Base Language - Bug #2578

incorrect promotion of local defined variables

05/19/2015 10:03 AM - Ovidiu Maxiniuc

Status:	Closed	**Start date:**	
Priority:	High	**Due date:**	
Assignee:	Ovidiu Maxiniuc	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Cleanup and Stablization for Server Features		
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#1 - 05/19/2015 10:29 AM - Ovidiu Maxiniuc

- Subject changed from incorect promotion of local defined variables to incorrect promotion of local defined variables

- Priority changed from Normal to High

When running some testcases for a customer I encountered cases when they were passing at a first run, then re-running them after some other tests they would fail. And the other way around.

After investigations I found out that some local defined variables (in a function) get promoted to external procedure. This is not correct because the variable will keep the state from a previous execution. Usually nothing happens because basically, all used variables are first initialized. However, with the aggressive use of NO-ERROR clause in the customer application server, the respective variable won't be (re-)initialized if the expression caused an error condition so it will keep the value form previous execution.

Consider the following P4GL code:

```
DEFINE TEMP-TABLE tt9
   FIELD ff9int AS INT
   FIELD ff9char AS CHAR.

FUNCTION assert-index RETURNS LOGICAL (INPUT extent-var-name AS CHARACTER, INPUT expected AS INTEGER) :
   DEFINE VARIABLE var-name AS CHARACTER NO-UNDO.
   DEFINE VARIABLE var-extent AS INTEGER NO-UNDO.

   var-name = ENTRY(1, extent-var-name ,"[").
   var-extent = INTEGER(TRIM(ENTRY(2, extent-var-name, "["), "]")) NO-ERROR.

   FIND FIRST tt9 WHERE ff9int = var-extent NO-ERROR.

   RETURN expected = var-extent.
END FUNCTION.

MESSAGE assert-index("xdsc",   0)
        assert-index("dsc[1]", 1)
        assert-index("dsc[6]", 6)
        assert-index("xdsc",   0)
        assert-index("dsc[6]", 6).
```

It should display a nice

```
yes yes yes yes yes
```

because the second parameter is always the expected index of the extent variable.

P2J fails on the 4th call and displays yes yes yes no yes. Looking in the generated code, we can see that the integer varExtent = new integer(0); is

promoted to main class variable (external procedure level). At runtime, it will retain the value from 3rd run (6) to the 4th run because the assignment to var-extent will not be executed (the error condition inside the NO-ERROR bracket). In consequence the function will return Nope because the value differs from the expected one.

The reason why the variable is promoted is related to the fact that it takes part into the FIND FIRST query. Looking at the other variable, var-name, it stays local because it it is not used in such constructs.

#2 - 05/19/2015 11:30 AM - Eric Faulhaber

Please check this 4GL code into the testcases project and post the name here.

#3 - 05/19/2015 12:02 PM - Ovidiu Maxiniuc

Please check the uast/scoping/local-variable-in-query.p.

#4 - 05/19/2015 12:22 PM - Eric Faulhaber

- Project changed from Database to Base Language

#5 - 05/19/2015 12:27 PM - Eric Faulhaber

- Assignee set to Ovidiu Maxiniuc

Ovidiu Maxiniuc wrote:

> The reason why the variable is promoted is related to the fact that it takes part into the FIND FIRST query. Looking at the other variable, var-name, it stays local because it it is not used in such constructs.

I'm assigning this to you; it appears you already are well on the way to tracking it down. I agree with your assessment that this is high priority.

#6 - 05/19/2015 03:23 PM - Ovidiu Maxiniuc

- Assignee deleted (Ovidiu Maxiniuc)

Eric,
As I expected, the variables are generally promoted if they are found to be referenced in a query substitution expression. This is documented in TRPL rules/annotations/scope_promotion.rules, line 312. From my test case, this is not always the correct. I am trying to understand when this is required, but I have no luck. The only case that I can think of is the variable is defined in a inner/anonymous class?

