Database - Feature #1593

Feature # 1592 (Closed): implement additional database built-in functions

add conversion and runtime support for TO-ROWID() built-in function

10/17/2012 12:01 AM - Eric Faulhaber

Status:	Closed	**Start date:**	10/17/2012
Priority:	Normal	**Due date:**	07/16/2013
Assignee:	Ovidiu Maxiniuc	**% Done:**	100%
Category:		**Estimated time:**	50.00 hours
Target version:	Runtime Support for Server Features		
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Blocks User Interface - Bug #1756: display skip fail in procedure	**New**

History

#1 - 10/17/2012 12:45 AM - Eric Faulhaber

- Assignee set to Ovidiu Maxiniuc

Both TO-ROWID(rowid-string) and the inverse/complementary use of STRING(ROWID(record)) must be supported.

The string format should be a hex string in the format 0xHHHHHHHH, where H is an uppercase, hexadecimal digit. Note that the strings generated by the P2J character.valueOf(rowid) method will not match those generated by Progress. Our internal representation of a ROWID is a signed, 64 bit integer (specifically, a java long). We do not know how the 4GL represents these values internally. Thus, round trips will only work within P2J; we cannot generate or read a Progress-compatible rowid-string.

Conversion support must be added to rules/convert/builtin_functions.rules.

TO-ROWID(rowid-string) should map to a new constructor for the rowid data type wrapper (i.e., new rowid(character)).
STRING(ROWID(record)) should map to a new static function character.valueOf(rowid).

#2 - 10/22/2012 02:17 PM - Eric Faulhaber

- Estimated time set to 8.00

#3 - 10/23/2012 05:29 AM - Ovidiu Maxiniuc

After some investigation on the OpenEdge installation on windev01 I observed that string representation of @ROWID@s:

- always must start with 0x' (lowercase 'x'). 0X is invalid
- hexadecimal digits 'a' to 'f' must be also in lowercase. 0x0A is invalid.
- must have at least 2 hexadecimal digits after 0x. 0x0 to 0xa are invalid.
- must have maximum 6 hexadecimal digits, the largest ROWID is 0xffffff. 0x1000000 is invalid.

Implementation on P2J should respect these, isn't it ?

#4 - 10/23/2012 08:20 AM - Ovidiu Maxiniuc

- Status changed from New to WIP

- File om_upd20121023a.zip added

In addition to previous constraints:

- strings representation must have even length. 0x012, 0xabcde are invalid.
- parsing strings with length > 6 are silently ignoring any extra digits. 0x12345678 is valid but will be parsed as 0x123456. Yet, 0x1234567 is invalid (accoring to previous constraint).

I have attached the update that implements this feature (it also contains the new code from Feature #1591:implement QUOTER built-in function).

#5 - 10/23/2012 10:07 AM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> After some investigation on the OpenEdge installation on windev01 I observed that string representation of @ROWID@s:
>
> - always must start with 0x' (lowercase 'x'). 0X is invalid
> - hexadecimal digits 'a' to 'f' must be also in lowercase. 0x0A is invalid.
> - must have at least 2 hexadecimal digits after 0x. 0x0 to 0xa are invalid.
> - must have maximum 6 hexadecimal digits, the largest ROWID is 0xffffff. 0x1000000 is invalid.
>
> Implementation on P2J should respect these, isn't it ?

The first 3 bullets make clear sense, and our implementation should match these requirements.

The 4th, however, seems odd. How did you determine the 6 digit limit? If it is correct, it suggests some sort of temporary mapping of string values to rowids (similar to what we do with the handle class), rather than a "pure" string representation of the underlying rowid value. But could it be that your test was simply limited to a number of records that did not require more than 3 bytes in order to represent the largest rowid internal value? Can you produce a test with more than 2^24 (very simple/small) records in a temp table to confirm this hypothesis?

#6 - 10/23/2012 10:34 AM - Ovidiu Maxiniuc

Bullet 4 is datailed in my next note (4).
0x1000000 is invalid just because the number of digits is odd. However, if you

```
DISPLAY STRING(TO-ROWID('0x12345678')).
```

you will obtain a 0x123456. Just tested on windev1.

2^24 is about 16 millions. I will try to create a small table with such record population.

#7 - 10/23/2012 01:43 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> Bullet 4 is datailed in my next note (4).
> 0x1000000 is invalid just because the number of digits is odd. However, if you
> [...]
> you will obtain a 0x123456. Just tested on windev1.
>
> 2^24 is about 16 millions. I will try to create a small table with such record population.

The odd vs. even digit constraint makes sense, since each pair of digits represents a byte.

