Database - Bug #6458

ensure that OO expressions resolve to query substitution parameters, like handle-based method calls do

05/27/2022 12:20 PM - Eric Faulhaber

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

Description

History

#1 - 05/27/2022 12:24 PM - Eric Faulhaber

We need to create test cases with OO expressions in the WHERE clause of a query/FIND/FOR/etc., which does not reference the current buffer. These should convert such that the OO expression is refactored into a query substitution parameter.

Constantin, I understand you discovered a problem with this. Do you already have a test case to start with?

#2 - 04/05/2023 03:24 AM - Eric Faulhaber

- Assignee set to Boris Schegolev

Constantin, Greg informed me that you were concerned about certain OO expressions embedded in WHERE clauses not converting properly to query substitution parameters.

Can you please help Boris scope/understand this task?

#3 - 04/05/2023 05:47 AM - Constantin Asofiei

I think the issue is that we don't emit as lambda the OO expression. Take this test:

```
for each tt1 where tt1.f1 = func1(1):
  message "d" tt1.f1.
end.
```

which converts in FWD like:

```
         forEach("loopLabel3", new Block((Init) () ->
         {
            query3.initialize(tt1, "tt1.f1 = ?", null, "tt1.recid asc", new Object[]
            {
               (P2JQuery.Parameter) () -> func1(new integer(1))
            });
         },
```

Note how a lambda is emitted.

Now, for OO case:

```
for each tt1 where tt1.f1 = oo.Foo:m2(1):
  message "a" tt1.f1.
end.
```

it converts like:

```
         forEach("loopLabel0", new Block((Init) () ->
         {
            query0.initialize(tt1, "tt1.f1 = ?", null, "tt1.recid asc", new Object[]
            {
               com.goldencode.testcases.oo.Foo.m2(new integer(1))
            });
         },
```

Note how we don't wrap this in a lambda parameter, and the evaluation is done only once.

From this, other tests where instance methods or both operands are OO calls can be written.

#4 - 05/05/2023 05:37 AM - Boris Schegolev

Gentlemen, I couldn't figure out details of the conversion procedure. So, anyone who has time and knowledge is more than welcome to enlighten me :)

What I know/did so far:

I created test cases to simulate the difference as described above. I understand the conversion goes through stages (preprocessor, lexer, parser, ASTs) and results of individual stages are stored next to the original *.p file as *.lexer, *.parser, *.jast, etc. The stages are defined in ConversionDriver and more steps in TransformDriver.

Then there are rule sets in /rules/, but I am not sure how they participate in the conversion. Are they only used for code validation/analysis (incorrect code triggers rules) or they actually generate the resulting code at some point?

Anyway, I can see that *.jast file already contains the transformed code including the difference described in this task. What I don't understand is what part of our code decided this difference should be there. Any guidance greatly appreciated!

#5 - 05/05/2023 05:41 AM - Constantin Asofiei

Boris, first, run conversion with -Drules.tracing=true. Use this for both the OO and non-OO tests.

You can look at the .ast and .jast file for the non-OO test, and there will be annotations which have the byRule text in them - this will specify which TRPL rule (from rules/) was responsible for emitting that AST node. So, looking at the lambda AST in the .jast file, you can find the .rules file which generated this. You can analyze and backtrack from there.

Otherwise, the TRPL code in rules/ folder is responsible for interpreting the parsed .ast, annotating it, and finally generating the .jast and the .java code.

#6 - 05/11/2023 05:27 PM - Boris Schegolev

- Priority changed from Normal to Urgent

#7 - 05/11/2023 05:27 PM - Boris Schegolev

- Status changed from New to WIP

#8 - 05/15/2023 04:19 PM - Eric Faulhaber

- Priority changed from Urgent to Normal

#9 - 05/22/2023 03:08 PM - Boris Schegolev

- File conversion.log added

I am running a simple conversion for OO mode. The class itself is simple:

```
using oo.*.

class oo.Foo:
   method public static int func1():
      return 1.
   end.
end class.
```

I end up with an error though. There are some nulls reported and then a rule is triggered:

```
Null annotation (full-java-class) for Foo [CLASS_NAME] @3:43 (309237645338)
Null annotation (simple-java-class) for Foo [CLASS_NAME] @3:43 (309237645338)
Null annotation (containing-package) for Foo [CLASS_NAME] @3:43 (309237645338)
Null annotation (found-in-full-java-class) for func1 [OO_METH_INT] @3:47 (309237645339)

com.goldencode.p2j.pattern.TreeWalkException: ERROR!  Active Rule:
------------------------
      RULE REPORT
------------------------
Rule Type :   POST
Source AST:  [ block ] BLOCK/ @0:0 {309237645313}
Copy AST  :  [ block ] BLOCK/ @0:0 {309237645313}
Condition :  persist()
Loop      :  false
--- END RULE REPORT ---
```

This only happens for classes, *.p conversions work just fine. Any suggestions greatly appreciated! Full conversion log attached.

