 05:59 PM - Stanislav Lomany

> In cases 1 and 2, the openScope/openReadOnlyScope calls and the assignments to the external procedure's Buf* instance variables should be in the Block's init() method, rather than in the body() method. These calls must only happen once for the block.

At this point buffer scopes for external procedures and triggers are opened in the body() method (or have we fixed it recently?).
Should I fix it for all (not only DB triggers related) procedures and triggers if it will be easy?

#13 - 01/21/2013 06:40 PM - Eric Faulhaber

Hmm... you're right. Greg and I can't seem to remember why this is, but leave it the way you initially proposed for now.

Does this apply to the Buf* instance variable assignments as well? I can't find an example of this at the moment.

#14 - 01/22/2013 07:20 AM - Stanislav Lomany

Eric Faulhaber wrote:

> Hmm... you're right. Greg and I can't seem to remember why this is, but leave it the way you initially proposed for now.
>
> Does this apply to the Buf* instance variable assignments as well? I can't find an example of this at the moment.

Yes.

#15 - 01/22/2013 08:23 AM - Stanislav Lomany

Is this conversion correct (part 2)?

5.
The main decision about session triggers is how we going to pass target buffer(s) to the trigger block. The buffer(s) are the instance variables of the trigger block (they can override buffers with the same name in the parent procedure class). My suggestion is to implicitly assign them upon trigger initialization. I.e.:

```
ON event OF table
[ NEW [ BUFFER ] buffer-name1 ]
[ OLD [ BUFFER ] buffer-name2 ]
[ OVERRIDE ]
trigger-block
```

will be converted to:

```
registerTrigger(DatabaseTriggerType.WRITE,
                TriggerBlock0.class,
                ExternalProc.this,
                Book.class,   /* target table */
                "book2",      /* second buffer name (for write triggers only) */
                true/false);  /* override schema trigger */
...
public class TriggerBlock0
extends Trigger
{
   Book.Buf book;               // assigned by runtime during trigger init

   Book.Buf book2;              // assigned by runtime during trigger init

   public void body()
   {
      RecordBuffer.openScope(book);
      book2.openReadOnlyScope();
      ...
   }
}
```

6.

```
ON event OF table
[ NEW [ BUFFER ] buffer-name1 ]
[ OLD [ BUFFER ] buffer-name2 ]
[ OVERRIDE ]
REVERT
```

to:

```
deregisterTrigger(DatabaseTriggerType.WRITE, Book.class);
```

#16 - 01/22/2013 10:13 AM - Eric Faulhaber

Stanislav Lomany wrote:

> Is this conversion correct (part 2)?

Why are we passing the name "book2" as a string to registerTrigger, instead of as a Book.Buf proxy instance? Also, is it necessary to pass in the target table type Book.class (btw, why not Book.Buf.class)? Can we not infer the type from the book2 proxy instance? Or, are you saying book2 is only a parameter for write triggers, but we always need the target table type for all triggers?

Other than that, looks good to me. Greg?

#17 - 01/22/2013 11:15 AM - Greg Shah

registerTrigger() is in the LogicalTerminal, which is not something that should be involved with database triggers. I don't know of a reason to reuse this. I prefer to have a database-specific trigger mechanism (like DatabaseTriggerManager.registerDatabaseTrigger()).

I also wonder about the "book2" String parameter. It seems related to the assignment of the 2nd buffer by the runtime. I presume that the first buffer is always assigned?

I would prefer an approach that implemented a trigger method with buffer parameters:

```
registerDatabaseTrigger(DatabaseTriggerType.WRITE,
                        TriggerBlock0.class,
                        ExternalProc.this,
                        true/false);  /* override schema trigger */
...
public class TriggerBlock0
extends DatabaseTrigger
{
   public void write(final Book.Buf book, final Book.Buf book2)
   {
      RecordBuffer.openScope(book);
      book2.openReadOnlyScope();
      ...
   }
}
```

The deregistration would be changed to deregisterDatabaseTrigger().