I think the issue here with the perceived 6 digit limit is that the default format string for a character displayed in a frame is "x(8)", which means you will see 8 characters (i.e. "0x" + six hex digits) in your output, regardless of the actual string length. It will truncate any characters after the left-most 8.

Try instead something like:

```
DISPLAY STRING(TO-ROWID("0x12345678")) format "x(50)".
```

I think that should work, but if not, try adding a format phrase to the STRING function as well:

```
DISPLAY STRING(TO-ROWID("0x12345678"), "x(50)") format "x(50)".
```

Try it with wider rowid strings, too (e.g., "0x123456789abcdef123456789abcdef") and see if there is a practical limit. Our practical limit in P2J is that of a 64 bit integer.

#8 - 10/24/2012 03:19 AM - Ovidiu Maxiniuc

You are right, the cutting is caused by default print format, in fact I could not find a limit for the size of this type of data.
So, to keep compatibility, a bigger datatype should be used like BigInteger. I'm pretty sure this does't make sense because of the performance penalty.

At this point, the 16M record count experiment will prove nothing, I guess.

On the other side, I observed that Progress stores the ROWID s internally as string. This is proved by the fact that STRING(TO-ROWID("0x000000") == "0x000000", for any even numbers of 0 s. This means that OpenEdge stores the actual string internally (?) and maybe perform a lazy-conversion only when needed (?).

I will adjust the implementation to support our internal 63 bit integer, as this is a fair record size for a table, but don't know how to handle the (rare but possible) overflow case (throw an exception or silently trim the extra digits ?).

#9 - 10/24/2012 02:52 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> You are right, the cutting is caused by default print format, in fact I could not find a limit for the size of this type of data.
> So, to keep compatibility, a bigger datatype should be used like BigInteger. I'm pretty sure this does't make sense because of the performance penalty.

No, I don't see a reason to use BigInteger, since our internal data type is a long.

> On the other side, I observed that Progress stores the ROWID s internally as string. This is proved by the fact that STRING(TO-ROWID("0x000000") == "0x000000", for any even numbers of 0 s. This means that OpenEdge stores the actual string internally (?) and maybe perform a lazy-conversion only when needed (?).

That is a fair guess, though their exact, internal representation is unknown. In any case, this is where our implementation diverges, since we use a long, and we do not expect to be able to convert an existing, 4GL rowid string to a meaningful P2J rowid at runtime. We will have to be aware of this difference if we come across a dependence on persisted, existing, 4GL rowid strings, but it is not something we can deal with in the implementation of this built-in function. Rather, it is an application issue we would need to correct/change before conversion.

> I will adjust the implementation to support our internal 63 bit integer, as this is a fair record size for a table, but don't know how to handle the (rare but possible) overflow case (throw an exception or silently trim the extra digits ?).

In the case of overflow, let's throw an ErrorConditionException, but not directly; use ErrorManager.recordOrThrowError. While this diverges from the 4GL behavior, in that we won't be able to support conversion to/from very large rowid strings, I think that sort of use is largely academic, and is not particularly pragmatic. I think we need to bias the implementation toward the more likely, production uses of this feature, which should normally be constrained to "real" rowids that actually represent records in the database.

#10 - 10/25/2012 05:10 AM - Ovidiu Maxiniuc

ErrorManager.recordOrThrowError() methods require a number or a NumberedException. I understand that this is in fact an error code of 4GL. What number should I use as long as Progress does not generate any errors/warnings here ? Should we use 0, -1 or other negative number as a special case of P2J code?

#11 - 10/25/2012 12:04 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> Should we use 0, -1 or other negative number as a special case of P2J code?

-1.

#12 - 10/26/2012 11:01 AM - Ovidiu Maxiniuc

- Status changed from WIP to Review

- File om_upd20121026a.zip added

Attached update file with changes from last notes and mail from Eric.

#13 - 10/26/2012 01:44 PM - Eric Faulhaber

Feedback from code review:

- Sorry, I was mistaken with respect to the format I originally specified for the rowid string format (beside the uppercase mistake). Since it must represent a 64 bit number, it must have up to 16 hex digits after the "0x" prefix, not 8. The implementation in rowid.java needs to be changed to reflect this (including in the JavaDoc).
- Please don't put a double-slash comment under the line of code it describes, but rather above or (for very short, targeted comments) to the right.
- In the event of any parsing or format problems in the rowid constructor, we effectively create a rowid instance which represents the unknown value (by setting the internal value member to null). Is this how TO-ROWID reacts to the same errors in Progress?
- Please fix the indent in builtin_functions.rules (should be 3 instead of 6 spaces).

#14 - 10/26/2012 01:49 PM - Eric Faulhaber

Where are the test cases you wrote for the wider rowid strings? I didn't see that aspect of your research reflected in the test case x2rowid.p.