#10 - 05/22/2023 03:10 PM - Constantin Asofiei

You must include Foo.cls in your conversion list.

#11 - 05/22/2023 03:15 PM - Boris Schegolev

Constantin Asofiei wrote:

> You must include Foo.cls in your conversion list.

But it's already reported as being parsed:

```
Lvl01 parse: ./oo/foo.cls
...
Lvl01 DONE:  ./oo/foo.cls
```

Nevertheless, it does the trick :)

Thank you, Constantin!

#12 - 05/22/2023 03:38 PM - Greg Shah

It is parsed because it is referenced elsewhere. As I've noted previously via email, just because it is parsed doesn't mean that it is added automatically to the conversion list. At this time you MUST explicitly include every non-skeleton class/interface/enum in the referenced object graphs of all converted classes or procedures. See #6082 for the task where we fix this. Until then, add everything referenced.

#13 - 05/24/2023 05:58 PM - Boris Schegolev

- % Done changed from 0 to 100

- Status changed from WIP to Review

Finally resolved and pushed as revision 6458a/14553. Please, review.

Thank you all for help with this one!

#14 - 05/25/2023 02:54 AM - Constantin Asofiei

The change looks good. Please post a sample of the 4GL and generated code where the OO method is involved.

#15 - 05/25/2023 08:38 AM - Greg Shah

I'd like to get this merged to trunk, today if possible.

After posting as requested by Constantin, please rebase from the latest trunk.

#16 - 05/25/2023 10:40 AM - Boris Schegolev

The 4GL code was simple like:

```
using oo.*.

for each test_tab1 where test_tab1.col2 = Foo:func1():
  message "d" test_tab1.col2.
end.
```

It newly translates to:

```
   @LegacySignature(type = Type.MAIN, name = "do2.p")
   public void execute()
   {
      externalProcedure(Do2.this, new Block((Body) () ->
      {
         AdaptiveQuery query0 = new AdaptiveQuery();

         forEach("loopLabel0", new Block((Init) () ->
         {
            RecordBuffer.openScope(testTab1);
            query0.initialize(testTab1, "testTab1.col2 = ?", null, "testTab1.recid asc", new Object[]
            {
               (P2JQuery.Parameter) () -> com.goldencode.testcases.oo.Foo.func1()
            });
         },
         (Body) () ->
         {
            query0.next();
            message(new Object[]
            {
               "d",
               (integer) new FieldReference(testTab1, "col2").getValue()
            });
         }));
      }));
   }
```

Branch is rebased, ready to merge.

#17 - 05/25/2023 10:47 AM - Constantin Asofiei

Please do some tests with an instance function instead of static, parameters (simple vars, more complex like a function call, etc), call via THIS-OBJECT:func1(). I'm pretty sure this should work, but please double-check.

#18 - 05/25/2023 11:14 AM - Boris Schegolev

Seems to work just as well:

```
public void execute()
   {
      externalProcedure(Do3.this, new Block((Init) () ->
      {
         ObjectOps.register(inst);
      },
      (Body) () ->
      {
         inst.assign(ObjectOps.newInstance(com.goldencode.testcases.oo.Foo.class));

         AdaptiveQuery query0 = new AdaptiveQuery();

         forEach("loopLabel0", new Block((Init) () ->
         {
            RecordBuffer.openScope(testTab1);
            query0.initialize(testTab1, "testTab1.col2 = ?", null, "testTab1.recid asc", new Object[]
            {
               (P2JQuery.Parameter) () -> inst.ref().func2(new integer(1))
            });
         },
         (Body) () ->
         {
            query0.next();
            message(new Object[]
            {
               "d",
               (integer) new FieldReference(testTab1, "col2").getValue()
            });
         }));
      }));
   }
```

#19 - 05/25/2023 11:37 AM - Constantin Asofiei

Thanks, looks good to merge to trunk.

#20 - 05/25/2023 11:37 AM - Greg Shah

Go ahead with the merge.

#21 - 05/25/2023 01:20 PM - Boris Schegolev

Merged as revision 14586.

#22 - 05/25/2023 02:03 PM - Greg Shah

- Status changed from Review to Closed

Files

conversion.log	32.1 KB	05/22/2023	Boris Schegolev