

Base Language - Bug #2288

incorrect conversion and results for unknown value comparisons

04/18/2014 12:30 AM - Eric Faulhaber

Status: Closed	Start date:
Priority: High	Due date:
Assignee: Eric Faulhaber	% Done: 100%
Category:	Estimated time: 0.00 hour
Target version: Cleanup and Stabilization for Server Features	case_num:
billable: No	version:
vendor_id: GCD	
Description	
Related issues:	
Related to Database - Bug #18: issues with ne/equal and true/false conversion...	WIP 10/23/2012

History

#1 - 04/18/2014 12:57 AM - Eric Faulhaber

Consider the following 4GL test (testcases/uast/unknown-comparison2.p):

```
def var u as logical init ?.  
  
message "u = yes" u = yes.  
message "u <> yes" u <> yes.  
message "u = no" u = no.  
message "u <> no" u <> no.  
message "u = ?" u = ?.  
message "u <> ?" u <> ?.
```

Results in 4GL:

```
u = yes no  
u <> yes yes  
u = no no  
u <> no yes  
u = ? yes  
u <> ? no
```

Results in P2J:

```
u = yes ?  
u <> yes ?  
u = no ?  
u <> no ?  
u = ? yes  
u <> ? no
```

The problem is that we are converting these expressions as if the comparison of an expression of unknown value to a logical constant is unknown, when in fact it is known (unknown value is not true and it is not false, so `u = yes` and `u = no` should both evaluate to no, while `u <> yes` and `u <> no` should both evaluate to yes).

However, we convert both `u = yes` and `u <> no` to `u`, and we convert `u = no` and `u <> yes` to `not(u)`. Perhaps instead, we should convert `u = yes` and `u <> no` to a new function, `isTrue(u)` (or `_isTrue(u)`, where a boolean return value is more appropriate), and `u = no` and `u <> yes` to `isFalse(u)` (`_isFalse(u)`)?

See also testcases/uast/unknown-comparison.p for similar problems with range comparisons.

#2 - 04/18/2014 02:10 PM - Greg Shah

Is this the same as [#18](#)?

#3 - 04/18/2014 02:28 PM - Eric Faulhaber

Seems it started off that way, though in the history, [#18](#) only describes the inequality comparisons with true/false, then expands into where clause issues. I guess that issue is still pending. However, I'd like to address the non-where-clause side of the issue (as described above) separately, which seems to be a much more straightforward situation.

#4 - 04/18/2014 10:27 PM - Eric Faulhaber

- File *ecf_upd20140418a.zip* added

The attached update, based on some of Vadim's findings documented in [#18](#) (note 9), resolves the issue.

#5 - 04/21/2014 11:20 AM - Eric Faulhaber

- File *ecf_upd20140420a.zip* added

Regression testing indicated that this change had the useful side effect of fixing some queries that were selecting the wrong index because two of the roll-up cases (i.e., value = false and value <> true) were not correctly excluding where clause expressions.

However, it also showed that this fix was too aggressive, in that reverted shortcuts like

```
if (TransactionManager._isRetry())  
...
```

to

```
if (_isEqual(TransactionManager.isRetry(), true))  
...
```

The attached update fixes this. I am regression testing again.

#6 - 04/21/2014 12:49 PM - Greg Shah

Code Review 0420a

Not that you were asking, but I am fine with the changes.

Are the remaining portions of [#18](#) only related to WHERE clauses?

#7 - 04/21/2014 01:13 PM - Eric Faulhaber

Greg Shah wrote:

Not that you were asking, but I am fine with the changes.

OK, the feedback is helpful, particularly since the changes primarily affect base language and control flow.

Are the remaining portions of [#18](#) only related to WHERE clauses?

Yes, I believe so.

#8 - 04/21/2014 01:31 PM - Greg Shah

It probably makes sense to note the status of this task in [#18](#) and to move that task into the Database project. It may also make sense to update the [#18](#) task name to reflect the WHERE clause nature.

#9 - 04/21/2014 10:06 PM - Eric Faulhaber

- File *ecf_upd20140421a.zip* added

Regression testing again showed an unwanted change with the 0420a update. Previously,

```
if false = false
then
  code path A
else
  code path B
```

was rolled up to:

```
code path A
```

and code path B was dropped as dead code. With the 0420a update, it became:

```
if (_isEqual(false, false))
{
  code path A
}
else
{
  code path B
}
```

The attached update fixes this, as well as many other variations on this theme (see [testcases/uast/unknown-comparison2.p](#)).

#10 - 04/22/2014 11:53 AM - Eric Faulhaber

The 0421a update has passed conversion regression testing. Now performing runtime testing.

#11 - 06/02/2014 12:10 PM - Eric Faulhaber

- % Done changed from 0 to 100
- Status changed from New to Closed
- Assignee set to Eric Faulhaber

Update 0421a passed runtime regression testing and was committed to bzt rev. 10523 (although in the commit message I mistakenly referenced issue 2266 instead of this issue).

#12 - 11/16/2016 12:06 PM - Greg Shah

- Target version changed from Milestone 11 to Cleanup and Stabilization for Server Features

Files

ecf_upd20140418a.zip	3.85 KB	04/19/2014	Eric Faulhaber
ecf_upd20140420a.zip	4.03 KB	04/21/2014	Eric Faulhaber
ecf_upd20140421a.zip	4.04 KB	04/22/2014	Eric Faulhaber