Base Language - Bug #2076

DMO setter fails when parameter is BaseDataType

03/04/2013 12:34 PM - Costin Savin

Status:	Closed	**Start date:**	03/04/2013
Priority:	Normal	**Due date:**	
Assignee:	Costin Savin	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	Conversion Support for Server Features		
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#1 - 03/04/2013 12:38 PM - Costin Savin

- Subject changed from DMO setter fails when parameter is BaseDataType to DMO setter fails when parameter is BaseDataType

Example:

```
define var h as handle.
do transaction:
create book.
assign book.book-title = h:BUFFER-FIELD("dsc"):BUFFER-VALUE(2).
end.
```

The DMO setters do not accept BaseDataType, the r-value needs to be wrapped in a c'tor based on the DMO property's type, when r-value is _POLY.

#2 - 03/05/2013 02:36 PM - Costin Savin

Trying to get the rules to create a new variable with the same field type to apply only for expression type ATTR_POLY or BaseDataType.

The ExpresionConversionWorker.expressionType method doesn't seem to return expression type as ATTR_POLY for this case.

In the javadoc there is mentioned

```
    * At this time there is support for all basic data types (date, character,
    * integer, decimal, logical, recid, rowid, raw and memptr). The special
    * FUNC_POLY, ATTR_POLY, METH_POLY types are only supported for
    * the following:
    *
    * ABSOLUTE based on the operand
    * ACCUM based on the accumulator subtype and sometimes the expression operand
    * BUFFER-VALUE with context analysis
```

since it doesn't return the value ATTR_POLY in this case I'm guessing the "with context analysis" part is not met?

I'll think of a way to solve this

#3 - 03/05/2013 03:34 PM - Greg Shah

ExpressionConversionWorker never returns the int type (e.g. ATTR_POLY) but instead will return the wrapper class name for the sub-expression. But it is true that where we have an *_POLY type whose actual return type cannot be determined, today we return null.

We can probably return "BaseDataType" instead of null for all of these _POLY cases. My only concern is that we have other logic that probably depends on null coming back (see convert/operators.rules). So we would have to carefully fixup all those dependencies. Or you can just put in logic that deals with null by assuming it is BaseDataType.

#4 - 03/06/2013 11:24 AM - Costin Savin

- File cs_upd20130306c.zip added

Added proposed update
NOTE
I first tried to use ExpresionConversionWorker.expressionType with an ast copy without a parent (because it would search up in the parent for the type and yield wrong result) this worked ok, but warnings were generated and the return was null - Instead I created isExprPolyType method which checks if the expression return type is one of the types for which we have to apply the fix.

#5 - 03/06/2013 11:45 AM - Greg Shah

Code Review:

> I first tried to use ExpresionConversionWorker.expressionType with an ast copy without a parent (because it would search up in the parent for the type and yield wrong result)

1. In what cases are we calculating the wrong type based on content analysis? I would rather fix these cases than work around them. As far as I know, the code before your change was supposed to already determine the type of the _POLY node from the lvalue of the ASSIGN node. You are doing that "manually". We need to figure out the problem rather than put workarounds in.

2. Related to 1 above, I don't understand why isExprPolyType() is needed. The code also seems wrong. It iterates through all children of the subtree at the EXPRESSION node, which could be hundreds (nested in arbitrary depth). How could that possibly be correct? In fact, why are we even looking at multiple children of the EXPRESSION node anyway?

3. There is no rule/ directory. It should be rules/.

4. A valid EXPRESSION node can never have a null parent (only the root of the tree can have that). The change on line 1022 should be backed out. Actually, all the ECW changes should be backed out. If there is something broken in the ECW, we will fix it properly (e.g. make the analyzeContext() work as expected.

#6 - 03/06/2013 12:16 PM - Costin Savin

I'll use this example to explain.

```
define var h as handle.
def temp-table test no-undo
    field char-field as char
    field int-field  as integer
    field dec-field  as decimal
    field i64-field  as int64
    field logical-field as logical.

do transaction:
   create test.
   assign test.char-field = h:BUFFER-FIELD("dsc"):BUFFER-VALUE(2.
   assign test.int-field = h:BUFFER-FIELD("dsc"):BUFFER-VALUE(2).
   assign test.dec-field = h:BUFFER-FIELD("dsc"):BUFFER-VALUE(2).
   assign test.i64-field = h:BUFFER-FIELD("dsc"):BUFFER-VALUE(2).
   assign test.logical-field = h:BUFFER-FIELD("dsc"):BUFFER-VALUE(2).
end.
```

and this statement: assign test.int-field = h:BUFFER-FIELD("dsc"):BUFFER-VALUE(2).

When we will try to get the expression type using ExpressionConversionWorker.expressionType it will get to the case for FUNC_POLY, ATTR_POLY, METH_POLY and it will start looking up in the parent context to check the type.

