Database - Bug #2295

In Progress output parameters ignore decimal precision

04/29/2014 11:06 AM - Hynek Cihlar

Status:	WIP	**Start date:**	04/29/2014
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to Base Language - Bug #2133: fix precision for decimal, dynamic-exte...	**Closed**	**01/21/2014**	**01/27/2014**
Related to Base Language - Bug #2592: improve support for implicit type conve...	**Closed**		

History

#1 - 04/29/2014 11:11 AM - Hynek Cihlar

The following code:

```
function foo_o returns int (output j as decimal).
  j = 1.5555555555.
end.

def var p as decimal decimals 2.
foo_o(output p).
message "p = 1.5555555555" p = 1.5555555555.
p = 1.5555555555.
message "p = 1.56" p = 1.56.
```

outputs:

```
p = 1.5555555555 yes
p = 1.56 yes
```

The same behavior can be observed for extents.

#2 - 07/01/2014 05:15 PM - Hynek Cihlar

The case above from note 1 has been fixed as part of the issue #2293, and about to be released.

A similar case exists for table fields, consider the code sample below.

```
def temp-table tt
   field f as decimal decimals 3 extent 2.

create tt.
find first tt.

function foo returns int extent (input-output p as decimal extent).
  p[1] = 1.5555555555.
end.

foo(input-output f).
if f[1] <> 1.5555555555 then message "ERR f[1] <> 1.5555555555".
```

After the call to foo, f is expected to contain the value 1.5555555555, however in P2J it will hold the value 1.556.

#3 - 07/02/2014 03:51 AM - Constantin Asofiei

Hynek Cihlar wrote:

> The case above from note 1 has been fixed as part of the issue #2293, and about to be released.
>
> A similar case exists for table fields, consider the code sample below.

This is another 4GL stupidity which doesn't make sense. In case of table fields, the decimal clause is something tightly coupled with the field's definition at the DB. Please double-check how this behaves with a physical table; use the p2j_test.df schema and add a new extent, decimal field to some existing table (maybe person?). The key here is how does the record end up in the physical DB: are the number of decimals specified in the schema enforced when the record is saved?

#4 - 07/12/2014 06:27 PM - Hynek Cihlar

- Status changed from New to WIP

#5 - 07/13/2014 03:35 PM - Hynek Cihlar

The case above from note 1 has been fixed by #2293.

#6 - 07/17/2014 04:36 PM - Hynek Cihlar

Constantin Asofiei wrote:

> Please double-check how this behaves with a physical table; use the p2j_test.df schema and add a new extent, decimal field to some existing table (maybe person?). The key here is how does the record end up in the physical DB: are the number of decimals specified in the schema enforced when the record is saved?

So I tested the input-output/output parameter assignment in respect to the decimal precision and the results are quite surprising!

In Progress, when the target of the input-output/output parameter assignment is a field backed by a physical database, the precision defined in the schema is ignored. In other words, Progress will happily persist a value with higher precision then specified in the schema.

P2J works as stated above. The value assigned back from an input-output/output parameter to a decimal field backed by a physical database is always rounded to the precision defined in the schema.

#7 - 07/17/2014 04:46 PM - Greg Shah

Please post the testcase(s) that you used to find this strange behavior. I'd like to see how applications might rely upon this unexpected behavior. It seems like the 4GL would cause the rounding to be honored at some point, but perhaps that rounding only occurs on assignment and until the next assignment no rounding occurs.

This will potentially be a problem for us. I think we actually configure the DDL to set a hard limit on the precision for our decimal columns.

#8 - 07/17/2014 05:24 PM - Hynek Cihlar

Greg Shah wrote:

> Please post the testcase(s) that you used to find this strange behavior. I'd like to see how applications might rely upon this unexpected behavior. It seems like the 4GL would cause the rounding to be honored at some point, but perhaps that rounding only occurs on assignment and until the next assignment no rounding occurs.

The test case is actually very simple. These are the steps:

- create a database and table with decimal field with the default number of decimals (2), see below for the actual df file.
- run the following program.

```
create test.
find first test.

function foo returns int extent (output p as decimal).
   p = 1.5555555555.
```

```
end.

foo(output test.dd).
```

- export the database data
  The exported data will contain one record with the dd value equal to 1.5555555555.

  This will potentially be a problem for us. I think we actually configure the DDL to set a hard limit on the precision for our decimal columns.

