Database - Feature #3816

table and table handle parameter options

11/25/2018 02:38 PM - Eric Faulhaber

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Database - Bug #3364: problems with TABLE-HANDLE and TABLE paramet...	**New**
Related to Base Language - Bug #4424: Additional def var needed to receive ta...	**Closed**
Related to Database - Feature #4516: testing and completion of runtime compat...	**New**

History

#1 - 11/25/2018 02:56 PM - Eric Faulhaber

Options which need support:

- BIND
- APPEND - mostly implemented; partial conversion
- BY-REFERENCE - mostly implemented; partial conversion

#2 - 11/25/2018 02:56 PM - Eric Faulhaber

- Related to Bug #3364: problems with TABLE-HANDLE and TABLE parameters related to BY-REFERENCE added

#3 - 04/22/2019 07:51 AM - Greg Shah

- Assignee set to Stanislav Lomany

3809a contains some changes that implement dataset parameters. This will be merged to trunk shortly. As part of this work, any overall changes to the table parameter approach which also affect datasets, should be discussed so that we can keep them in sync.

#4 - 04/22/2019 07:56 AM - Greg Shah

Please document the specific questions we have that will require 4GL testcases. I don't know if we will be able to get a 4GL developer to help us in time, but it is a possibility. We will need specifics to give that a chance.

#5 - 05/06/2019 01:59 AM - Marian Edu

Greg Shah wrote:

> Please document the specific questions we have that will require 4GL testcases. I don't know if we will be able to get a 4GL developer to help us in time, but it is a possibility. We will need specifics to give that a chance.

Greg, is that something urgent or we can leave it for later? We did some tests cases for datasets using by-reference, bind, append so guess you need something similar for temp-tables?

#6 - 05/06/2019 06:22 AM - Greg Shah

Yes, this one is more urgent since we are about to start work on it.

#7 - 05/16/2019 09:22 AM - Greg Shah

Status update on 4GL testcases development:

> Subject: Feature #3816 - table and table handle parameter options
> Date: 15 May 2019 at 09:29:26
>
> We have implement following tests regarding passing table and table-handle as parameters:

- Table passed – STATC / table defined in procedure - STATIC
  - Parameter mode: INPUT – with and w/o PK defined
    - BY-VALUE
    - BY-REFERENCE
    - BIND
      - BIND – case of table defined in procedure w/o REFERENCE-ONLY
      - BIND - case of table defined in procedure with REFERENCE-ONLY but w/o BIND option
      - BIND - case of table defined in procedure with REFERENCE-ONLY and with BIND option
  - Parameter mode: OUTPUT – with and w/o PK defined
    - BY-VALUE
      - BY-VALUE – APPEND option
    - BY-REFERENCE
      - BY-REFERENCE – APPEND option
    - BIND
      - BIND – case of table defined in procedure w/o REFERENCE-ONLY
      - BIND - case of table defined in procedure with REFERENCE-ONLY but w/o BIND option
      - BIND - case of table defined in procedure with REFERENCE-ONLY and with BIND option
      - BIND – APPEND option
  - Parameter mode: INPUT-OUTPUT – with and w/o PK defined
    - BY-VALUE
      - BY-VALUE – APPEND option
    - BY-REFERENCE
      - BY-REFERENCE – APPEND option
    - BIND
      - BIND – case of table defined in procedure w/o REFERENCE-ONLY
      - BIND - case of table defined in procedure with REFERENCE-ONLY but w/o BIND option
      - BIND - case of table defined in procedure with REFERENCE-ONLY and with BIND option
      - BIND – APPEND option
- Table passed – DYNAMIC / table defined in procedure - DYNAMIC
  - Parameter mode: INPUT – with and w/o PK defined
    - BY-VALUE
    - BY-REFERENCE
    - BIND
  - Parameter mode: OUTPUT – with and w/o PK defined
    - BY-VALUE
      - BY-VALUE – APPEND option
    - BY-REFERENCE
      - BY-REFERENCE – APPEND option
    - BIND
      - BIND – APPEND option
  - Parameter mode: INPUT-OUTPUT – with and w/o PK defined
    - BY-VALUE
      - BY-VALUE – APPEND option
    - BY-REFERENCE
      - BY-REFERENCE – APPEND option
    - BIND
      - BIND – APPEND option
- Table passed – DYNAMIC / table defined in procedure – STATIC
  - Parameter mode: INPUT – w/o PK defined
    - BY-VALUE
    - BY-REFERENCE
    - BIND
  - Parameter mode: OUTPUT – w/o PK defined
    - BY-VALUE
    - BY-REFERENCE
    - BIND
  - Parameter mode: INPUT-OUTPUT – w/o PK defined
    - BY-VALUE
    - BY-REFERENCE
    - BIND
- Table passed – STATIC and with different definition like procedure / table defined in procedure – DYNAMIC
  - Parameter: BY-VALUE – w/o PK defined

        - INPUT
        - INPUT-OUTPUT
        - OUTPUT
    - Parameter: BY-REFEENCE – w/o PK defined
        - INPUT
        - INPUT-OUTPUT
        - OUTPUT
    - Parameter: BIND – w/o PK defined
        - INPUT
        - INPUT-OUTPUT
        - OUTPUT

This is what was tested in procedural code - aka, procedure/function - we're doing the same thing for object oriented right now.

#8 - 05/31/2019 04:36 PM - Greg Shah

See Testcases to access the 4GL code.

#9 - 06/18/2019 03:59 PM - Stanislav Lomany

- Status changed from New to WIP

I have a couple of questions about testcases:

1. Is there a runner file for this set of testcases? I couldn't find one.
2. While trying to run individual testcases I often have been encountering error "EMPTY-TEMP-TABLE is not a queryable attribute for TEMP-TABLE widget". The issue is that EMPTY-TEMP-TABLE method can be applied to buffer and temp-table objects in the latest 4GL version. While I use VM with 4GL 11.6.3 where this method can be applied only buffer objects. It's not a problem to replace tableHdl:empty-temp-table() with tableHdl:DEFAULT-BUFFER-HANDLE:empty-temp-table(). Should I do it and commit to the testcases project? Note that potentially we can have other compatibility issues in the future.

