Database - Bug #2084

Conversion misplaced parameter for can-find statement

03/07/2013 07:44 AM - Costin Savin

Status:	Closed	**Start date:**	03/07/2013
Priority:	Normal	**Due date:**	
Assignee:	Eric Faulhaber	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Conversion Support for Server Features		
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 03/07/2013 07:46 AM - Costin Savin

Example:

```
def temp-table tt1 field f1 as int.
def input parameter table for tt1.
procedure proc0.
   if not can-find(tt1) then message "a".
end.
```

The CAN-FIND, instead of using the "tt1" buffer name, it goes and uses the "tt1Buf2" name:

```
   public void execute(final TableParameter tt1Buf2)
   {
      externalProcedure(new Block()
      {
         public void body()
         {
            RecordBuffer.openScope(tt1);
            TemporaryBuffer.associate(tt1Buf2, tt1, true, false);
         }
      });
   }
```

```
   public void proc0()
   {
      internalProcedure(new Block()
      {
         public void body()
         {
            if (_not(new FindQuery(tt1Buf2, (String) null, null, "tt1Buf2.id asc", LockType.NONE).hasOne()))
            {
               message("a");
            }
         }
      });
   }
```

#2 - 03/07/2013 02:10 PM - Costin Savin

The problem seems to be happening in the annotation phase in the following way:
The BUFFER_SCOPE with a bufname" value="*can-find-tt1" is created and the javaname annotation is correctly set to tt1.
When the def input parameter table for tt1. is processed in **buffer_name_conflicts.rule** it checks if a buffer is instantiated and proceed to find a new name for that buffer by appending "Buf" and an incrementing number, when the name is found it is marked for rename

```
<!-- all further downstream references in buffer_scope nodes
              must be remapped -->
<action>addDictionaryString("remap", javaname, newname)</action>
```

The problem is this is done in the "can-find" buffer scope which will result in can find having a wrong buffer scope.

Shouldn't we have a separate BUFFER_SCOPE for that instead?

#3 - 03/08/2013 04:01 AM - Costin Savin

- File cs_upd20130308a.zip added

Added proposed update,
In previous comment I didn't understood exactly the process so I assumed there's an additional buffer_scope required (it can be ignored), It was only necessary to not change the javaname in post-rules if we have the case of a can-find buffer_scope.

#4 - 03/08/2013 04:51 AM - Constantin Asofiei

The changes looks good, also solves other compile errors where wrong buf name was used, in cases like:

```
def temp-table tt1 field f1 as int.
procedure proc3.
   def input parameter table for tt1.
   if not can-find(tt1) then message "a".
end.
```

which gets converted to:

```
   public void proc3(final TableParameter tt1Buf2)
   {
      internalProcedure(new Block()
      {
         public void body()
         {
            TemporaryBuffer.associate(tt1Buf2, tt1, true, false);
            if (_not(new FindQuery(tt1Buf2, (String) null, null, "tt1Buf2.id asc", LockType.NONE).hasOne()))
            {
               message("a");
            }
         }
      });
   }
```

instead of:

```
public void proc3(final TableParameter tt1Buf2)
{
   internalProcedure(new Block()
   {
      public void body()
      {
         TemporaryBuffer.associate(tt1Buf2, tt1, true, false);
         if (_not(new FindQuery(tt1, (String) null, null, "tt1.id asc", LockType.NONE).hasOne()))
         {
            message("a");
         }
      }
   });
}
```

I'm putting this through conversion regression testing.

#5 - 03/08/2013 06:53 AM - Constantin Asofiei

Conversion regression testing has failed, as the tests above have passed because the javaname in these cases is the same as the legacy name...

#6 - 03/08/2013 12:30 PM - Eric Faulhaber

- Project changed from Base Language to Database

#7 - 03/08/2013 01:53 PM - Eric Faulhaber

Constantin Asofiei wrote:

> Conversion regression testing has failed, as the tests above have passed because the javaname in these cases is the same as the legacy name...

I don't follow what went wrong exactly. Can you explain this more, please?

I changed the test cases slightly to use a legacy name of "tt-1" instead of "tt1". Conversion generates "tt1" from the legacy "tt-1", and the output still looks like what you posted above. What's wrong?

#8 - 03/08/2013 01:59 PM - Constantin Asofiei

Sorry, I should have been more explicit. The failures are caused by the fact that, when you have a table named prodcode, during conversion this gets the ProductCode DMO and the productCode buffer name. After the update is applied, the javaname of the buffer used in a CAN-FIND(prodcode) statement remains set to prodcode instead of productCode.

#9 - 03/11/2013 01:46 PM - Costin Savin

- File cs_upd20130311a.zip added

Added proposed update which checks if the javaname is marked for remap but not as a generated increment name.