#18 - 01/22/2013 03:40 PM - Stanislav Lomany

> Why are we passing the name "book2" as a string to registerTrigger, instead of as a Book.Buf proxy instance? Also, is it necessary to pass in the target table type Book.class (btw, why not Book.Buf.class)? Can we not infer the type from the book2 proxy instance? Or, are you saying book2 is only a parameter for write triggers, but we always need the target table type for all triggers?

Any trigger uses as input parameter buffer which contains the changed DMO. For write trigger we can have the second buffer which contains the previous version of the DMO. These buffers are scoped to the trigger ONLY, i.e. it is not the buffers that used in the external procedure (even if the have the same name), so we cannot pass proxy.

> btw, why not Book.Buf.class

Mistype.

#19 - 01/22/2013 03:49 PM - Stanislav Lomany

```
registerDatabaseTrigger(DatabaseTriggerType.WRITE,
TriggerBlock0.class,
ExternalProc.this,
true/false);  /* override schema trigger */
```

Greg, triggers are registered per event type and target table, so we probably need to specify the target table or DMO class in the registerDatabaseTrigger (unless you want to extract it from the trigger class definitions).

> The deregistration would be changed to deregisterDatabaseTrigger().

The event type and target table should be specified.

```
public class TriggerBlock0
extends DatabaseTrigger {
public void write(final Book.Buf book, final Book.Buf book2)
```

OK, does DatabaseTrigger define some interface?

#20 - 01/22/2013 04:42 PM - Greg Shah

```
 triggers are registered per event type and target table, so we probably need to specify the target table or D
MO class in the registerDatabaseTrigger (unless you want to extract it from the trigger class definitions).
```

Yes, you're right.

```
    The deregistration would be changed to deregisterDatabaseTrigger().

The event type and target table should be specified.
```

Yes. I was really just referring to the method name.

```
OK, does DatabaseTrigger define some interface?
```

I'm not sure we need an interface, but I was thinking it might have an empty implementation for each of the trigger event types. Perhaps generics can be used to make this work?

#21 - 01/23/2013 12:57 PM - Stanislav Lomany

> I'm not sure we need an interface, but I was thinking it might have an empty implementation for each of the trigger event types. Perhaps generics can be used to make this work?

My thought is that it shouldn't define an interface or have empty implementations (what is the reasoning?). And it probably should look like this:

```
public class TriggerBlock0
extends DatabaseTrigger
{
   Book.Buf book;

   Book.Buf book2;

   public void write(final Book.Buf book, final Book.Buf book2)
   {
      trigger(new Block()         // static method of DatabaseTrigger, calls topLevelBlock
```

```
      {
         public void body()
         {
            RecordBuffer.openScope(book);
            book2.openReadOnlyScope();
            TriggerBlock0.this.book = book;
            TriggerBlock0.this.book2 = book2;
```

Also, validators, put expressions and nested trigger classes should be nested inside the database trigger class, not the parent procedure class. Should this be implemented?

#22 - 01/23/2013 01:42 PM - Greg Shah

```
My thought is that it shouldn't define an interface or have empty implementations (what is the reasoning?).
```

The primary advantage is that the runtime would be able to call the proper methods without any use of reflection.

I was thinking that generics might help:

```
public class DatabaseTrigger<B extends Buffer.Buf>
extends Trigger
{
   public void write(final B buf1, final B buf2) { }

   public void create(final B buf1) { }

...
}
```

The generated code might look something like this:

```
public class TriggerBlock0
extends DatabaseTrigger<Book.Buf>
{
   public void write(final Book.Buf book, final Book.Buf book2)
   {
      trigger(new Block()
      {
         public void body()
         {
            RecordBuffer.openScope(book);
            book2.openReadOnlyScope();
```

A question: do we really need the Book.Buf book and Book.Buf book2 members? It seems like the final parameters should be enough. Perhaps I am forgetting something.

#23 - 01/23/2013 02:14 PM - Stanislav Lomany

> The primary advantage is that the runtime would be able to call the proper methods without any use of reflection.

WRITE can have a single parameter (new buffer) or two parameters (new buffer + old buffer). So we probably need to specify which overloaded function will be used upon trigger registration, because we need to know whether we need to generate the old buffer for a trigger.

