

Database - Bug #5978

handle implicit data type conversion in runtime (dynamic) conversion

01/17/2022 02:27 PM - Ovidiu Maxiniuc

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

History

#1 - 01/17/2022 02:56 PM - Ovidiu Maxiniuc

The static and dynamic conversion differ a bit from the point of view of handling compatible data types. We noticed that for some cases, the dynamic conversion will do implicit conversion when operands of an expression are numeric and text, for example. In case of static conversion, these cases generate error 223 (Incompatible data types in expression or assignment.)

Some changes in this regard were added in #3574 and #5905, but the behaviour is not identical. We should allow FWD to be as permissible and raise errors as 4GL does. Also we need to heck whether other data types (like logical/date) are affected in same way.

Here is a short procedure which help identify the sensitive cases:

```
DEFINE TEMP-TABLE ttl FIELD num AS INT /* or DECIMAL for some tests */.
DEFINE VARIABLE hqry AS HANDLE.

CREATE ttl.
num = 33.

CREATE QUERY hqry.

hqry:SET-BUFFERS (BUFFER ttl:HANDLE) .
hqry:QUERY-PREPARE ("for each ttl where " +
                    "num modulo 4 >= '2'"). /* predicate line */
hqry:QUERY-OPEN() .
hqry:GET-NEXT() .

IF NOT hqry:QUERY-OFF-END THEN
    MESSAGE ttl.num.

hqry:QUERY-CLOSE.
```

We can see at predicate line that there is a type mismatch which will cause error 223 to be thrown when compiled statically. However, at runtime/dynamic mode, implicit conversion data type will occur. Below is a table with the results for other predicates:

num value	predicate	expression value	output	conclusion
33	num modulo 40 eq (3 + '3')	33 = 33	prints 33	+ evaluates using concatenation, then the terms are compared with positive result. It is unknown if compare operation is performed numeric or string.
33	num modulo 40 eq (32.7 + '0.3')	33 ≠ 32.70.3	none	+ evaluates using concatenation, then the terms are compared with negative result. It looks like the compare

				operation is performed on string versions of the terms.
33	num modulo 40 eq (33.3 - '0.3')	33 = 33	prints 33	- evaluates using numeric operator after converting the second term to numeric (decimal)
33	num modulo 40 eq '330' / 10	33 = 33	prints 33	/ evaluates using numeric operator after converting the first term to numeric (decimal)
33	num modulo 30 eq chr(asc('0') + 3)	N/A	fails: error 223	fail because of occurrence of function(s) ?
33.3	num + 0 eq (33 + '.3')	33.3 = 33.3	prints 33.3	works for both string and numeric data type
33.3	num + 0 eq (32 + '1.3')	33.3 ≠ 321.3	none	right term evaluated as string concatenation
33.3	num + 288 eq (32 + '1.3')	321.3 = 321.3	prints 33.3	right term evaluated as string concatenation
33.3	num + 0 eq '333' / 10.0"	33.3 = 33.3	prints 33.3	right term evaluated as division after converting the string to numeric