As a result the return value in this case will be **integer** not null, not "ATTR_POLY" or anything that can help tell there is really the case to create a new variable with the type expected by the field.

The solution to not go up in the parent to look for context was to make a copy of the expression AST set the parent as null and use it to get the type without the method looking up in the parent context since that would be null the change on line 1022 was required for that.

If I can't use expressionType for an ast without a parent, and either the logic from isExprPolyType() and expressionType in the current state doesn't return the _POLY type or null for this case what would be the solution to determine that we have a _POLY expression?

#7 - 03/06/2013 12:39 PM - Greg Shah

Please post the AST here for the assign test.int-field = h:BUFFER-FIELD("dsc"):BUFFER-VALUE(2)..

#8 - 03/06/2013 12:44 PM - Costin Savin

The AST

```
<ast col="0" id="300647710812" line="0" text="statement" type="STATEMENT">
  <ast col="4" id="300647710813" line="12" text="assign" type="KW_ASSIGN">
    <annotation datatype="java.lang.Boolean" key="bracket" value="true"/>
    <ast col="26" id="300647710816" line="12" text="=" type="ASSIGN">
      <annotation datatype="java.lang.Long" key="peerid" value="304942678086"/>
      <ast col="11" id="300647710819" line="12" text="test.int-field" type="FIELD_INT">
        <annotation datatype="java.lang.Long" key="oldtype" value="6"/>
        <annotation datatype="java.lang.String" key="schemaname" value="test.int-field"/>
        <annotation datatype="java.lang.String" key="bufname" value="test"/>
        <annotation datatype="java.lang.String" key="dbname" value=""/>
        <annotation datatype="java.lang.Long" key="recordtype" value="14"/>
        <annotation datatype="java.lang.String" key="name" value="test.int-field"/>
        <annotation datatype="java.lang.Long" key="type" value="367"/>
        <annotation datatype="java.lang.String" key="fieldname" value="intField"/>
        <annotation datatype="java.lang.Long" key="bufreftype" value="21"/>
        <annotation datatype="java.lang.String" key="uniquename" value="test_test"/>
        <annotation datatype="java.lang.Long" key="refid" value="300647710922"/>
        <annotation datatype="java.lang.String" key="methodtxt" value="setIntField"/>
      </ast>
      <ast col="0" id="300647710822" line="0" text="expression" type="EXPRESSION">
        <annotation datatype="java.lang.Long" key="peerid" value="304942678088"/>
        <ast col="29" id="300647710823" line="12" text=":" type="COLON">
          <annotation datatype="java.lang.Long" key="peerid" value="304942678095"/>
          <ast col="28" id="300647710824" line="12" text="h" type="VAR_HANDLE">
            <annotation datatype="java.lang.Long" key="oldtype" value="2341"/>
            <annotation datatype="java.lang.Long" key="refid" value="300647710723"/>
            <annotation datatype="java.lang.Long" key="peerid" value="304942678089"/>
          </ast>
          <ast col="49" id="300647710826" line="12" text=":" type="COLON">
            <annotation datatype="java.lang.Long" key="peerid" value="304942678092"/>
            <ast col="30" id="300647710827" line="12" text="BUFFER-FIELD" type="METH_HANDLE">
              <annotation datatype="java.lang.Long" key="oldtype" value="972"/>
              <ast col="43" id="300647710828" line="12" text="&amp;quot;dsc&amp;quot;" type="STRING">
                <annotation datatype="java.lang.Long" key="peerid" value="304942678090"/>
              </ast>
            </ast>
            <ast col="50" id="300647710829" line="12" text="BUFFER-VALUE" type="ATTR_POLY">
              <annotation datatype="java.lang.Long" key="oldtype" value="978"/>
              <ast col="63" id="300647710830" line="12" text="2" type="NUM_LITERAL">
                <annotation datatype="java.lang.Boolean" key="use64bit" value="false"/>
                <annotation datatype="java.lang.Long" key="peerid" value="304942678091"/>
              </ast>
            </ast>
          </ast>
        </ast>
      </ast>
    </ast>
  </ast>
</ast>
```

Note (I must be positioned on the EXPRESSION node to correctly emit the constructors)

#9 - 03/06/2013 01:09 PM - Greg Shah

The best thing to do here is to move the current logic for ECW.expressionType(Aast) to a new public worker method ECW.expressionType(Aast, boolean) and then create a ECW.expressionType(Aast) that calls that method with 2nd parameter set to true.

All current call sites continue using the ECW.expressionType(Aast) method, but you would use the new worker directly (passing false).

When the new boolean flag is false, then analyzeContext() will NOT implement the check on the ASSIGN lvalue. Of course, you will have to modify the analyzeContext() to take a 2nd boolean parameter too (you don't need a separate worker since you can fix up all calls to pass the flag). In that way, you will be able to get a null return if and only if we cannot determine the type solely from the rvalue expression. If we can, then **most likely** we won't have a compile issue. There still may be cases where we have to refine this, but for now I think it will meet your needs.