> A question: do we really need the Book.Buf book and Book.Buf book2 members? It seems like the final parameters should be enough. Perhaps I am forgetting something.

These members can be used in nested classes (validators, put expressions and nested triggers).

#24 - 01/23/2013 02:55 PM - Greg Shah

OK, go ahead with this approach.

#25 - 02/14/2013 06:10 PM - Stanislav Lomany

- File svl_test20130213a.zip added

Test update, doesn't handle trigger deregistration and DISABLE-*-TRIGGERS functions.

#26 - 02/15/2013 04:17 PM - Eric Faulhaber

Code review 20130213a:

Overall, this first drop looks quite good, though honestly it's such a big set of changes it's hard to evaluate just from looking at the code. Some questions/comments:

- Do you have test cases? Please provide a separate update if those are ready.
- There are many rules changes that affect procedures and record scoping. Have you converted Majic to ensure the converted code hasn't been broken?
- The static [de]registration functions are still in LogicalTerminal, rather than DatabaseTriggerManager. Is this necessary? Is it possible for a database trigger to operate in the context of a WAIT-FOR statement? If not, database trigger processing should be managed from a database-specific class, separate from the UI.
- I don't quite follow the changes in ui_statements.rules with respect to persistent triggers. Constantin: can you review this piece, please? There were some other changes in this file, one having to do with LogicalTerminal.waitFor, but that may be due to your baseline being older than the latest version I have from bzr.
- You modified some pretty heavily used functions in common-progress.rules (explicit_buffer_def_rel and record_types_rel). Did you check for side effects? I guess comparing Majic conversion output with the previous build's output as noted above would go a long way to doing that.

#27 - 02/16/2013 08:01 AM - Constantin Asofiei

ui_statements.rules looks OK (is built on top of H081); the only comment is about the "== true" and "!= true" tests.

A note about WAIT-FOR statement: at this time, WAIT-FOR I know it can listen for UI, Sockets and DB events. When implementing WAIT-FOR support for these specific events, please check if there are mixed cases in the project; my concern is that our UI event listening loop is on the client side, the other events are generated on the server, and when we mix them togheter, things will get complicated.

#28 - 02/16/2013 08:41 AM - Greg Shah

There are no mixed cases in the server project. I don't have a good check for the GUI right now, but that is OK.

I understand that the mixed case is a problem and I am willing to defer dealing with that at this time. While I understand that the sockets and UI events are both handled in WAIT-FOR (which in 4GL is all on the same client system and in our architecture is split). But are you sure that WAIT-FOR can "see" DB events?

As far as I understand it, the DB events processing is pretty separate. For example, the ON statement does NOT mix the DB events with widget events. Your trigger must be for one or the other, but not both.

If the DB events were dependent upon WAIT-FOR, then how could they execute during CREATE/FIND/ASSIGN...? These are all operations that occur when WAIT-FOR is NOT active. That suggests to me that DB events cannot be processed in WAIT-FOR OR that any pending WAIT-FOR would have to have been suspended and we are processing a nested trigger. So if a notification event was passed to the client, it would just sit in the queue and would not be processed until after the rest of the nested trigger returned. But this is all just speculation. We need some better information on the limits here.

What I prefer is to completely separate the events processing. I don't want the DB events processing or DB trigger registration/deregistration to have anything to do with the UI trigger manager or anything in the ui package. If we need to reuse the trigger manager processing, then we may need to refactor it into a base class.

#29 - 02/16/2013 08:54 AM - Constantin Asofiei

> But are you sure that WAIT-FOR can "see" DB events?

I'm mistaken on this, I was doing a wait-for find of person.first-name and it worked as person.first-name was in a frame (i.e. was referencing a widget).

#30 - 02/16/2013 03:27 PM - Stanislav Lomany

- Do you have test cases? Please provide a separate update if those are ready.

I'll provide them.

- There are many rules changes that affect procedures and record scoping. Have you converted Majic to ensure the converted code hasn't been broken?

I'll make it this night.

- The static [de]registration functions are still in LogicalTerminal, rather than DatabaseTriggerManager. Is this necessary?

