Database - Bug #3173

use of DYNAMIC-FUNCTION() in the WHERE clause of a dynamic query silently fails to find a record in the 4GL while P2J acts as one would expect

08/19/2016 12:59 PM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 08/19/2016 01:03 PM - Greg Shah

Please see uast/dynamic-queries/p2488-a.p which is heavily annotated.

The following is the code as of this moment:

```
DEFINE TEMP-TABLE tt
   FIELD af AS INTEGER.
CREATE tt.
tt.af = 11.

FUNCTION f RETURNS INTEGER (INPUT i AS INTEGER):
   RETURN i.
END FUNCTION.

FORM book.book-id WITH FRAME f-static-output DOWN.

/* ------------------ static for each ----------------------*/
FOR EACH book, EACH tt WHERE BUFFER tt:NAME = book.isbn:
   /* this will never get executed unless there is an isbn "number" that is "tt" */
   /* (which should not happen) */
   DISPLAY book-id WITH FRAME f-static-output.
   DOWN WITH FRAME f-static-output.
END.

FOR EACH book, EACH tt WHERE BUFFER tt::af <> book.book-id:
   DISPLAY book-id WITH FRAME f-static-output.
   DOWN WITH FRAME f-static-output.
END.

FOR EACH book, EACH tt WHERE BUFFER tt::af <> f(book.book-id):
   DISPLAY book-id WITH FRAME f-static-output.
   DOWN WITH FRAME f-static-output.
END.

/* static results */

/*
┌────────┐
| Book ID|
|────────|
|00000001|
|00000002|
|00000003|
|00000004|
|00000005|
|00000006|
|00000007|
|00000008|
```

```
|00000009|
|00000010|
|00000011|  <--- THIS IS WIERD, SEE #3171
|00000012|
|00000013|
|00099999|
|00000001|
|00000002|
|00000003|
|00000004|
|00000005|
|00000006|
|00000007|
|00000008|
|00000009|
|00000010|
|00000011|  <--- THIS IS WIERD, SEE #3171
|00000012|
|00000013|
|00099999|
|        |
|        |
|        |
|        |
|        |
|        |
|        |
|        |
└────────┘

Press space bar to continue.
*/

/* can't use DEFINE FRAME here because it won't open a scope and the nest uses */
/* below require the scope to already be open at the external proc level else  */
/* there is a compile error */
FORM book WITH FRAME f-dyn-output DOWN.

/* ------------------ dynamic for each ------------------------*/
DEFINE VARIABLE hQuery AS HANDLE NO-UNDO.
CREATE QUERY hQuery.
hQuery:SET-BUFFERS(BUFFER book:HANDLE, BUFFER tt:HANDLE).
hQuery:QUERY-PREPARE("FOR EACH book, EACH tt WHERE BUFFER tt:NAME = book.isbn").
hQuery:QUERY-OPEN.
REPEAT:
   hQuery:GET-NEXT().
   IF available book THEN
   DO:
      /* this never gets executed because the where clause above is silly */
      DISPLAY book WITH FRAME f-dyn-output.
      DOWN WITH FRAME f-dyn-output.
   END.
   ELSE
      LEAVE.
END.
hQuery:QUERY-CLOSE().
DELETE OBJECT hQuery.

CREATE QUERY hQuery.
hQuery:SET-BUFFERS(BUFFER book:HANDLE, BUFFER tt:HANDLE).

/* We add the INTEGER() call here otherwise we get ** Incompatible data types in expression or assignment. (22
3) */
/* when trying to use the :: operator which is has a polymorphic type */

hQuery:QUERY-PREPARE("FOR EACH book, EACH tt WHERE INTEGER(BUFFER tt::af) <> book.book-id").
hQuery:QUERY-OPEN.
REPEAT:
   hQuery:GET-NEXT().
   IF available book THEN
   DO:
      DISPLAY book WITH FRAME f-dyn-output.
      DOWN WITH FRAME f-dyn-output.
   END.
```

```
   ELSE
      LEAVE.
END.
hQuery:QUERY-CLOSE().
DELETE OBJECT hQuery.

CREATE QUERY hQuery.
hQuery:SET-BUFFERS(BUFFER book:HANDLE, BUFFER tt:HANDLE).

/* Must use DYNAMIC-FUNCTION() to execute user-defined functions, cannot call them directly, else we get this
*/
/* list of errors at runtime: */
/* ** Unable to understand after -- "::af) <> f". (247) */
/* PREPARE syntax is: {FOR | PRESELECT} EACH  OF.. WHERE ... etc". (7324) */

/* In this case, evidently the 4GL allows the use of the polymorphic :: operator since it is being compared to
 */
/* the polymorphic dynamic-function builtin.  We don't get an ** Incompatible data types in expression or assi
gnment. (223) */