Indeed, fixing this would require to relax the type definition in the DB schema and to manage the decimal precision at the application layer. Another option would be to announce this as a known P2J limitation.

```
ADD TABLE "test"
  AREA "Schema Area"
  DUMP-NAME "test"

ADD FIELD "dd" OF "test" AS decimal
  FORMAT "->>,>>9.99"
  INITIAL "0"
  POSITION 2
  MAX-WIDTH 17
  DECIMALS 2
  ORDER 10

.
PSC
cpstream=ISO8859-15
.
0000000186
```

#9 - 07/18/2014 04:21 AM - Constantin Asofiei

Hynek, please check the following too:

1. what happens if the record (for which the field's decimals doesn't match the schema decmials clause) is re-read by 4GL into another buffer, right after the default buffer was changed:

```
create test.
find first test.

function foo returns int extent (output p as decimal).
   p = 1.5555555555.
end.

foo(output test.dd).
def buffer b for test.
find first test.
```

Do some other variations, and try to determine if 4GL always forces the decimals clause (when reading a record) or it uses the actual value from the DB.

2. what is the maximum number of decimals the record can end up with? Is it related to MAX-WIDTH?

#10 - 07/18/2014 08:25 AM - Hynek Cihlar

When the data is read into another buffer the buffer will still hold the unconstraint decimal value. I tried your sample above as well as the following.

```
find first test.
message test.dd.
```

The maximum number of decimals the record will end up with seems to relate to the implicit decimals value for parameters. Which is MAX-WIDTH.

#11 - 07/19/2014 05:03 AM - Constantin Asofiei

Hynek Cihlar wrote:

> When the data is read into another buffer the buffer will still hold the unconstraint decimal value. I tried your sample above as well as the following.

Please try to find where the field's DECIMALS clause is forced: assign operator, buffer-copy, buffer-compare, used within WHERE clause, etc. The idea here is this: if the DECIMALS clause is forced only during assignment, then we need to adjust the schema/DMO conversion to set the field's max decimals differently (via MAX-WIDTH or something else).

> The maximum number of decimals the record will end up with seems to relate to the implicit decimals value for parameters. Which is MAX-WIDTH.

Is not clear what you mean here. How are parameters involved? When I referred to MAX-WIDTH, I was referring to the field's schema definition.

#12 - 07/19/2014 07:49 PM - Hynek Cihlar

Constantin Asofiei wrote:

> Please try to find where the field's DECIMALS clause is forced: assign operator, buffer-copy, buffer-compare, used within WHERE clause, etc. The idea here is this: if the DECIMALS clause is forced only during assignment, then we need to adjust the schema/DMO conversion to set the field's max decimals differently (via MAX-WIDTH or something else).

OK, I am on it.

> > The maximum number of decimals the record will end up with seems to relate to the implicit decimals value for parameters. Which is MAX-WIDTH.
>
> Is not clear what you mean here. How are parameters involved? When I referred to MAX-WIDTH, I was referring to the field's schema definition.

By MAX-WIDTH did you mean the default number of decimal places, i.e. 10? The maximum number of decimal places a field can have is either the places declared in the schema or the default number of decimal places (10). The connection to parameters is, (1) parameter assignment causes the field value "overflow" the precision and (2) parameters can't be set the number of decimals, hence they hold the default number of decimal places (10). I hope this answers your question from the note 9.

#13 - 07/20/2014 08:02 AM - Constantin Asofiei

Hynek Cihlar wrote:

> By MAX-WIDTH did you mean the default number of decimal places, i.e. 10? The maximum number of decimal places a field can have is either the places declared in the schema or the default number of decimal places (10). The connection to parameters is, (1) parameter assignment causes the field value "overflow" the precision and (2) parameters can't be set the number of decimals, hence they hold the default number of decimal places (10). I hope this answers your question from the note 9.

OK, I think I understand what you mean now. What you need to test is this: change the field definition's MAX-WIDTH to something small, i.e. 5:

```
ADD FIELD "dd" OF "test" AS decimal
  FORMAT "->>,>>9.99"
  INITIAL "0"
  POSITION 2
  MAX-WIDTH 5
  DECIMALS 2
  ORDER 10
```

and check how the dd field is saved, if it ends up with 10 decimals, i.e. 1.5555555555. Does the DB record hold 10 decimals or something smaller?

#14 - 07/20/2014 03:16 PM - Hynek Cihlar

Constantin Asofiei wrote:

> OK, I think I understand what you mean now. What you need to test is this: change the field definition's MAX-WIDTH to something small, i.e. 5:
> [...]
> and check how the dd field is saved, if it ends up with 10 decimals, i.e. 1.5555555555. Does the DB record hold 10 decimals or something smaller?

It seems MAX-WIDTH doesn't influence the decimal storage. Setting it to a small number or even zero doesn't affect the number of decimal places. I tested this by creating new DB, importing a DF file with updated MAX-WIDTH and running the cases mentioned in the previous notes.

What semantics MAX-WIDTH holds anyway? I didn't find any useful documentation explaining the attribute but I have an impression it is related only to the field's formatting.