#10 - 06/20/2019 04:07 PM - Stanislav Lomany

Do we support 4GL classes (.cls)?

```
     [java] ./abl/table/dynamic/io/test_input_all_dynamic_in_static_procedure.p
     [java] Non-existent package Progress.Lang.!
     [java] WARNING: failed 1 level parse. Cannot find class/interface Progress.Lang.Object in PROPATH.
     [java] ./abl/table/dynamic/io/test_input_all_dynamic_in_static_procedure_oo.p
     [java]    Recursive level 1 parse of class: ./table/help/oo/InputStatic.cls
     [java] Non-existent package Progress.Lang.!
     [java] WARNING: failed 1 level parse. Cannot find class/interface AppError in PROPATH.
     [java]    Done 1st level parsing for ./table/help/oo/InputStatic.cls
     [java] Failure in file './abl/table/dynamic/io/test_input_all_dynamic_in_static_procedure_oo.p':
     [java] com.goldencode.ast.AstException: Error processing ./table/dynamic/io/test_input_all_dynamic_in_sta
tic_procedure_oo.p
     [java]     at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:972)
     [java]     at com.goldencode.p2j.uast.ScanDriver.lambda$scan$0(ScanDriver.java:375)
     [java]     at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:410)
     [java]     at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:248)
```

```
    [java]     at com.goldencode.p2j.convert.TransformDriver.runScanDriver(TransformDriver.java:349)
    [java]     at com.goldencode.p2j.convert.TransformDriver.front(TransformDriver.java:220)
    [java]     at com.goldencode.p2j.convert.TransformDriver.executeJob(TransformDriver.java:845)
    [java]     at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:983)
    [java] Caused by: java.lang.NullPointerException
    [java]     at com.goldencode.p2j.uast.SymbolResolver.parseHierarchy(SymbolResolver.java:3662)
    [java]     at com.goldencode.p2j.uast.SymbolResolver.parseHierarchy(SymbolResolver.java:3679)
    [java]     at com.goldencode.p2j.uast.SymbolResolver.loadClass(SymbolResolver.java:3639)
    [java]     at com.goldencode.p2j.uast.ProgressParser.user_defined_type_name(ProgressParser.java:9387)
    [java]     at com.goldencode.p2j.uast.ProgressParser.var_type(ProgressParser.java:49501)
    [java]     at com.goldencode.p2j.uast.ProgressParser.as_clause(ProgressParser.java:22172)
    [java]     at com.goldencode.p2j.uast.ProgressParser.def_var_stmt(ProgressParser.java:12624)
    [java]     at com.goldencode.p2j.uast.ProgressParser.define_stmt(ProgressParser.java:11218)
    [java]     at com.goldencode.p2j.uast.ProgressParser.stmt_list(ProgressParser.java:25216)
    [java]     at com.goldencode.p2j.uast.ProgressParser.statement(ProgressParser.java:8266)
    [java]     at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:6916)
    [java]     at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:6609)
    [java]     at com.goldencode.p2j.uast.ProgressParser.external_proc(ProgressParser.java:6536)
    [java]     at com.goldencode.p2j.uast.AstGenerator.parse(AstGenerator.java:1502)
    [java]     at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:967)
    [java]     ... 7 more
```

#11 - 06/20/2019 05:09 PM - Greg Shah

> Do we support 4GL classes (.cls)?

Yes, most use cases are already supported. See #3751.

You probably need to add the skeletons project to your installation and then point to it in cfg/p2j.cfg.xml using something like <parameter name="oo-skeleton-path" value="./abl/skeleton/" />. This adds all the "built-in" classes which must be present for any OO usage.

#12 - 06/21/2019 03:58 PM - Stanislav Lomany

Adding skeleton helped to pass the errors above. However I cannot convert a file referencing a class. I suppose it may be a configuration issue (prefixes etc.). Do you have a sample project with OO working?

```
rObj = new table.help.oo.InputStatic() no-error.
```

```
[java] ------------------------------------------------------------------------------
     [java] Code Conversion Annotations
     [java] ------------------------------------------------------------------------------
     [java]
     [java] Optional rule set [customer_specific_annotations_prep] not found.
     [java] ./abl/table/static/io/test_input_by_reference_oo.p
     [java] WARNING: MISSING OO NAME MAPPING for file ./table/help/oo/InputStatic.cls (qualified OO name table
.help.oo.inputstatic, id 12884902958)
     [java] WARNING: MISSING OO NAME MAPPING for file ./table/help/oo/InputStatic.cls (qualified OO name table
.help.oo.inputstatic, id 12884902967)
     [java] WARNING: MISSING OO NAME MAPPING for file ./table/help/oo/InputStatic.cls (qualified OO name table
.help.oo.inputstatic, id 12884902969)
     [java] WARNING: MISSING OO NAME MAPPING for file ./table/help/oo/InputStatic.cls (qualified OO name table
.help.oo.inputstatic, id 12884902971)
     [java] WARNING: MISSING OO NAME MAPPING for file ./table/help/oo/InputStatic.cls (qualified OO name table
.help.oo.inputstatic, id 12884902989)
     [java] WARNING: MISSING OO NAME MAPPING for file ./table/help/oo/InputStatic.cls (qualified OO name ./tab
le/help/oo/InputStatic.cls, id 12884902990)
     [java] WARNING: MISSING OO NAME MAPPING for file ./table/help/oo/InputStatic.cls (qualified OO name table
.help.oo.inputstatic, id 12884903095)
     [java] WARNING: MISSING OO NAME MAPPING for file ./table/help/oo/InputStatic.cls (qualified OO name ./tab
le/help/oo/InputStatic.cls, id 12884903096)
     [java] WARNING: MISSING OO NAME MAPPING for file ./table/help/oo/InputStatic.cls (qualified OO name table
.help.oo.inputstatic, id 12884903281)
     [java] WARNING: Null annotation (full-java-class) for define [DEFINE_VARIABLE] @144:1 (12884902948)
     [java] WARNING: Null annotation (simple-java-class) for define [DEFINE_VARIABLE] @144:1 (12884902948)
     [java] WARNING: Null annotation (containing-package) for define [DEFINE_VARIABLE] @144:1 (12884902948)
```

#13 - 06/23/2019 04:56 PM - Greg Shah

Most of the code in testcases/uast/oo/ converts OK (not all, but most). Please post your sample here for review.

#14 - 06/28/2019 06:18 PM - Stanislav Lomany

- File conv.log added

> Most of the code in testcases/uast/oo/ converts OK (not all, but most). Please post your sample here for review.

I used FWD testcases project to convert a single file using a class. Whatever I do, it ends up with similar errors. I've attached log.

#15 - 06/29/2019 01:33 AM - Greg Shah

Please post both the test driver and the Printme.cls or commit them to testcases/uast/.

One thing that you MUST do is to include the oo/Printme.cls in the conversion list. I suspect that is your issue.

#16 - 07/02/2019 04:46 PM - Stanislav Lomany

> One thing that you MUST do is to include the oo/Printme.cls in the conversion list. I suspect that is your issue.

Yes, thank you, it worked!