/* something happens differently with the execution of DYNAMIC-FUNCTION() such that */
/* this query will never retrieve any records; I would guess that the DYNAMIC-FUNCTION() */
/* is only executed once is otherwise not working right */
hQuery:QUERY-PREPARE("FOR EACH book, EACH tt WHERE BUFFER tt::af <> DYNAMIC-FUNCTION('f', book.book-id)").
hQuery:QUERY-OPEN.
REPEAT:
   hQuery:GET-NEXT().
   IF available book THEN
   DO:
      /* never executes in the 4GL */
      DISPLAY book WITH FRAME f-dyn-output.
      DOWN WITH FRAME f-dyn-output.
   END.
   ELSE
      LEAVE.
END.
hQuery:QUERY-CLOSE().
DELETE OBJECT hQuery.

/* dynamic results (all this output is generated by the 2nd dynamic query, the 1st and 3rd have no output) */

/*
┌──────────────────────────────────────────────────────────────────────────────────────────────────────────────
──────────────────────────────┐
| Book ID Title                                Publisher            ISBN            On Hand       Cost Pub Date
   Author ID  Sold      Price|
|──────── ──────────────────────────────────── ──────────────────── ─────────────── ──────── ────────── ────────
── ────────── ────── ──────────|
|00000001 Progress Programming                 Atlantis             1-111111-11-1     00005      24.95 08/01/19
97   0000000     3      29.95|
|00000002 Java Programming                     Sun                  2-222222-22-2     00035      18.95 01/01/20
00   0000000 1,200      24.95|
|00000003 Ruby Programming                     Gems                 3-333333-33-3     00020      21.95 01/01/20
04   0000000   350      27.00|
|00000004 Perl Programming                     Gems                 4-444444-44-4     01250      15.00 01/01/20
03   0000000   650      19.95|
|00000005 Python Programming                   Animals              5-555555-55-5     00023      18.95 01/01/20
04   0000000   220      21.95|
|00000006 PHP Programming                      Web                  6-666666-66-6     00045      12.95 01/01/20
02   0000000   425      14.95|
|00000007 HTML Programming                     Web                  7-777777-77-7     00020      16.00 01/01/19
98   0000000 3,500      19.95|
|00000008 Javascript Programming               Sun                  8-888888-88-8     00012      14.95 01/01/20
01   0000000 1,800      17.95|
|00000009 Cobol Programming                    Atlantis             9-999999-99-9     03000      32.50 01/01/19
90   0000000    32      35.00|
|00000010 Eclipse Programming                  Sun                  1-123456-78-9     00055      24.95 01/01/20
03   0000000   200      28.95|
|00000011 Visual Basic Programming             Borg                 1-222222-33-4     03500      12.95 01/01/19
98   0000000 2,000      14.95|
|00000012 XML Programming                      Web                  2-333333-44-5     00012      24.95 01/01/20
01   0000000 1,200      29.95|
|00000013 Smalltalk Programming                Sun                  9-777999-7-9      01000      18.95 03/31/19
92   0000000     2      24.95|
|00099999 Bogus Programming                    ECF                  abcdefghij        05000      80.00 02/26/20
```

```
06   0000999     0     100.00│
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
│
                              │
└──────────────────────────────────────────────────────────────────────────────────────────────────
──────────────────────────────┘

Procedure complete. Press space bar to continue.
*/
```

In P2J the last dynamic query generates a full set of results that are added to the frame. In the 4GL, that does not happen. The difference seems to be in how (or how often) the DYNAMIC-FUNCTION is evaluated in this case.

Interestingly, this article describes some of the internal sub-expression evaluation going on (read the sample code carefully). I wonder if in the 4GL, they somehow evaluate this once only.