#7 - 05/19/2015 04:32 PM - Eric Faulhaber

- Assignee set to Ovidiu Maxiniuc

I think here we are over-generalizing for a specific case handled at line 619 of where_clause_post.rules, where we used to emit a query substitution parameter defined as an anonymous inner class assigned to an instance variable of the enclosing Java class. This is for a case where we have to evaluate a query substitution parameter later than query construction and thus we cannot emit the parameter inline in the query constructor or addComponent method as we normally would.

However, with my recent change to buffer scoping, I moved these query substitution expressions to be scoped to the nearest enclosing procedure (internal or external), function, or trigger. So, I think this promotion is no longer necessary. Unless the variable is referenced beyond the scope of the

nearest enclosing procedure, function, or trigger, I think it will be safe to drop this promotion rule.

So, I think you can drop the following from that rule:

```
...
ancestor(prog.client_where, -1)       or
ancestor(prog.client_where_subst, -1) or
ancestor(prog.query_subst, -1)        or
...
```

This will produce A LOT of changes in converted code, so you will have to review the output carefully.

#8 - 05/20/2015 09:24 AM - Ovidiu Maxiniuc

After some local tests with proposed fix I did a first conversion on Majic. There are indeed a lot of changes, 10% the files from this project are affected. It is impossible to review them all, piece by piece. However, I inspected some of them and here are my findings:

- most of affected files moved local variables from class scope to anonymous Block of execute() method scope.
- there are some files that were defining a duplicate for same variable, using the init() to set the 'external': ExpAddco.this.ipinCustNum = ipinCustNum;. These were completely removed. This is a good thing, as they were not used in the method at all.
- if two or more functions had the same name of parameters and they were promoted, the RefN suffix was appended to make them different. Since they are not promoted any more, they can keep their original name (of course, in java dialect). The second good thing.
- there is one regression in conversion: the AircraftProgram failed to compile because curAircraft is used in the EmbeddedAssignmentExpr0 class that is defined at the 'top' level in the public class, where the variable is not visible. After moving manually the inner class to anonymous Block of execute() method scope (there is an inner class Validation0 at same level), the Majic compiled without any warnings.

I think that is the correct location/scope of the inner EmbeddedAssignmentExpr0 class (like the inner Validation classes). I'm checking now why it was 'promoted' like the normal variables.

#9 - 05/20/2015 12:29 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> I think that is the correct location/scope of the inner EmbeddedAssignmentExpr0 class (like the inner Validation classes). I'm checking now why it was 'promoted' like the normal variables.

In my recent changes, where I moved record buffers from the top level to the more specific scopes, I tried to minimize the change by only moving those constructs which referenced the buffers (validation expressions, client where expressions, query substitution expressions). I did not make any changes to embedded assignment expressions.

#10 - 05/20/2015 12:49 PM - Ovidiu Maxiniuc

That's exactly what I should do in this regard, too: to move the embedded assignment expressions to a more specific scope. I looked into trpl code and I saw that the embedded_assignment_inner_class_def seems to be attached to a getSectionAnchor(java.cs_inner_classes). However, it is not the most specific section.
And while checking the expressions.rule I saw that some other inner classes might be linked to same anchor, including: HandleExpr, PutExpr, ExportExpr. I need to test these too, to be sure.

#11 - 05/20/2015 12:52 PM - Eric Faulhaber

See get_top_level_section_anchor in common-progress.rules. I use this as a substitute for getSectionAnchor.

#12 - 05/20/2015 01:25 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> See get_top_level_section_anchor in common-progress.rules. I use this as a substitute for getSectionAnchor.

Thanks, it looks promising. It has fixed the location for my testcases.

#13 - 05/21/2015 11:32 AM - Ovidiu Maxiniuc

- File om_upd20150521a.zip added

I have tested this update with majic conversion and it looks fine to me.
There are a lot of changes in the code: aside from variables, the HandleExpr, PutExpr, ExportExpr and the EmbeddedAssignmentExpr definitions are also pushed to more specific scopes of the inner-classes. I am now running the runtime tests to have a final confirmation that there are no regressions.

#14 - 05/21/2015 01:02 PM - Eric Faulhaber

Code review 0521a:

Looks good to me in terms of the persistence-related changes, though I did notice a typo in the header for expressions.rules: "Attaced" instead of "Attached".