#17 - 07/09/2019 09:31 PM - Stanislav Lomany

There is a procedure dataset/help/get_tt_num_rows.p which is used by pretty much all testcases (which now they are not properly executed). This procedure has this line of code:

```
TemporaryBuffer.createDynamicTable(_tableHandle, tableHandle, true, false);
```

This line is not emitted (which is a bug) if another specific procedure named dataset/help/cmp_tt_schema.p is present in the conversion list. I'll check what causes this behavior.

#18 - 07/10/2019 09:47 AM - Stanislav Lomany

Consider we have two procedures:

```
define input parameter table-handle tableHandle1.
message "test1".
```

and

```
define input parameter table-handle tableHandle2.
message "test2".
```

If they are converted in one batch, one of them misses

```
TemporaryBuffer.createDynamicTable(_tableHandle, tableHandle, true, false);
```

statement.

#19 - 07/10/2019 03:37 PM - Stanislav Lomany

Could someone please confirm the issue from the previous note? That it's not an issue with my RAM.

#20 - 07/10/2019 03:45 PM - Greg Shah

Do you have a standalone testcase I can use?

#21 - 07/10/2019 03:47 PM - Stanislav Lomany

- File test.p added

- File test2.p added

Yes, convert both attached files in a single run and check if TemporaryBuffer.createDynamicTable presents in both converted files.

#22 - 07/10/2019 04:13 PM - Greg Shah

- File SvlTest.java added

- File SvlTest2.java added

One has it and the other does not. I've attached my output (I changed the .p names but the code is the same).

#23 - 07/10/2019 04:25 PM - Stanislav Lomany

Good, thank you! convert/buffer_definitions.rules has this variable

```
<variable name="lastthparam"   type="com.goldencode.ast.Aast" />
```

which is not reset between conversion of different files. How is that possible?

Sure it can be fixed by putting

```
<rule>lastthparam = null</rule>
```

into the init-rules.

#24 - 07/11/2019 09:11 AM - Stanislav Lomany

Okay, I fixed NPE and now I can get meaningful results for the testcases. It is about a hundred of testcases. I didn't run all of them because only one of them finished without errors. I saw around 10 different types of errors. It doesn't seem to be real to provide a meaningful estimate without putting efforts into categorization of error, so I'll just work on the issues sequentially. Overall task looks like a substantial chunk of issues to fix.

#25 - 07/11/2019 09:45 AM - Stanislav Lomany

Created task branch 3816a from P2J trunk revision 11324.

#26 - 07/11/2019 10:47 AM - Greg Shah

The priority should be on the implementation of conversion and runtime support for the core features listed in #3816-1. We need this in the trunk ASAP.

Then a second pass can be done to get all error handling exactly right.

#27 - 07/11/2019 01:59 PM - Greg Shah

FYI, branch 3809c has significant changes/improvements to table parameter processing (and dataset parameter processing). You will want to start with that revision as your foundation. The intention is to get it into the trunk today.

#28 - 07/14/2019 04:19 AM - Stanislav Lomany

Rebased task branch 3816a from P2J trunk revision 11325.

#29 - 07/15/2019 10:22 AM - Stanislav Lomany

- File tests.log added

After I've rebased to the latest trunk, the errors of the following type occur:

```
    [javac] /home/svl/projects/tests/src/com/goldencode/tests/dataset/help/IoStaticParameters.java:383: error:
 no suitable method found for associate(DataSetParameter,DataSet,boolean,boolean,boolean)
    [javac]          DataSet.associate(dsMaster_1, dsMaster, true, false, true);
    [javac]
    [javac] /home/svl/projects/tests/src/com/goldencode/tests/dataset/help/IostaticParameters.java:239: error:
 no suitable constructor found for DataSetParameter(DataSetParameter,boolean,ParameterOption)
    [javac]          doMasterBind(new DataSetParameter(new DataSetParameter(hds), false, BIND));
    [javac]
    [javac] /home/svl/projects/tests/src/com/goldencode/tests/dataset/static_/io/TestDatasetBindOo.java:156: e
rror: no suitable constructor found for DataSetParameter(DataSetParameter)
    [javac]          silent(() -> hproc.ref().doMasterBind(new DataSetParameter(new DataSetParameter(hds))));
```

Full log attached.

#30 - 07/16/2019 05:28 PM - Stanislav Lomany

Some thoughts about BIND:

1. Temp-tables declared as REFERENCE-ONLY are initialized in FWD as usual buffers (i.e. StaticTempTable is created and multiplex scope is opened). Shouldn't we correct this behavior?
2. Tables passed by reference are actually copied in FWD. This can be fixed because BY-REFERENCE can be considered as a limited way of binding.
3. The main issue I'm considering is how to implement parameter/buffer binding, especially if the bound table is referenced from outside its current outermost scope.

#31 - 07/16/2019 08:24 PM - Greg Shah

> Temp-tables declared as REFERENCE-ONLY are initialized in FWD as usual buffers (i.e. StaticTempTable is created and multiplex scope is opened). Shouldn't we correct this behavior?

Yes. Any deviations or limitations or missing features need to be resolved.

I assume the point is that there is a passed through reference to the same instance. Make sure to handle any special cases. I seem to recall that there are some limits on which cases can be marked REFERENCE-ONLY. Please list the rules here.

#32 - 07/18/2019 08:55 PM - Stanislav Lomany

Guys, I'm interested, how technically we can pass a table parameter by reference? In this case 4GL passes the default buffer of the table which replaces the default buffer of the table on the calling side temporary (BY-REFERENCE) or until the routines' life time (BIND).

1. The BY-REFERENCE case is the simplest one. We can make these calls for an existing target buffer tt1:

```
Tt1_1_1.Buf save = tt1.
tt1 = RecordBuffer.defineAlias(srcBuf...);
openScope(tt1);
...
tt1 = save;
```

   Does that seem correct?
2. BIND call when the target table is reference-only: we unbind the previous bond (if any) and make

```
tt1 = RecordBuffer.defineAlias(srcBuf...)
```

   call.
3. BIND call when the source table is reference-only: I'm not sure how it is supposed to look on the calling side.

#33 - 07/19/2019 01:13 PM - Eric Faulhaber

Stanislav Lomany wrote:

> Guys, I'm interested, how technically we can pass a table parameter by reference? In this case 4GL passes the default buffer of the table which replaces the default buffer of the table on the calling side temporary (BY-REFERENCE) or until the routines' life time (BIND).
>
> 1. The BY-REFERENCE case is the simplest one. We can make these calls for an existing target buffer tt1:
>   [...]
>   Does that seem correct?

This seems reasonable to me, but I am basing that only on my limited understanding of what this option means. I have not used it in practice.