#15 - 07/21/2014 03:31 AM - Constantin Asofiei

Hynek Cihlar wrote:

> What semantics MAX-WIDTH holds anyway? I didn't find any useful documentation explaining the attribute but I have an impression it is related only to the field's formatting.

I can't tell either. But if no schema setting can limit the number of decimals of a field, then we need to confirm the assumption that the schema decimals are honoured only by assignments.

#16 - 07/21/2014 09:01 AM - Eric Faulhaber

Hynek, please confirm that your test cases showing the unconstrained storage behavior work the same way on Progress 9.1d (installed on windev01). I would think so, but let's be sure.

#17 - 07/22/2014 03:04 PM - Hynek Cihlar

If other storage systems are planned to be supported I think these should be tested as well. I can imagine (and I actually expect this) the decimal constraint would work differently on MSSQL for example.

#18 - 07/22/2014 06:27 PM - Hynek Cihlar

Constantin Asofiei wrote:

> Please try to find where the field's DECIMALS clause is forced: assign operator, buffer-copy, buffer-compare, used within WHERE clause, etc. The idea here is this: if the DECIMALS clause is forced only during assignment, then we need to adjust the schema/DMO conversion to set the field's max decimals differently (via MAX-WIDTH or something else).

Tested assign operator, buffer-copy, buffer-compare, where clause, validate, display, update, prompt-for with these observations.

- input-output/output parameter assignment doesn't honour the field's number of decimals
- assign operator honours the field's number of decimals
- buffer-compare, buffer-copy, where clause and validate don't modify the overflown decimal value and read the field's value without rounding
- display shows the decimal value according to field's format defined in the schema, rounds if the field's value doesn't fit into the format
- update and prompt-for show and read the decimal value according to field's format defined in the schema, round if the field's value doesn't fit into the format

Also, the decimal field's initial value in the *.df schema can be set to higher number of decimal places than what is defined in the schema. The initial value is then read into the buffers without rounding.

I didn't find differences in behavior between Progress 10.2B and 9.1D

#19 - 07/23/2014 02:56 AM - Constantin Asofiei

Hynek Cihlar wrote:

> Constantin Asofiei wrote:
>
> - display shows the decimal value according to field's format defined in the schema, rounds if the field's value doesn't fit into the format
> - update and prompt-for show and read the decimal value according to field's format defined in the schema, round if the field's value doesn't fit into the format

After the update/prompt-for, does the field get saved with the format's decimals or does it honour the DECIMALS clause? Because a simple test like this:

```
def temp-table tt1 field f1 as dec decimals 2 format ">>,>>>.99999".

create tt1.
update tt1.f1.

find first tt1.
display tt1.f1.
```

shows that the save is done with 2 decimals, not 5.

Are all the findings valid for temp-tables, too?

#20 - 07/23/2014 08:57 AM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > Constantin Asofiei wrote:
> >
> > - display shows the decimal value according to field's format defined in the schema, rounds if the field's value doesn't fit into the format
> > - update and prompt-for show and read the decimal value according to field's format defined in the schema, round if the field's value doesn't fit into the format
>
> After the update/prompt-for, does the field get saved with the format's decimals or does it honour the DECIMALS clause? Because a simple test like this:
> [...]
> shows that the save is done with 2 decimals, not 5.

Yes, you are right and I didn't mention it. update saves to database honouring precision. So even if the format allows you to enter a decimal value with more decimal places, update will round to the schema's number of decimals.

prompt-for reads the value according to the format defined in the schema. If the format won't allow you to enter the actual value stored in the database/buffer, prompt-for will not find it.

> Are all the findings valid for temp-tables, too?

Yes, temp tables exhibit the same behaviour for the use cases described.

#21 - 07/25/2014 11:08 AM - Hynek Cihlar

Hynek Cihlar wrote:

If other storage systems are planned to be supported I think these should be tested as well. I can imagine (and I actually expect this) the decimal constraint would work differently on MSSQL for example.

Should I perform this test?

Should I gather any more evidence, some other use cases, etc.?

#22 - 07/25/2014 11:17 AM - Greg Shah

I don't think anything more is needed right now. Eric will make the call on this soon. We will probably have to fix this by setting the precision for all decimal columns to 10 and then updating the runtime code to "leak" the precision only in the specific cases found.

In addition, this bug is really no longer a base language problem, but it is really a database issue. So I will be moving this to that project and Eric will take over from here.

#23 - 07/25/2014 11:18 AM - Greg Shah

- Project changed from Base Language to Database

#24 - 07/10/2015 02:15 PM - Eric Faulhaber

- Assignee deleted (Hynek Cihlar)

- Target version set to Milestone 11

I don't know of any specific code in the customer application that is affected by this, but I am setting this issue to M11 so we don't lose track again.

#25 - 07/23/2016 10:48 AM - Eric Faulhaber

- Target version deleted (Milestone 11)