Greg, please also review/comment, particularly in light of the effect on HandleExpr, PutExpr, ExportExpr, and EmbeddedAssignmentExpr.

#15 - 05/21/2015 11:48 PM - Eric Faulhaber

I tested om_upd20150521a.zip with the current project. After conversion, there were a number of compile errors (actually, seems to be the same one repeated many times). The problem is that the HandleExprN inner class definitions are now scoped to the external procedure, but there are still instance variables of the top level class which reference these inner classes and try to construct instances of them.

There were 100 of these errors in total, which means there may be other errors (and categories of error) hiding behind these.

The build log and a sampling of the failing files are posted at #1868, note 56.

#16 - 05/22/2015 11:28 AM - Ovidiu Maxiniuc

- File om_upd20150522a.zip added

I moved the HandleExprN instance variable definitions to same scope as their class. The Majic conversion testing passed (there was no difference in the generated code). I already started the customer server project conversion.

#17 - 05/22/2015 02:01 PM - Ovidiu Maxiniuc

Please ignore my om_upd20150522a.zip. It brings nothing new, I zipped the wrong files. Sorry.

#18 - 05/26/2015 01:10 PM - Greg Shah

Do you have a new version that I should use for the code review?

#19 - 05/27/2015 10:55 AM - Ovidiu Maxiniuc

My candidate has the following flaw I am trying to fix: when a FORM HEADER is defined locally in an internal procedure (see the pBeginOutput from cgrnimp0.p) and the frame (FrRecords) is DEFINED in the main/external procedure, the whole FORM HEADER is 'promoted'.
With the current handling:

- the dynamic-expr evaluations classes remain attached to the old location, I.e. the pBeginOutput procedure block, so they are not visible to the scope they are used. I dislike the idea of always sent them to global scope based on a particular annotation like (form-header), but link the HeaderExpr with the registerHeader construct so they share the same scope.
- is the FORM HEADER contains a local variable (to pBeginOutput), the reference to it will 'leak' to external procedure, where it is not visible, in fact. I think we should 'promote' it to solve the visibility issue. I don't know if such case exits in the customer server code, I just found this while building my testcases.

#20 - 05/27/2015 03:27 PM - Ovidiu Maxiniuc

- File om_upd20150527b.zip added

This update is not the ideal solution I wanted but it fixes both last issues.
It passed my testcases, as well as the conversion of cgrnimp0.p from the customer project.

#21 - 05/28/2015 01:08 PM - Eric Faulhaber

Code review 0527b:

The changes look correct to me, but I am not expert in this area of the code. Greg will be reviewing this today. What makes this "not the ideal solution" you wanted?

The good news is that I reconverted the customer server project with P2J rev. 10870 plus this update and tested the result, and this turns out to be the mother of all high-leverage fixes. **It fixed over 4,000 unit tests!** Nice one!

Please get this regression tested (conversion and runtime) and checked in as soon as possible, unless something needs attention after Greg's review.

#22 - 05/28/2015 02:08 PM - Ovidiu Maxiniuc

- File om_upd20150528a.zip added

Eric Faulhaber wrote:

> The changes look correct to me, but I am not expert in this area of the code. Greg will be reviewing this today. What makes this "not the ideal solution" you wanted?

The 0527b will promote more variables than needed. This is what we tried to overcome with this task. I continued working; the attached update will leave those variable at their scope but still fails to promote in some particular cases. This is because the FORM HEADER is attached to the point where the FRAME is DEFINED at a later moment than the variables are processed. This is a known flaw in this update and I am still not aware if there are such constructs in the customer server project.
Please see the uast/scoping/form-header-scopes.p that I uploaded on testcase repository for three cases where FORM HEADER is promoted or not.

> The good news is that I reconverted the customer server project with P2J rev. 10870 plus this update and tested the result, and this turns out to be the mother of all high-leverage fixes. **It fixed over 4,000 unit tests!** Nice one!

Glad to hear such news. Then only a few hundreds remain. With #2552 I hope they will drop rapidly again.

> Please get this regression tested (conversion and runtime) and checked in as soon as possible, unless something needs attention after Greg's review.