1. BIND call when the target table is reference-only: we unbind the previous bond (if any) and make
   [...]
   call.
2. BIND call when the source table is reference-only: I'm not sure how it is supposed to look on the calling side.

That's a good question. Today we define a buffer using a simple instance variable assignment (declaration + definition when the external procedure class is instantiated). Would we have to define the buffer on the calling side in a way that can be mutable, if we know it is passed as a parameter somewhere in the program with the BIND option?

#34 - 07/19/2019 07:20 PM - Constantin Asofiei

I found in a project code like this:

```
    method public void doSomething (output dataset-handle ds1):
        GetDatasetCopy (dataset dsSource:handle, input-output dataset-handle ds1).
        /* delayed delete (by-value dataset)*/
        delete object ds1 no-error.
    end method.
```

The code does something like "copy data from dsSource to ds1", and also deletes ds1 (a dynamic dataset)! In FWD, this is done immediately, and thus the dataset is empty.

The delayed delete comment in the code is intriguing - it suggests that 4GL delays the delete (maybe until after all references to it have been discarded?). There is a DATASET:NUM-REFERENCES attribute (and for buffer and temp-table, too), but the docs state that it applies only to reference-only parameters. And it might make sense, because this method is called like:

```
FetchException(output dataset ds1 by-reference).
```

There is a good delete behavior that this code is related to BY-REFERENCE, which delays the delete until all NUM-REFERENCES is zero. I haven't read all the comments here, but this attribute I think is incremented when the by-reference argument is passed to another call.

#35 - 07/19/2019 07:56 PM - Constantin Asofiei

Looks like this states that the 'delete will not happen if the DATASET is not owned by the procedure attempting the delete': https://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/dvpds%2Fdeleting-a-dynamic-prodataset-passed-as-a-parame.html%23

And I think the root cause for what I'm checking is that FWD is deleting a static dataset...

#36 - 07/19/2019 08:36 PM - Constantin Asofiei

The static dataset delete was just a minor issue. Even if I'm assuming that INPUT-OUTPUT behaves the same as BY-REFERENCE, I'm advancing just a little in the code. We need to figure out how chain calls act, when the source is DATASET ds1 BY-REFERENCE and after that this same instance is passed as OUTPUT or INPUT-OUTPUT DATASET-HANDLE ds1 to other methods. I think somehow 4GL 'knows' that the DATASET is BY-REFERENCE, while FWD does stuff like creating a dynamic dataset to hold a INPUT or INPUT-OUTPUT passed to a DATASET-HANDLE; and when FWD sees the delete, it thinks the dataset is dynamic, so it deletes it.

BTW, if Marian's tests are too complex, start with something simple and just check the :HANDLE attribute for:

- the received argument at the internal procedure
- if the parameter is DATASET (not DATASET-HANDLE), then check the :HANDLE for the source dataset (in the external program).

This will help you identify easily how BY-REFERENCE cascades...

#37 - 07/20/2019 05:57 AM - Constantin Asofiei

To continue with previous notes, as it may be related to TABLE-HANDLE, too: I think a DELETE for a DATASET-HANDLE argument is postponed until after the procedure has finished (i.e. after any copy output operations have performed). Without BY-REFERENCE, the DATASET-HANDLE does get reported as dynamic in 4GL, so we are good here.

#38 - 07/20/2019 08:44 AM - Constantin Asofiei

Constantin Asofiei wrote:

> To continue with previous notes, as it may be related to TABLE-HANDLE, too: I think a DELETE for a DATASET-HANDLE argument is postponed until after the procedure has finished (i.e. after any copy output operations have performed). Without BY-REFERENCE, the DATASET-HANDLE does get reported as dynamic in 4GL, so we are good here.

I have some fixes for this, but they are from some small testing and that 'it makes sense' - the delete of a DATASET-HANDLE OUTPUT argument gets postponed until after the output parameter has been copied.

#39 - 07/21/2019 12:12 PM - Constantin Asofiei

I think I have a fix for BY-REFERENCE issues I had; kind of a hack (do not delete a handle holding a DATASET, if the handle is registered as a OutputDataSetHandleCopier), but looks like is working. Will commit later today.

OTOH, the NUM-REFERENCES attribute needs to be implemented, for BY-REFERENCE to work properly (e.g. do not delete DATASET if is referenced by a BY-REFERENCE argument).

#40 - 07/21/2019 12:27 PM - Constantin Asofiei

And I need to add the same approach for TABLE-HANDLE parameters with BY-REFERENCE arguments.

#41 - 07/21/2019 01:42 PM - Constantin Asofiei

Constantin Asofiei wrote:

> And I need to add the same approach for TABLE-HANDLE parameters with BY-REFERENCE arguments.

Actually the NPE I was getting was because of a TABLE-HANDLE BY-REFERENCE argument for a TABLE parameter.

#42 - 07/22/2019 10:32 AM - Greg Shah

Constantin wrote:

> we need good tests for BY-REFERENCE, TABLE, TABLE-HANDLE, DATASET, DATASET-HANDLE when combined with a DELETE statement

Marian: Do the tests handle this already?

#43 - 07/22/2019 11:42 AM - Constantin Asofiei

My fixes are in 4124a rev 11361.

#44 - 07/22/2019 06:31 PM - Constantin Asofiei

There's one more case I need to fix: TABLE-HANDLE passed as OUTPUT argument for a TABLE parameter, without using BY-REFERENCE. In this case, I think I should create a copy of the TABLE, and not just pass the handle. Anyway, I need to check.

#45 - 07/23/2019 12:32 AM - Marian Edu

Constantin Asofiei wrote:

> There's one more case I need to fix: TABLE-HANDLE passed as OUTPUT argument for a TABLE parameter, without using BY-REFERENCE. In this case, I think I should create a copy of the TABLE, and not just pass the handle. Anyway, I need to check.

Unless BY-REFERENCE is used always make a copy, even for INPUT-OUTPUT. For TABLE/DATASET-HANDLE the dynamic copy created can (need) to be deleted inside the routine else it might leak. It is usually done in a FINALLY block but it should always be delayed till returned to the caller.

All this should be covered by tests in io folder for dataset and table.

#46 - 07/23/2019 03:38 AM - Constantin Asofiei

Marian Edu wrote:

> All this should be covered by tests in io folder for dataset and table.

Do you have tests for cases like:

- a OUTPUT BY-REFERENCE argument (both TABLE-HANDLE at argument and definition parameter).
- the handle parameter is assigned a newly created dynamic table dt1
- this dt1 dynamic table is assigned to another handle, h2
- you do a delete on dt1 using handle h2 - does this delete the table or not?