Let me know if you need further clarification.

#10 - 03/06/2013 01:20 PM - Costin Savin

That crossed my mind too and started in that direction in the morning, but thought about the copy Ast with null parent which didn't require too much modification in the code of expressionType. I hope I can still recover to those changes.
I'll do it like that then.

#11 - 03/07/2013 06:06 AM - Costin Savin

- File cs_upd20130307a.zip added

Added proposed update (uploaded it first by mistake to #2070)

#12 - 03/07/2013 12:20 PM - Greg Shah

Code Feedback:

1. Your zip still has the wrong directory name "rule" instead of "rules". How is that even getting in there? Don't you develop in a real copy of P2J?

2. Your change is too aggressive in disabling analyzeContext(). I don't want analyzeContext() to immediately return null when lookup is true. I want normal context analysis to continue working and only disable the "case ASSIGN:" part (see the bottom of analyzeContext()).

Current code:

```
               case EXPRESSION:
                  Aast grampa = past.getParent();
                  int ttype = grampa.getType();

                  switch (ttype)
                  {
                     case ASSIGN:
                        // if this is an assignment, check the type of the lvalue
                        otype = expressionType(grampa.getChildAt(0));
                        break;
                     case KW_WHEN:
```

Instead of calling expressionType() of the lvalue, it would look like this:

```
               case EXPRESSION:
                  Aast grampa = past.getParent();
                  int ttype = grampa.getType();

                  switch (ttype)
                  {
```

```
                    case ASSIGN:
                       if (lookup)
                       {
                          // if this is an assignment, check the type of the lvalue
                          otype = expressionType(grampa.getChildAt(0));
                       }
                       break;
                    case KW_WHEN:
```

3. Change the description of lookup to explain that context analysis of the lvalue in an ASSIGN is disabled by passing false.

4. Add the lookup parameter javadoc to analyzeContext(). Use the same description as in 3.

#13 - 03/07/2013 01:14 PM - Costin Savin

- File cs_upd20130307b.zip added

Added proposed update.
Sorry I missed the "rule" directory , what I do when adding an update: I manually create the project directory structure for the modified files, then copy the files from my workspace into the folders and archive it at the end.
This probably can be done in a better way.

#14 - 03/07/2013 02:25 PM - Greg Shah

This update is fine. I will conversion regression test it.

> Sorry I missed the "rule" directory , what I do when adding an update: I manually create the project directory structure for the modified files, then copy the files from my workspace into the folders and archive it at the end.

After I am done with a change, I use the following command to get a list of my changes:

```
ant clean && bzr status -v
```

Then I just create a zip command based on that output:

```
zip ges_upd20130307a.zip file1 file2 file3 file4
```

You can cut and paste the command together (instead of the file1...). It is less work and avoids most errors.

#15 - 03/07/2013 04:32 PM - Greg Shah

Conversion testing has finished. There is this one unexpected change:

```
diff -r generated/20130305b/src/aero/timco/majic/ap/Ap02R.java generated/20130307a/src/aero/timco/majic/ap/Ap0
2R.java
1127c1127
<                                       tDetail.setEdiType(EdiVoucherUtils.getVoucherType(xAptrxp));
---
>                                       tDetail.setEdiType(new character(EdiVoucherUtils.getVoucherType(xAptrxp)
));
```

Constantin: what do you think?

#16 - 03/07/2013 04:41 PM - Constantin Asofiei

> Constantin: what do you think?

The result will not affect runtime, is just some small overhead. I think this is because of how the get-voucher-type function gets converted (the node gets retyped from prog.func_char to prog.customer_specific, which later gets converted to a java.static_method_call). We might get rid of this if we add to the prog.customer_specific node the classname="character" annotation.

#17 - 03/07/2013 04:46 PM - Greg Shah

No, that is alright. The difference is not significant at runtime. Thanks.

I will check this in and distribute it.

#18 - 03/07/2013 04:58 PM - Greg Shah

Passes conversion testing and is checked into bzr as revision 10264.

#19 - 03/07/2013 05:00 PM - Greg Shah

- Status changed from WIP to Closed

#20 - 03/09/2013 11:25 AM - Constantin Asofiei

There is a regression related to this update:

```
def temp-table tt1 field f1 as handle.
create tt1.
tt1.f1 = this-procedure.
```

will not pass conversion.

#21 - 03/09/2013 03:57 PM - Greg Shah

A simple change to ECW fixes this. I am making some other changes to the ECW as well. Then I will conversion test this.

#22 - 11/16/2016 11:00 AM - Greg Shah

- Target version changed from Milestone 4 to Conversion Support for Server Features

Files

cs_upd20130306c.zip	18.2 KB	03/06/2013	Costin Savin
cs_upd20130307a.zip	18.2 KB	03/07/2013	Costin Savin
cs_upd20130307b.zip	18.1 KB	03/07/2013	Costin Savin