The testing on devsrv01 is in progress. I understand you already have the customer application conversion and runtime results for 0527b. I put the new update to test but the result will only be available tomorrow.

#23 - 05/28/2015 03:45 PM - Greg Shah

Code Review om_upd20150528a.zip

1. Are the form header changes are necessary to fix this bug? If I understand correctly, the actual query-related promotion changes are as simple as just removing this code from scope_promotion.rules:

```
               ancestor(prog.client_where, -1)       or
               ancestor(prog.client_where_subst, -1) or
               ancestor(prog.query_subst, -1)        or
```

There seem to be no other non-query changes in that file. All the other changes are only to formatting or comments.

2. This code in ui_statements.rules is very fragile:

```
         <!-- in the case that the HEADER was declared in a procedure, we need to check if the
              bogus peer the node was 'promoted' to external procedure -->
         <rule>copy.getAncestor(-1, prog.procedure) != null
            <!-- TODO: check if same constructs can appear in functions & triggers -->
            <action>ref2 = osref.getAncestor(-1, java.static_method_call)</action>
            <while>ref2 != null
               <rule>ref2.getText().equals("externalProcedure")
                  <action>copy.putAnnotation("promoteHeaderItems", true)</action>
                  <action>ref2 = null</action>

                  <action on="false">
                     ref2 = ref2.parent.getAncestor(-1, java.static_method_call)
                  </action>
               </rule>
            </while>
         </rule>
```

This code might break anytime we modify the JAST structure, token types and so forth. Why can't we determine the exact place in the conversion rules where we decide to change the attachment point? The place a rule there to leave behind an annotation. This detection would operate on the Progress ASTs and thus it would be much less fragile.

3. It seems that this code in ui_statements.rules is just dead code?

```
      <rule>evalLib("variables") and type != prog.sys_handle and isNote("refid")

         <action>ref2 = copy.getAncestor(-1, prog.statement)</action>
         <rule>ref2 != null and ref2.isAnnotation("promoteHeaderItems")
            <!-- look for the declaration of the variable -->
            <action>ref2 = getAst(getNoteLong("refid"))</action>

            <rule>!(ref2.isAnnotation("promote"))
               <!-- promote the variable definition associated with this variable reference -->
               <!-- TODO: unfortunately this si too late, variables were already defined-->
               <action>ref2.putAnnotation("promote", true)</action>
            </rule>
         </rule>
      </rule>
```

#24 - 05/29/2015 05:00 AM - Ovidiu Maxiniuc

Greg Shah wrote:

> Code Review om_upd20150528a.zip
> 1. Are the form header changes are necessary to fix this bug? If I understand correctly, the actual query-related promotion changes are as simple as just removing this code from scope_promotion.rules:
> [...]
> There seem to be no other non-query changes in that file. All the other changes are only to formatting or comments.

The FORM HEADER will generate HeaderExpr inner classes that were places in the public class section. Since the local variables that they returned are now local (only in the scope they are used), the HeaderExpr won't be able to access them.

> 2. This code in ui_statements.rules is very fragile:
> [...]
> This code might break anytime we modify the JAST structure, token types and so forth. Why can't we determine the exact place in the conversion rules where we decide to change the attachment point? The place a rule there to leave behind an annotation. This detection would operate on the Progress ASTs and thus it would be much less fragile.

That's the place where the we decide to create the attachment point. Please see above line where the java.bogus peer AST is created. What I am trying to do the the while loop is to verify if the attachment was done under the external procedure for a FRAME HEADER declared in an internal one in P4GL code. In this case the promoteHeaderItems annotation is left behind to let the items from header that they need to be promoted.
The when attaching the inner classes definitions the promoteHeaderItems annotation is checked in expressions.rules and the HeaderExpr are attached to correct scope.

> 3. It seems that this code in ui_statements.rules is just dead code?
> [...]

This is true. As you probably have seen when converting the testcase I mentioned in note 22, the variable is not promoted because this is processed too late. I considered it too risky to change the order of variable_definition after the ui_statements.
I decided to keep it at least as documentation. The good part is that there are no such cases in either of the two projects.

#25 - 06/04/2015 03:16 PM - Greg Shah

OK, I've dug into this code more deeply. I think I understand better what you are trying to do.