My understanding is that a e.g table can't be deleted once was assigned to a BY-REFERENCE parameter, and this somehow depends on the NUM-REFERENCES attribute.

#47 - 07/24/2019 04:55 AM - Marian Edu

Constantin Asofiei wrote:

> Marian Edu wrote:
>
> > All this should be covered by tests in io folder for dataset and table.
>
> Do you have tests for cases like:
>
> - a OUTPUT BY-REFERENCE argument (both TABLE-HANDLE at argument and definition parameter).
> - the handle parameter is assigned a newly created dynamic table dt1
> - this dt1 dynamic table is assigned to another handle, h2
> - you do a delete on dt1 using handle h2 - does this delete the table or not?
>
> My understanding is that a e.g table can't be deleted once was assigned to a BY-REFERENCE parameter, and this somehow depends on the NUM-REFERENCES attribute.

Constantin, it doesn't matter how many handles to you have to one object (temp-table) those are all pointing to the same instance so deleting dt1 is the same as deleting h2. You can delete any dynamic object regardless of how many references to that object exists - we actually have tests for BIND where you can bind a REFERENCE-ONLY table to a dynamic one and later delete the dynamic table... the REFERENCE-ONLY table become again unbind and can't be used.

BY-REFERENCE means that you pass the actual table not a copy of it as it happens when the default BY-VALUE is used. BIND is also like BY-REFERENCE only that the object you have as REFERENCE-ONLY remains bound to the object received even after the routine returns to the caller.

I went through the tests we have in dataset/static/io, dataset/dynamic/io, table/static/io and table/dynamic/io and there are a couple of combinations missing (mixin table with table-handle) and I think the tests should be broken down in smaller ones to be more clear... will post back when we're done updating that.

Other specific questions you might have let me know.

#48 - 07/24/2019 05:26 AM - Greg Shah

> BIND is also like BY-REFERENCE only that the object you have as REFERENCE-ONLY remains bound to the object received even after the routine returns to the caller.

Do I understand correctly that BIND would only have a useful effect when stored as a resource of a persistent procedure or a class instance? In other words, it is useful for resources passed as a parameter that outlive the function/procedure/method in which they were passed.

#49 - 07/24/2019 05:35 AM - Marian Edu

Greg Shah wrote:

> Do I understand correctly that BIND would only have a useful effect when stored as a resource of a persistent procedure or a class instance? In other words, it is useful for resources passed as a parameter that outlive the function/procedure/method in which they were passed.

This is the main use case indeed and it only work when you 'bind' to a REFERENCE-ONLY table or dataset. There are some interesting notes here that you might be interested in when working on implementation:
- if a procedure/class defines a static dataset/temp-table that was passed with BIND to another object, even if you delete the procedure/class that defined it in the first place the reference will remain valid in the callee.
- if you pass a dynamic dataset/temp-table with BIND the callee will have a valid reference as long as the object is not deleted otherwise it goes back to 'reference-only' and it can't be used even if it was one bind to a valid dataset/temp-table.

It would have been better to start by crafting some implementation specs not just writing test-cases although test-driven development could also work I guess.

#50 - 07/24/2019 05:39 AM - Constantin Asofiei

Marian Edu wrote:

> It would have been better to start by crafting some implementation specs not just writing test-cases although test-driven development could also work I guess.

Greg, for this case, the implementation specs will help a lot, and we will have the 'big picture' from the start - otherwise, TDD would be more time-consuming.

#51 - 07/24/2019 05:52 AM - Greg Shah

> It would have been better to start by crafting some implementation specs not just writing test-cases although test-driven development could also work I guess.

Marian: It would be great if your team would write these specs.

#52 - 07/24/2019 05:53 AM - Greg Shah

Greg Shah wrote:

> > It would have been better to start by crafting some implementation specs not just writing test-cases although test-driven development could also work I guess.
>
> Marian: It would be great if your team would write these specs.

The specs probably should also cover dataset and dataset-handle.

#53 - 07/24/2019 01:29 PM - Stanislav Lomany

> Marian: It would be great if your team would write these specs.

While these specs are in progress I suppose we can start discussing/implementing the basic things:

1. When a temp-table is defined as REFERENCE-ONLY, it cannot be used until it is initialized by a RUN call with a source table passed with BY-REFERENCE or BIND "Attempt to reference uninitialized temp-table“.
2. BY-REFERENCE calls.
3. BIND calls.

I like the idea of a buffer being mutable, because it solves all of the issues above. And solves with no (almost) conversion changes. Aren't there any basic things in FWD that'll not allow to write some buf1.assign(buf2) function that will make buf1 variable reference buf2 through some sort of wrapping?

#54 - 07/25/2019 11:15 AM - Eric Faulhaber

Stanislav Lomany wrote:

> I like the idea of a buffer being mutable, because it solves all of the issues above. And solves with no (almost) conversion changes. Aren't there any basic things in FWD that'll not allow to write some buf1.assign(buf2) function that will make buf1 variable reference buf2 through some sort of wrapping?

I suspect there probably are; RecordBuffer manages a lot of state. However, the buffers at the business logic layer are proxies for RecordBuffer instances, so there may be something clever we can do here to switch out the RecordBuffer instance with which a proxy is associated.

#55 - 07/25/2019 12:47 PM - Marian Edu

Stanislav Lomany wrote:

> 1. When a temp-table is defined as REFERENCE-ONLY, it cannot be used until it is initialized by a RUN call with a source table passed with BY-REFERENCE or BIND "Attempt to reference uninitialized temp-table“.

That's correct, REFERENCE-ONLY is used just to have the definition in order to be able to work with the table statically (for each/find) but at that point there is no instance of the table available so any attempt to access it will throw an error. If you have a routine that has that table as input you need to send a valid table instance with BY-REFERENCE and the table will be bound to that instance only for the time of the routine or with BIND when the table will remain bound to the instance even after the routine returns to the caller.

#56 - 07/26/2019 02:42 AM - Constantin Asofiei

The current approach with datasets, where a OUTPUT dsHandle BY-REFERENCE argument is passed to a OUTPUT DATASET parameter is incorrect. I'll think of another approach.

#57 - 07/26/2019 09:26 AM - Ovidiu Maxiniuc

Can you add more detail? Why is it incorrect?

#58 - 07/26/2019 09:40 AM - Constantin Asofiei

Ovidiu Maxiniuc wrote:

> Can you add more detail? Why is it incorrect?

See the changes in 4124a rev 11384. The idea was: if the argument is an unknown handle, and by-reference, then we can't assume 'INPUT-OUTPUT'.