#15 - 10/26/2012 02:06 PM - Eric Faulhaber

- % Done changed from 0 to 80

#16 - 10/30/2012 08:49 AM - Ovidiu Maxiniuc

- % Done changed from 80 to 90

- File om_upd20121030a.zip added

My mistake too, I took your affirmation about hexdigits for granted.
I fixed the other issues and added more testcases including 63-bit and wider rowids into the test file.

#17 - 10/30/2012 12:11 PM - Eric Faulhaber

Feedback from code review (1030a):

- Please confirm that in the event of a failure in converting a rowid string to a rowid, Progress creates a rowid representing unknown value (rather than raising an error). This is what our implementation of rowid.parseRowidString() does in most failure cases (by returning null) and I want to be sure it is the correct behavior.
- The first overflow test in rowid.parseRowidString() is good, but the second one should not be necessary. Overflow of the sign bit is OK, it will just result in a negative long, which will still represent a unique primary key. So, it is not necessary to treat the rowid as a 63-bit long, but rather as a 64-bit long.
- Please fix the references in the JavaDoc to 0xhhhhhhhh to reflect the 16-digit hex format supported (0xhhhhhhhhhhhhhhhh).
- There are still some small format issues (opening curly braces in parseRowidString on wrong line, no space between // and comment in one case). Please fix these.

#18 - 10/30/2012 12:29 PM - Ovidiu Maxiniuc

- yes, OE displays [yes] when running the following code: display to-rowid("dksajdhk") = ?.
- ok, I change the code to use the sign bit, too, but this seems to me a patch rather than natural
- the string representation of rowid are not mandatory 16 digits long, any even length is legal. Maybe this should be clarified.
- I found the missing space after double slash, but the { was harder to spot. It's difficult to get rid of old habits.

#19 - 10/30/2012 12:37 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> - yes, OE displays [yes] when running the following code: display to-rowid("dksajdhk") = ?.

OK, thanks.

> - ok, I change the code to use the sign bit, too, but this seems to me a patch rather than natural

Since we support any signed Long as a rowid, I think we have to do it this way to be consistent.

> - the string representation of rowid are not mandatory 16 digits long, any even length is legal. Maybe this should be clarified.

Good idea.

> - I found the missing space after double slash, but the { was harder to spot. It's difficult to get rid of old habits.

Sorry, I know our choice here is uncommon in the Java world.

#20 - 10/31/2012 09:54 AM - Ovidiu Maxiniuc

- File om_upd20121031a.zip added

Uploaded update patch with modifications we discussed yesterday.

The most important change is that special code is needed to use all 64 long's bits as an unsigned. Both utility methods Long.toString() and Long.parseLong() are adapted to unsigned data. So Long.parseLong() will throw exceptions on strings representing integer values larger than 0x8000000000000000. The Long.toString() will use '-' prefix to convert negative numbers, ex: -1 instead of 0xffffffffffffffff.

The issue with this approach is that rowids cannot be compared any more with normal relational operators (<, >, <=, >=). They will fail when comparing two rowids internal representation if at least one of them is negative (larger than 0x8000000000000000 from Progress' point of view). However, this might not be a problem as those operators are not available at 4GL level.

#21 - 10/31/2012 04:07 PM - Greg Shah

- Target version set to Milestone 7

#22 - 11/05/2012 11:19 PM - Eric Faulhaber

Please update the Conversion Reference (builtin_database_functions_methods_attributes.odt chapter) to reflect the implementation of this built-in function.

#23 - 11/06/2012 07:50 AM - Greg Shah

Don't forget that there is also the "master" table of all supported built-in functions in the "Functions" section of the "Expressions" chapter in that same book. Please update that table too.

#24 - 04/25/2013 10:42 AM - Eric Faulhaber

- Estimated time changed from 8.00 to 50.00

#25 - 04/25/2013 03:49 PM - Eric Faulhaber

- Due date set to 07/09/2013

#26 - 05/01/2013 11:47 PM - Eric Faulhaber

- Due date changed from 07/09/2013 to 07/16/2013

#27 - 12/05/2013 07:07 PM - Eric Faulhaber

- % Done changed from 90 to 100

- Status changed from Review to Closed

#28 - 11/16/2016 11:43 AM - Greg Shah

- Target version changed from Milestone 7 to Runtime Support for Server Features

Files

om_upd20121023a.zip	50.8 KB	10/23/2012	Ovidiu Maxiniuc
om_upd20121026a.zip	50.4 KB	10/26/2012	Ovidiu Maxiniuc
om_upd20121030a.zip	51.4 KB	10/30/2012	Ovidiu Maxiniuc
om_upd20121031a.zip	52.3 KB	10/31/2012	Ovidiu Maxiniuc