Base Language - Bug #6482

POLY version of ArrayAssigner.assignMulti, when the r-value can be an extent or scalar

06/01/2022 12:15 PM - Constantin Asofiei

Status:	Test	**Start date:**	
Priority:	High	**Due date:**	
Assignee:	Ovidiu Maxiniuc	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#2 - 06/01/2022 12:17 PM - Constantin Asofiei

- Priority changed from Normal to High

The rvalue of an = operator can be either a scalar on an extent. If the lvalue is dynamic extent, then its reference must be assigned, as in v1 = assignMulti(v1, someOtherExt);.

The following test shows the :: dereference operator can be used for this:

```
def temp-table tt1 field f1 as char extent 5 field f2 as progress.lang.object extent 5.

def var v as class progress.lang.object extent.
def var ch1 as char extent.
def var hbtt1 as handle.

hbtt1 = buffer tt1:handle.
create tt1.
tt1.f1 = "abc".

ch1 = hbtt1::f1.
v = hbtt1::f2.
```

As the conversion side has no idea of the r-value's datatype, we need a kind of POLY version of assignMulti, to cover both scalar and extent cases. Both conversion and runtime need to be fixed.

#3 - 06/01/2022 12:22 PM - Constantin Asofiei

- Status changed from New to WIP

- Assignee set to Constantin Asofiei

#4 - 06/01/2022 03:04 PM - Constantin Asofiei

- % Done changed from 0 to 70

- Assignee changed from Constantin Asofiei to Ovidiu Maxiniuc

Ovidiu, please see 6129a/13903, and add runtime for derefPoly in AbstractTempTable, Dataset and BufferImpl.

#5 - 06/03/2022 02:38 PM - Constantin Asofiei

The test for deferPoly is this:

```
def temp-table tt1 field f1 as char extent 5 field f2 as progress.lang.object extent 5.
def temp-table tt2 field f1 as char field f2 as progress.lang.object.

def var v1 as class progress.lang.object extent 5.
def var v2 as class progress.lang.object extent.
def var ch1 as char extent 5.
def var ch2 as char extent.
def var hbtt1 as handle.
def var hbtt2 as handle.

hbtt1 = buffer tt1:handle.
hbtt2 = buffer tt2:handle.

create tt1.
tt1.f1[1] = "abc".
tt1.f1[2] = "def".

create tt2.
tt2.f1 = "abc".
tt2.f1 = "def".

extent(ch2) = 5.
extent(v2) = 5.

ch1 = hbtt1::f1.
ch2 = hbtt1::f1.
v1 = hbtt1::f2.
v2 = hbtt1::f2.
hbtt1::f1 = ch1.
hbtt1::f1 = ch2.
hbtt1::f2 = v1.
hbtt1::f2 = v2.

ch1 = hbtt2::f1.
ch2 = hbtt2::f1.
v1 = hbtt2::f2.
v2 = hbtt2::f2.
hbtt2::f1 = ch1[1].
hbtt2::f1 = ch2[1].
hbtt2::f2 = v1[1].
hbtt2::f2 = v2[1].
```

#6 - 06/03/2022 08:51 PM - Ovidiu Maxiniuc

- Status changed from WIP to Review

- % Done changed from 70 to 100

I added implementation for AbstractTempTable, Dataset and BufferImpl.

The conversion and runtime works even for constructs like this:

```
hds::tt1::f11 = hds::tt2::f21.
```

The problem is, in Java, the converted code looks 'long' (and difficult to read):

```
hds.unwrapDereferenceable().dereference(handle.class, "tt1").unwrapDereferenceable().dereference("f11", hds.un
wrapDereferenceable().dereference(handle.class, "tt2").unwrapDereferenceable().dereference("f21"));
```

Since you used derefPoly as a shorter name, I wonder if (in ling term) we will make shorter code like this:

```
hds.unwrapDeref().deref(handle.class, "tt1").unwrapDeref().deref("f11", hds.unwrapDeref().deref(handle.class,
"tt2").unwrapDeref().deref("f21"));
```

Or even more compact, with a bit of syntactic sugar,

```
hds.deref("tt1").deref("f11", hds.deref("tt2").deref("f21"));
```

Committed revision 13922.

#7 - 06/04/2022 07:08 AM - Greg Shah

> Or even more compact, with a bit of syntactic sugar,

I like this. The handle type is terrible (in the 4GL). It is OK for us to add things to it to support cleaner unwrapping in Java.

#8 - 06/04/2022 08:34 AM - Constantin Asofiei

Ovidiu, I see two issues in 61291/13922:

- BufferImpl.derefPoly - shouldn't this return an array of the property's type, instead a BaseDataType[]?
- The change in handle$InvalidAttributeAccess.invoke is wrong - possibleChained flag by default is true. Please revert this change.

#9 - 06/04/2022 08:56 AM - Constantin Asofiei

There is another regression: BufferImpl.saveRowChangesImpl2 has no support for extent fields! And your change in BufferImpl.dereference to throw error 14905 (Whole-array assignment target and source must have the same extent unless the target is indeterminate.) regresses SAVE-ROW-CHANGES (previously it was using unknown).

Please fix SAVE-ROW-CHANGES to add extent support.

#10 - 06/06/2022 08:29 AM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> Ovidiu, I see two issues in 61291/13922:
>
> - BufferImpl.derefPoly - shouldn't this return an array of the property's type, instead a BaseDataType[]?

In ArrayAssigner.assignMultiPoly(BaseDataType[] target, Object source), if source.getClass().isArray() it will cast this result as (BaseDataType[]) source.

- The change in handle$InvalidAttributeAccess.invoke is wrong - possibleChained flag by default is true. Please revert this change.

I found a testcase where this is not needed, even more, that line breaks the normal/expected execution flow of the procedure. I remember we have already talked about this some time ago and, although not perfect, that line was the best we could come out at that time. I will investigate further to see a better solution.

#11 - 06/06/2022 05:42 PM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> There is another regression: BufferImpl.saveRowChangesImpl2 has no support for extent fields!

This was not a regression, but a lack in the implementation. Added it now.

> And your change in BufferImpl.dereference to throw error 14905 (Whole-array assignment target and source must have the same extent unless the target is indeterminate.) regresses SAVE-ROW-CHANGES (previously it was using unknown).

Actually the error is correct. I have testcases which show the need for it. However, I added a short comment and we will keep it disabled for now, until we understand why/when it is not needed.

> Please fix SAVE-ROW-CHANGES to add extent support.

Done. Actually the implementation leverages on the new polyDeref() method to work.

Committed revision 13930.

Greg Shah wrote:

> > Or even more compact, with a bit of syntactic sugar,
>
> I like this. The handle type is terrible (in the 4GL). It is OK for us to add things to it to support cleaner unwrapping in Java.

The time is short for this now. It involves changes both in conversion and runtime. I will do this when I will find some spare time.

#12 - 06/07/2022 07:09 AM - Constantin Asofiei

Ovidiu Maxiniuc wrote:

> Actually the error is correct. I have testcases which show the need for it. However, I added a short comment and we will keep it disabled for now, until we understand why/when it is not needed.

I meant the error from BufferImpl.dereference:

```
      if (property.extent != 0)
      {
         ErrorManager.recordOrThrowError(14905);
         // Whole-array assignment target and source must have the same extent unless the target is indetermin
ate. (14905)
         return new unknown();
      }
```

I've re-enabled this test and will commit to 6129a, as SAVE-ROW-CHANGES now works with extent, too.

#13 - 06/07/2022 07:37 AM - Constantin Asofiei

- Status changed from Review to Test