#10 - 03/11/2013 09:54 PM - Eric Faulhaber

Code review 20130311a:

- Do you have a test case that illustrates the problem described by Constantin in notes 5 and 8 above is now fixed with this update?
- Please remove debug code before you submit candidate fixes. There are print statements and an annotation (see next bullet) that appear to be for debugging only.
- By convention (and for performance reasons), we should only use lowercase annotation names. You have added an annotation with an uppercase name (and a strange one at that): putNote("ISNEWSHIT", newname). I suspect this is only for debug purposes, so you can find it easily in the AST (since it is not read anywhere after being added). If it is just debug code, see previous point. In any case, we can't keep this annotation name in the code.
- There is a check you've added that I think could be more efficient (especially since it's in a loop):

```
!ref.getAnnotation("bufname").toString().startsWith("*can-find-") or
(lookupDictionaryString("remap", javaname) != null and
 lookupDictionaryString("remap", javaname).equals(javaname))
```

This hits the dictionary twice (a relatively expensive operation) for the same information. This is equivalent to the following, more efficient expression:

```
!ref.getAnnotation("bufname").toString().startsWith("*can-find-") or
lookupDictionaryString("remap", javaname) == javaname
```

In TRPL, the == operator will perform a null test for the left operand and evaluate to false if it is null.

But anyway, this last point is a nuance. Most important is to have a test case that proves this actually solves the problem we're trying to solve.

#11 - 03/12/2013 04:54 AM - Costin Savin

- File cs_upd20130312a.zip added

I'm sorry about that, it seems I forgot to save after doing the clean-up, and created the archive with some unwanted debug code
For the test case, it seems this happens when we have some UAST hints that will rename a table name:
Added a .hints file in ../data/namespace to rename person to employee:

p2j.schema.hints

```
<?xml version="1.0"?>

<!-- UAST hints -->
<hints>
   <schema>
      <table name="person">
          <phrase match="person" replace="employee" />
      </table>
   </schema>
</hints>
```

the test case is the following

```
find first person.
form person.emp-num validate(can-find(first person where person.emp-num = input person.emp-num),"") with frame
 f1.
update person.emp-num with frame f1.
```

#12 - 03/12/2013 11:28 AM - Costin Savin

It seems there are still some weird cases where it fails. Working on reproducing and fixing it

#13 - 03/12/2013 02:33 PM - Costin Savin

- File cs_upd20130312b.zip added

Added proposed update
It seems the problem is in some situation the remap scope is deleted before we get into the <post-rules> and check the dictionary so I moved the logic to <ascent-rules> before the remap scope is deleted and that seem to solve the cases where it didn't work. This still might need some more testing to be sure.

#14 - 03/13/2013 05:25 AM - Constantin Asofiei

There are changes in two files with this update, I think they are expected, but Costin is checking them now.

#15 - 03/13/2013 08:11 AM - Costin Savin

There seems to be a problem with some buffers being used in different db schemas,
after the change it seems the buffer for the wrong db is used.

#16 - 03/13/2013 10:53 AM - Eric Faulhaber

I haven't looked at this issue in detail nor tried to reproduce the problem yet, but the last update you sent looks wrong. You have added the loop which walks the dependents list to the ascent-rules section, with no qualifying rule. This means this loop is being executed *on the ascent event from every leaf and branch in the tree*. It is only necessary to do this once per file, which is why it was in the post-rules section originally.

BTW, when reporting difficulties, please be precise. Instead of mentioning that "there are still some weird cases where it fails" or "there seems to be a problem with some buffers being used in different db schema", provide an exact description of what those cases are. Describe what you were doing (ideally, provide a test case that illustrates the problem), what isn't working, and what you think the result should be. Otherwise, nobody can help you, and it leaves us wondering what the status of the task is.

I will take a look at the test case in note 11 above. Please provide details for note 15.

#17 - 03/13/2013 11:09 AM - Costin Savin

The testcase from note 11 passed, there was another file from conversion regression tests that failed (with the remap scope being deleted before we check the dictionary), I worked directly with the test files that failed, I'll work on reproducing the problems from the tests fails.

#18 - 03/13/2013 03:13 PM - Costin Savin

Tried to replicate the test which failed on update cs_upd20130312a.zip (~~where~~ before I moved the logic from post-rules to ascent-rules) The file which fails(compile) for test is item-t.p : lines 1015 - 1024 and the problem occurs on **update .. validate(can-find ...) statements**
The equivalent of the code that failed there :

```
def temp-table tt1 like book
  field site-id       like person.site-id
  field first-name    like person.first-name
  field last-name     like person.last-name
  field hours         like person.hours.

   update tt1.first-name
                validate(can-find (first person
                         where person.first-name = tt1.first-name) or
                         can-find (first book
                         where book.publisher = tt1.publisher),
                         "Invalid ONE")
          tt1.last-name
                validate(can-find (first person
                         where person.last-name = input tt1.last-name
                         and person.hours = tt1.hours) or
                         can-find (first book
                         where book.publisher = input tt1.last-name
                         and book.sold-qty = tt1.sold-qty),
                         "Invalid TWO").
```

unfortunately when converting this, it passes, I'm not sure what is the fail cause in the original test -looked for .hints file refferences but didn't find anything and at the moment I can't think of what else to look for.