I still believe that the changes in ui_statements.rules are not correct.

In the while loop we are trying to determine if the header inner class needs promotion because the registration is being done in the external procedure. I believe you are wrong about the fact of where we determine that the registration is to be moved. By the time the bogus node is created, the "decision" is already known. We can make this code less fragile and more clear by adding processing at annotations time. The location of the registration is determined by the existence and location of the FRAME_ALLOC node. That is the frame-specific node that causes the open scope. This code in ui_statements.rules "finds" that node:

```
         <action>osref = copy.getImmediateChild(prog.kw_form, null)</action>
         <action>osref = getAst(#(java.lang.Long) osref.getAnnotation("frame-id"))</action>
```

After this osref is either null or it is the frame_alloc node. This node is artificial and is created in annotations/frame_scoping.rules. Anytime after that, we can determine the decision that will be made.

If that node exists under the external procedure BUT the current form header node is inside an internal procedure, then we put the promoteHeaderItems annotation into the form_header node. I think this code would do it:

```
<rule>
   type == prog.statement            and
   downPath("KW_FORM/KW_HEADER")     and
   descendant(prog.content_array, 1) and
   copy.getAncestor(-1, prog.procedure) != null)

   <action>osref = copy.getImmediateChild(prog.kw_form, null)</action>
   <action>osref = getAst(#(java.lang.Long) osref.getAnnotation("frame-id"))</action>

   <rule>osref != null and osref.getAncestor(-1, prog.procedure) == null
      <action>copy.putAnnotation("promoteHeaderItems", true)</action>
   </rule>

</rule>
```

This won't be fragile. It is (in my opinion), easier to understand.

If you then do this in a new annotations rule set that executes AFTER frame_scoping.rules but BEFORE scope_promotion.rules, then you can change scope_promotion.rules to check for the existence of promoteHeaderItems and cause proper variable promotion in the normal way. This would eliminate the need for the second code block in ui_statements.rules (the code that is "dead" today). Again, I think this is a better approach.

What do you think? Did I miss something or misunderstand something? If not, then ui_statements.rules would not be changed at all and the changes would be placed back in annotations.

#26 - 06/05/2015 12:21 PM - Ovidiu Maxiniuc

- File om_upd20150605a.zip added

- Status changed from New to WIP

Thanks, Greg, this is exactly my idea. I was sure the ui_statements is not the right place - it is also called too late and I dropped it from the update. Since scope_promotion.rules handles promotion annotations I put evaluated the promoteHeaderItems at the beginning of the walk-rules. When evaluating the promote flag for variables, I add (simple) vars with this annotation to 'header' set that will be forcefully promoted. The expression.rules is unchanged from previous update. It works fine, the annotation is already in place.
With this update, the form-header-scopes.p testcase converts correctly now.
I put the new update to Majic test.

#27 - 06/05/2015 01:10 PM - Greg Shah

Code Review om_upd20150605a.zip

I'm good with this approach. You can check it in if it passes testing.

#28 - 06/06/2015 12:21 PM - Eric Faulhaber

Update 0605a did well with the full group of search unit tests.

#29 - 06/08/2015 10:48 AM - Ovidiu Maxiniuc

The update passed the regression testing, was committed to bzr as revno 10874 and it was distributed by mail.

#30 - 06/08/2015 10:53 AM - Eric Faulhaber

- % Done changed from 0 to 100

- Status changed from WIP to Closed

Nice work, Ovidiu.

#31 - 11/16/2016 12:06 PM - Greg Shah

- Target version changed from Milestone 11 to Cleanup and Stablization for Server Features

Files

om_upd20150521a.zip	18.9 KB	05/21/2015	Ovidiu Maxiniuc
om_upd20150522a.zip	18.9 KB	05/22/2015	Ovidiu Maxiniuc
om_upd20150527b.zip	18.9 KB	05/27/2015	Ovidiu Maxiniuc
om_upd20150528a.zip	41.9 KB	05/28/2015	Ovidiu Maxiniuc
om_upd20150605a.zip	19.3 KB	06/05/2015	Ovidiu Maxiniuc