#59 - 07/28/2019 02:05 PM - Constantin Asofiei

Next issue I need solved is REFERENCE-ONLY for DATASET and TEMP-TABLE: these really are the same resources as, for example, the argument associated with it. It also inherits the record on which the source buffer is positioned (this comes naturally, as they are the same resource in 4GL).

Stanislav, did you get a chance to work on this? I can 'emulate' certain features (like ensuring the buffer is on the same record, when entering and exiting a procedure), but, for example, this will not work if we execute something from the caller, and reposition the buffer to another record - the REFERENCE-ONLY will not reflect the record change.

#60 - 07/29/2019 04:00 AM - Marian Edu

Constantin Asofiei wrote:

> Next issue I need solved is REFERENCE-ONLY for DATASET and TEMP-TABLE: these really are the same resources..

Constantin, are you referring to BY-REFERENCE here? REFERENCE-ONLY is just an option on temp-table/dataset definition to not have a full instance created but still have the definition in order to be able to work with it in a statical way. A temp-table/dataset defined with REFERENCE-ONLY can only be used if a full-fledge instance is passed as input so the REFERENCE-ONLY temp-table/dataset become bound to it for the time of the call or indefinitely if BIND is used.

You can look at it like an interface, you have the structure (meta-data) but until you get a real thing bound to it you can't use any functionality of the temp-table/dataset - each attempt to do so will throw an error saying the temp-table/dataset is only a reference and that reference is not pointing to anything useful.

#61 - 07/29/2019 04:20 AM - Constantin Asofiei

Marian Edu wrote:

> Constantin, are you referring to BY-REFERENCE here? REFERENCE-ONLY is just an option on temp-table/dataset definition to not have a full instance created ...

I'm talking about both, as we are not handling properly REFERENCE-ONLY at this time; we create concrete DATASET/TEMP-TABLE instances, as our implementations for these are not mutable (so we can't switch at runtime - yet - to the BY-REFERENCE resource, to be truly 'by reference').

There is some refactoring required on our side to make these mutable. We need to determine how these behave (I know you have testcases for these, but please put together some specifications). Like, can you DELETE a resource once is assigned to a REFERENCE-ONLY? What happens if the BY-REFERENCE argument is not dynamic, and the instantiating program gets deleted, while the resource is in a REFERENCE-ONLY 'shell', does it survive? What happens if BIND is not used, but you have something like:

- program p1 defines a REFERENCE-ONLY table/dataset, r1
- program p2 calls a procedure proc0 in p1 with BY-REFERENCE - so r1 holds the resource from program p2
- while in proc0, this ends up calling program p3 which in turn calls proc1 from p1, with another BY-REFERENCE, for the same target, r1 - does r1 'evict' the current referenced resource and assigns itself a new one, or there is an error, as r1 already holds a reference? If there is no error, does r1 get restored once proc1 finishes, to the resource from proc0?

Same for BIND - if you try to call BY-REFERENCE again, without BIND, and the REFERENCE-ONLY already has a resource, does this get an error? I'm interested in rules which determine:

- the 'lifetime' of the REFERNCE-ONLY resource, and if it can be switched to another reference, if it already holds one

- how DELETE behaves when the referenced resource is:
  - dynamic
  - static
  - at an argument with or without BY-REFERENCE (are you forced to use BY-REFERENCE if the target is REFERENCE-ONLY)
- what if there is an unknown DATASET-HANDLE/TABLE-HANDLE argument - does REFERENCE-ONLY remain 'unset'?
- how does INPUT and OUTPUT modes affect BY-REFERENCE? My understanding in this case is that BY-REFERENCE passes the real pointer, so it is as if the mode was INPUT-OUTPUT.

#62 - 07/29/2019 04:51 AM - Marian Edu

I'm just saying there is no real dependency between REFERENCE-ONLY and BY-REFERENCE. For BIND however one of the instances must be BY-REFERENCE, either in caller (OUTPUT) or in callee (INPUT/INPUT-OUTPUT).

You are right when you say BY-REFERENCE is passing the real pointer, however this is not exactly how INPUT-OUTPUT works unless BY-REFERENCE is used of course :)

By default INPUT-OUTPUT pass the data structure around BY-VALUE, aka it makes a copy of the structure in the caller and pass that to the callee, when the routine returns the structure returned will actually override the initial structure in the caller but that only happens if the routine does not return/throw an error in which case the structure in the caller remains unchanged.

I need to finish all tests to make sure we cover everything and will prepare the specification to make it clear.

#63 - 07/29/2019 06:22 PM - Stanislav Lomany

> I can 'emulate' certain features (like ensuring the buffer is on the same record, when entering and exiting a procedure), but, for example, this will not work if we execute something from the caller, and reposition the buffer to another record - the REFERENCE-ONLY will not reflect the record change.

Constantin, as far as I tested, BY-REFERENCE makes caller and callee reference the same buffer, therefore changes are always reflected regardless how the buffer is accessed. Hope my comment suits what you meant to say. Anyway, rules have to be determined.

> Stanislav, did you get a chance to work on this?

No, I didn't work on it.

#64 - 07/31/2019 01:00 PM - Eric Faulhaber

Constantin, et al.: what are we thinking, in terms of an implementation of BY-REFERENCE for buffers in FWD?

Do we need the ability to re-associate a different RecordBuffer instance with a Buffer proxy, such that a callee for a BY-REFERENCE buffer can reassign the underlying RecordBuffer instance to the one it is using?

In other words, is this the situation?...

Procedure A defines buffer proxy bufA, passes a reference to bufA to procedure B, which has defined its own buffer proxy bufB (defined for the same backing table as bufA). Then procedure B calls something like bufA.assign(bufB), such that the RecordBuffer instance underlying bufB becomes the same instance as underlies bufA?

Or is it the other way around? That is, procedure B calls bufB.assign(bufA), such that the RecordBuffer instance underlying the caller's bufA becomes the same instance as that which underlies bufB?

In either case, what happens to the "discarded" RecordBuffer instance?

Or have I misunderstood the intended design?

#65 - 07/31/2019 01:15 PM - Constantin Asofiei

Eric Faulhaber wrote:

> Do we need the ability to re-associate a different RecordBuffer instance with a Buffer proxy, such that a callee for a BY-REFERENCE buffer can reassign the underlying RecordBuffer instance to the one it is using?

Yes, but not only that, the BufferImpl instance needs to be switched, too.

> Procedure A defines buffer proxy bufA, passes a reference to bufA to procedure B, which has defined its own buffer proxy bufB (defined for the same backing table as bufA).
> ...
> Or is it the other way around? That is, procedure B calls bufB.assign(bufA), such that the RecordBuffer instance underlying the caller's bufA becomes the same instance as that which underlies bufB?