For the second case that fails (after applying the fix from cs_upd20130312b.zip which moves the logic to ascent-rules) the fail (not compile) is in cp-sv.p file which uses a cp-sv.p.hints file to create an alias for database. The fail consist in having the buffer for the wrong schema.

The equivalent of the part that is failing is:

can-find-wrong-schema-buff.p

```
def buffer buf1                 for bla.person.
define shared var var1     as   character.
define shared var var2     as   character.

 update var1
           validate(can-find(buf1 where buf1.first-name = "S"
                                and buf1.last-name = ""),
                             " Validation Failes")
       var2
           validate(not can-find(p2j_test.person where p2j_test.person.last-name = "bla") and
                    not can-find(p2j_test.person where p2j_test.person.first-name = "bla"),
                             " Validation Failes")
```

with a hints file

can-find-wrong-schema-buff.p.hints

```
<?xml version="1.0"?>

<!-- UAST hints -->
<hints>
  <alias
     name="bla"
     database="p2j_test"  />
</hints>
```

Again this converts correctly but in the test file fails.

As a possible solution I've created a function which takes as parameters the existing javaname for the buffer and the one to be replaced, searching the buffer_scope to which they belong and checking if the schemaname is the same for both, but if the first solution which moves the logic from post-rules to ascent-rules is not ok I'll have to rethink it.

Hope this explains a bit where the problem occurs, maybe tomorrow I'll figure out what can be the fail cause.

#19 - 03/14/2013 03:35 PM - Eric Faulhaber

Costin, please provide a detailed update of what you have found/done on this today.

#20 - 03/14/2013 03:46 PM - Costin Savin

- File cs_upd20130314a.zip added

Went back and searched for another solution, since the last one started from the wrong premise that can-find BUFFER_SCOPE should not be changed.

The solution I adopted was to check the refid Ast from which we take the new javaname and see if it is in a DEFINE PARAMETER TABLE structure (which is where the problem seems to come from) and ignore the javaname change in this case.

Converted the tests that previously failed and they converted fine.
I hope this finally fixes it.

#21 - 03/15/2013 12:30 PM - Constantin Asofiei

Eric: I've put this through conversion regression testing (which has passed) plus server folder conversion, and I don't see any regressions caused by it.

#22 - 03/17/2013 03:31 PM - Eric Faulhaber

- File ecf_upd20130317b.zip added

The net effect of the cs_upd20130314a.zip update is that the correct Java output is produced, but the change still leaves behind an incorrect AST. So, I'm concerned it masks the problem and may not work in all cases.

The root cause of this problem was further back than buffer_name_conflicts.rules. The CAN-FIND's "fake" buffer scope was dependent upon the wrong "real" buffer scope. The defect was in record_scoping_post.rules, during the collection of primary buffers for dependent buffers to reference. This rule-set was properly collecting the primary buffer of the DEFINE TEMP-TABLE statement. However, upon encountering the node for the secondary buffer associated with the DEFINE PARAMETER FOR TABLE statement, it was improperly replacing the primary buffer with this secondary buffer. So, the fake buffer scope of the CAN-FIND attached itself to this secondary buffer instead of the correct, primary buffer. The secondary buffer then had its javaname reassigned by the name conflict resolution logic, and the CAN-FIND picked up this incorrect name.

The attached update fixes the original problem program and the test case. It has passed conversion regression testing, and is committed to bzr rev. 10295.

#23 - 03/17/2013 03:32 PM - Eric Faulhaber

- Assignee set to Eric Faulhaber

- Status changed from WIP to Review

- Target version set to Milestone 4

#24 - 03/23/2013 10:11 PM - Eric Faulhaber

- % Done changed from 0 to 100

- Status changed from Review to Closed

#25 - 11/16/2016 11:06 AM - Greg Shah

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

Files

cs_upd20130308a.zip	2.63 KB	03/08/2013	Costin Savin
cs_upd20130311a.zip	2.73 KB	03/11/2013	Costin Savin
cs_upd20130312a.zip	2.67 KB	03/12/2013	Costin Savin
cs_upd20130312b.zip	2.66 KB	03/12/2013	Costin Savin
cs_upd20130314a.zip	2.77 KB	03/14/2013	Costin Savin
ecf_upd20130317b.zip	3.91 KB	03/17/2013	Eric Faulhaber