No.

Is it possible for a database trigger to operate in the context of a WAIT-FOR statement?

No.

If not, database trigger processing should be managed from a database-specific class, separate from the UI.

OK.

- I don't quite follow the changes in ui_statements.rules with respect to persistent triggers. Constantin: can you review this piece, please? There were some other changes in this file, one having to do with LogicalTerminal.waitFor, but that may be due to your baseline being older than the latest version I have from bzr.
- You modified some pretty heavily used functions in common-progress.rules (explicit_buffer_def_rel and record_types_rel). Did you check for side effects? I guess comparing Majic conversion output with the previous build's output as noted above would go a long way to doing that.

#31 - 02/18/2013 06:13 PM - Stanislav Lomany

- File svl_upd20130219a.zip added

#32 - 02/20/2013 02:50 PM - Stanislav Lomany

- File svl_upd20130219a.zip added

#33 - 02/20/2013 02:55 PM - Stanislav Lomany

TODO

This update doesn't handle the case if an UI trigger is 1. nested into DB trigger and 2. references DB trigger input parameter. Like that:

```
on create of book do:
   on "d" anywhere do:
      display book with frame f1.
   end.
   ...
end.
```

#34 - 02/21/2013 01:30 PM - Stanislav Lomany

- Status changed from WIP to Review

#35 - 02/21/2013 01:41 PM - Eric Faulhaber

- % Done changed from 0 to 100

- Status changed from Review to Closed

#36 - 02/22/2013 08:16 AM - Stanislav Lomany

- File svl_testcase20130219a.zip added

#37 - 02/27/2013 05:24 AM - Constantin Asofiei

Problem with DB triggers in server project:

```
on delete of book override do: end.
```

```
EXPRESSION EXECUTION ERROR:
----------------------------
undoLoc = #(int) getNoteLong("numUndoLocal")
          ^  { java.lang.NullPointerException }
----------------------------
ERROR:
java.lang.RuntimeException: ERROR!  Active Rule:
-----------------------
      RULE REPORT
-----------------------
Rule Type :   WALK
Source AST:  [ block ] BLOCK/STATEMENT/KW_ON/TRIGGER_BLOCK/BLOCK/ @0:0 {12884901906}
Copy AST  :  [ block ] BLOCK/STATEMENT/KW_ON/TRIGGER_BLOCK/BLOCK/ @0:0 {12884901906}
Condition :  undoLoc = #(int) getNoteLong("numUndoLocal")
Loop      :  false
--- END RULE REPORT ---

    at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:732)
```

```
    at com.goldencode.p2j.convert.ConversionDriver.processTrees(ConversionDriver.java:948)
    at com.goldencode.p2j.convert.ConversionDriver.back(ConversionDriver.java:852)
    at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1729)
Caused by: com.goldencode.expr.ExpressionException: Expression execution error @1:11 [BLOCK id=12884901906]
```

#38 - 02/27/2013 09:19 AM - Ovidiu Maxiniuc

- File om_upd20130227a.zip added

Looks like for trigger blocks, the undoable counter are not initialized (written as annotation).
The attached proposed update tests if they are not, and the value 0 (zero) is used instead.

#39 - 02/27/2013 09:40 AM - Ovidiu Maxiniuc

Indeed, the annotations are not set unless some variables are first declared. And, because in our sample the block is empty, no variables declared those annotation are missing.
So the provided solution is proved to be correct: if annotations are not present means that no variables were declared so their count is 0.

#40 - 02/27/2013 09:41 AM - Greg Shah

The code looks OK. We need to conversion test it.

#41 - 02/27/2013 09:45 AM - Greg Shah

I have started conversion testing.

#42 - 02/27/2013 12:16 PM - Greg Shah

Conversion testing passed. Please check in the update and distribute it.

#43 - 02/27/2013 01:45 PM - Greg Shah

Passed conversion testing and is checked in as bzr revision 10221.