This one. You have something like run proc0 in h (input table bufA by-reference) in procedure A (an external program) and in procedure B you have:

```
def temp-table bufB field f1 as int.
procedure proc0.
def input paramater table for bufB.
end.
```

In procedure B, bufB will be the same resource (BUFFER and TEMP-TABLE!) as the on passed by procedure A. And looks like any other explicit buffer defined in procedure B for bufB will have the temp-table from procedure A, too! So is not just switching the reference in the target buffer, we need to adjust all explicitly defined buffers for a temp-table, too.

> In either case, what happens to the "discarded" RecordBuffer instance?

My understanding/assumption (from some minimal testing) is:

- if BY-REFERENCE is used, then the reference in procedure B gets restored to whatever is set prior to this call
- if BIND is used, then the reference remains set - but this is possible only with REFERENCE-ONLY; don't know what happens if other attempts at calling procedure B after bufB was bound.

  Or have I misunderstood the intended design?

The idea here is that we need the Java structure we use now (a proxy) to be mutable; the concept in 4GL is to pass a table as by-reference, not a buffer. But in FWD we don't have converted Java fields which map a 4GL temp-table, we have only buffers. So we need to ensure that any buffers defined for a temp-table which has been 'switched' are updated, too.

Also, another unknown is what happens with the buffer's referenced record, for the implicit and explicit buffers, in 4GL - do they get restored after the procedure exits?

#66 - 07/31/2019 01:25 PM - Eric Faulhaber

Hm, I was thinking of making the proxy itself mutable, but the only instance data the proxy holds is the invocation handler, which has an implicit this reference to its enclosing RecordBuffer instance. It sounds like that will not be enough, since we have state also in BufferImpl and associated with the underlying temp-table.

#67 - 07/31/2019 01:33 PM - Eric Faulhaber

I should clarify that last statement. The proxy instance also holds instance data which BufferImpl (the proxy's parent class) defines. Should we push BufferImpl's instance variables into RecordBuffer, such that BufferImpl holds no instance state of its own? Thus, swapping out the underlying RecordBuffer (actually TemporaryBuffer in this case) would effectively meet the need of swapping the BufferImpl instance. This would also ensure that we are using the same multiplexID, so we are accessing the correct, virtual temp-table. Is this enough?

#68 - 07/31/2019 02:01 PM - Eric Faulhaber

Constantin Asofiei wrote:

> Also, another unknown is what happens with the buffer's referenced record, for the implicit and explicit buffers, in 4GL - do they get restored after the procedure exits?

This I don't know. Marian?

In terms of FWD's implementation, if we needed to "un-assign" the buffer (i.e., invocation handler instance) which had been assigned, we could store the proxy's original invocation handler and restore it when the procedure exits.

#69 - 07/31/2019 02:14 PM - Constantin Asofiei

Eric Faulhaber wrote:

> In terms of FWD's implementation, if we needed to "un-assign" the buffer (i.e., invocation handler instance) which had been assigned, we could store the proxy's original invocation handler and restore it when the procedure exits.

I think the rule is to restore to whatever instance was before the switch occurred (so not necessarily the 'original invocation handler instance'; look at it like a stack, where you push a new value when the switch occurs, and you pop/restore the value when the procedure ends.

#70 - 07/31/2019 02:29 PM - Eric Faulhaber

Constantin Asofiei wrote:

> Eric Faulhaber wrote:
>
> > In terms of FWD's implementation, if we needed to "un-assign" the buffer (i.e., invocation handler instance) which had been assigned, we could store the proxy's original invocation handler and restore it when the procedure exits.
>
> I think the rule is to restore to whatever instance was before the switch occurred (so not necessarily the 'original invocation handler instance'; look at it like a stack, where you push a new value when the switch occurs, and you pop/restore the value when the procedure ends.

Yes, I didn't explain it very well, but this is roughly the idea I had in mind. However, I haven't done any testing in this area and I have limited time to work on this, so I need some guidance as to whether the core design idea of doing this at the proxy level is feasible (assuming the BufferImpl instance becomes stateless and all its current state is pushed into RB).

#71 - 07/31/2019 02:32 PM - Constantin Asofiei

Eric Faulhaber wrote:

> ... so I need some guidance as to whether the core design idea of doing this at the proxy level is feasible (assuming the BufferImpl instance becomes stateless and all its current state is pushed into RB).

I don't really see how you can make BufferImpl stateless, and still keep the 4GL's resource behavior. But yes, if BufferImpl is stateless, then your idea should work.

#72 - 07/31/2019 02:33 PM - Eric Faulhaber

Hm, I just realized BufferImpl inherits all the instance data from HandleChain and its ancestry. This plan is getting too messy :-(

#73 - 07/31/2019 02:42 PM - Constantin Asofiei

Eric Faulhaber wrote:

> Hm, I just realized BufferImpl inherits all the instance data from HandleChain and its ancestry. This plan is getting too messy :-(

I wouldn't insist on refactoring RecordBuffer and the like, if it we could switch the field reference (at the Java level) in any and all cases. But there are cases where the source and target tables are mapped by different DMOs, so the assignment can't happen at the Java level.

#74 - 07/31/2019 03:30 PM - Constantin Asofiei

The mess is getting even messier. The source and target buffers must match only on the number of fields and their types (in the proper order), as far as I understand. The field names can differ. An example:

- procedure A

```
def temp-table ttA field fA as int.

def var h as handle.
run procB.p persistent set h.

run proc0 in h (input table ttA by-reference).
```

- procedure B

```
def temp-table ttB field fB as int.

procedure proc0.
   def input parameter table for ttB.

   create ttB.
   ttB.fB = 10.
   buffer ttB::fA = 20.
end.
```

This shows that at the target (in procedure B), after we are making the switch, the fields in the 'switched buffer' can be:

- referenced via the conversion-time name
- if :: operator is used, referenced only by the runtime field name (for the current buffer resource, held in this 'shell', as Ovidiu names it).

So, when we are making the switch, we need a mapping of original field names (not from the current 'invocation handler instance', but the original one) to the associated field from the 'switched buffer'.

#75 - 07/31/2019 06:35 PM - Eric Faulhaber

A few questions on your last post...

What happens if ttB does not match the structure (i.e., type and order of fields) of ttA? Is this a compile error (I'm not sure the compiler would have enough information to raise an error)? Or since the caller and callee cannot be referencing the same object, is the BY-REFERENCE option ignored and the relationship between the parameter passed from procedure A and the parameter defined by proc0 in procedure B defaults to a different type (e.g., BIND)?