#44 - 02/28/2013 08:25 AM - Ovidiu Maxiniuc

During the investigations for #1985, note 69, resulted that a possible bug is in the parser.
The following code:

```
ON WRITE OF Customer DO: END.
```

will convert just fine. Here is the .p.parser output (the .p.ast is similar):

```
ON [KW_ON] @13:1
   database event [DB_EVENT] @0:0
      WRITE [KW_WRITE] @13:4
   OF [KW_OF] @13:10
   customer [TABLE] @13:13
   trigger block [TRIGGER_BLOCK] @0:0
      DO [KW_DO] @13:22
      block [BLOCK] @0:0
```

However, if the code changes to:

```
TRIGGER PROCEDURE FOR WRITE OF customer NEW BUFFER customer OLD BUFFER old_customer.
   ...
ON WRITE OF customer DO: END.
```

The conversion will fail with a NPE somewhere in ui_statements.rules:2230.
The reason is that the second time Customer is treated as BUFFER instead of TABLE:

```
ON [KW_ON] @13:1
   database event [DB_EVENT] @0:0
      WRITE [KW_WRITE] @13:4
   OF [KW_OF] @13:10
   customer [BUFFER] @13:13
   trigger block [TRIGGER_BLOCK] @0:0
      DO [KW_DO] @13:22
      block [BLOCK] @0:0
```

For some reason the symbol definitions from the previous TRIGGER statement into ON statement.
Stanislav,
do you have any idea why this happens ? Is this intentional ?

#45 - 02/28/2013 09:06 AM - Stanislav Lomany

> do you have any idea why this happens ? Is this intentional ?

No. No. I'll take it as my top priority.

#46 - 03/01/2013 03:29 AM - Stanislav Lomany

```
TRIGGER PROCEDURE FOR WRITE OF book NEW BUFFER book.
```

declares a buffer which name matches the table name. The same effect can be achieved with (Progress allows such notation):

```
define buffer book for book.
```

There are several options to solve this. The main ones are: 1. do not declare a buffer in this case. 2. force ON DB_EVENT statement to use TABLE instead of BUFFER by 2.1 modifying record grammar rule by adding the possibility to forcibly specify the target record type OR 2.2 replace BUFFER with TABLE later in .rules.

I don't know what to prefer.

#47 - 03/01/2013 07:59 AM - Eric Faulhaber

Option 2.1 seems most correct to me, because it makes the tree accurately reflect the meaning of the code from the beginning.

Greg, Constantin, thoughts?

#48 - 03/01/2013 08:16 AM - Constantin Asofiei

2.1 would be OK only if the ON WRITE statement would work only with table names, but looks like it works with buffer names too:

```
def buffer b for book.
on write of b do: end. /* this compiles OK */
```

So, why not test for for KW_BUFFER too, where you test for KW_TABLE, in database_general.rules ?

#49 - 03/01/2013 08:31 AM - Greg Shah

Constantin's point is a good one. It looks to me like the parser is fine (of course I like saying that :)).

Can we just handle both tables and buffers since the 4GL syntax allows it?

#50 - 03/01/2013 12:59 PM - Stanislav Lomany

- File svl_upd20130301a.zip added

#51 - 03/01/2013 01:01 PM - Stanislav Lomany

- File db_trig_mixed.p added

#52 - 03/01/2013 03:04 PM - Eric Faulhaber

This update looks good and has passed conversion regression testing. Please check it into bzr and distribute.

#53 - 04/19/2013 09:03 AM - Eric Faulhaber

- Estimated time set to 254.00

#54 - 11/16/2016 11:06 AM - Greg Shah

- Target version changed from Milestone 4 to Conversion Support for Server Features

Files

svl_test20130213a.zip	177 KB	02/14/2013	Stanislav Lomany
svl_upd20130219a.zip	183 KB	02/18/2013	Stanislav Lomany
svl_upd20130219a.zip	184 KB	02/20/2013	Stanislav Lomany
svl_testcase20130219a.zip	1.16 KB	02/22/2013	Stanislav Lomany
om_upd20130227a.zip	13 KB	02/27/2013	Ovidiu Maxiniuc
svl_upd20130301a.zip	46.8 KB	03/01/2013	Stanislav Lomany