Is the list of ways fields can be referenced in the called procedure you provided exhaustive? I imagine not. For instance, even though ttB::fA is a legal reference in proc0, a reference of ttB::fB (to the same field) is still legal, correct?

Do we have a set of test cases already defined which can help with these BY-REFERENCE rules?

#76 - 07/31/2019 07:57 PM - Greg Shah

> The source and target buffers must match only on the number of fields and their types (in the proper order), as far as I understand. The field names can differ.

As noted in #3751-492, for method parameters the exact matching of tables is based on:

- The structural comparison rules check the number of fields, type of each field, extent of each field and the order these fields appear. There are the only cases are detected as differences in structure.
- The following things are are ignored: table names, table options, field names, field options (other than EXTENT), anything to do with indexes.

It is my understanding that methods are treated as internal procedures for void return and as functions for non-void returns. Perhaps the parameter processing is handle the same way for all three cases.

It is worth reading that entry, there are more details in regard to tables and table-handles as method parameters.

#77 - 08/01/2019 08:26 AM - Ovidiu Maxiniuc

Actually the indexes must be a match.
I tried to run the following test procedure:

```
DEFINE TEMP-TABLE tT-1RO /*REFERENCE-ONLY*/
   FIELD F11RO AS CHAR
//   INDEX finxRO AS UNIQUE F11RO
   .

DEFINE TEMP-TABLE tT-1
```

```
   FIELD F11 AS CHAR
   INDEX finx AS UNIQUE F11
   .

PROCEDURE test-it:
   DEFINE INPUT PARAMETER TABLE FOR tT-1RO BIND.

   CREATE tT-1RO. F11 = "C". F12 = 17. F13 = 02/17/98. F14 = 17.01.
   CREATE tT-1RO. F11 = "A". F12 = 11. F13 = 11/16/66. F14 = 11.16.
//   CREATE tT-1RO. F11 = "C". F12 = 17. F13 = 02/17/99. F14 = 17.02.
END.

RUN test-it(INPUT TABLE tT-1 BY-REFERENCE).
```

and I got the following error:
test-it p32099_Untitled2.ped Cannot bind BY-REFERENCE parameter tables or datasets unless column datatypes, positions, extents and indexes match for tables tT-1 and tT-1RO. (12766)

However, the name of the index is not important. Uncommenting the finxRO will make the test run with success. Somehow P4GL maps the right indexes with same order of fields. I tested with multiple indexes, in different order. The also work, **as long as the name of each index in the TEMP-TABLE passed as parameter is a unique prefix for the name an index the target TEMP-TABLE** (as it is in this example). Then, if a name of a passed TEMP-TABLE parameter is a prefix for more indexes names in target TEMP-TABLE will not work.

#78 - 08/01/2019 09:06 AM - Greg Shah

Actually the indexes must be a match.

I wonder if my testing was previously somehow incorrect or not extensive enough. My method parameter testing did not see a difference when one table had an index and the other did not. But I didn't try different index configurations.

Questions:

- Is the index dependency only for these BY-REFERENCE or BIND cases? I was not testing those at all. Considering that the default table/table-handle approach is to copy the contents into the target procedure's table, I guess that indexes don't matter in that case. We should check this.
- Can this condition be properly described as "all indexes in the source table must have matching/compatible indexes in the target table"? What happens if either of the target table has more indexes (but is a superset so that all source table indexes are matched)?
- Does this hold for non-unique indexes as well?

  However, the name of the index is not important. Uncommenting the finxRO will make the test run with success. Somehow P4GL maps the right indexes with same order of fields. I tested with multiple indexes, in different order. The also work, as long as the name of each index in the TEMP-TABLE passed as parameter is a unique prefix for the name an index the target TEMP-TABLE (as it is in this example). Then, if a name of a passed TEMP-TABLE parameter is a prefix for more indexes names in target TEMP-TABLE will not work.

The statement "the name of the index is not important" seems to contradict the later statement "The also work, as long as the name of each index in the TEMP-TABLE passed as parameter is a unique prefix for the name an index the target TEMP-TABLE".

I guess what you mean is that "the name of the index is not important **if there is only one index in the each of the source and target tables**"?

In the multi-index case, I don't understand why the name of the index would matter at all. The structure of the index is what makes it uniquely identifiable not the name. Or is it possible to have two identical indexes with different names in the same table?

#79 - 08/01/2019 10:56 AM - Ovidiu Maxiniuc

Sorry, I was not very explicit because I reached some of the conclusion while writing the note. Here is an more complex example that illustrate it. Just the table defs are changed.

Test 1: valid.

```
DEFINE TEMP-TABLE tT-1RO
   FIELD F11RO AS CHAR
   FIELD F12RO AS INT
   FIELD F13RO AS DATE
   FIELD F14RO AS DECIMAL
   INDEX finx1RO AS UNIQUE F11RO
   INDEX finx3RO AS UNIQUE F11RO F13RO
   INDEX finx2RO AS UNIQUE F12RO
   .

DEFINE TEMP-TABLE tT-1
   FIELD E11 AS CHAR
   FIELD E12 AS INT
   FIELD E13 AS DATE
   FIELD E14 AS DECIMAL
   INDEX finx3 AS UNIQUE E11 E13
   INDEX finx1 AS UNIQUE E11
   INDEX finx2 AS UNIQUE E12
   .
```

Notice that the order of the indexes is not important (but the order of the fields in a index is, of course). Also each index in tT-1 is is unique pair in tT-1RO. The names are prefixes.

Test 2: invalid.

```
DEFINE TEMP-TABLE tT-1RO
   FIELD F11RO AS CHAR
   FIELD F12RO AS INT
   FIELD F13RO AS DATE
   FIELD F14RO AS DECIMAL
   INDEX finx1RO AS UNIQUE F11RO
   INDEX finx3RO AS UNIQUE F11RO F13RO
   INDEX finx2RO AS UNIQUE F12RO
   .

DEFINE TEMP-TABLE tT-1
   FIELD E11 AS CHAR
   FIELD E12 AS INT
   FIELD E13 AS DATE
   FIELD E14 AS DECIMAL
   INDEX finx3 AS UNIQUE E11 E13
   INDEX finx1 AS UNIQUE E11
   INDEX finx AS UNIQUE E12 // this was names finx2 in Test #1
   .
```

I renamed the finx2 from tT-1 into finx. In this case the bijective association also exists, each index in tT-1RO has an index in tT-1 whose name is a prefix, but finx now prefixes more than an index in tT-1RO. This is an invalid configuration for ABL, even nothing changes in the structure.

#80 - 08/01/2019 12:02 PM - Eric Faulhaber

Guys, what do you think about this approach?

Instead of returning void, TemporaryBuffer.associate returns <T>, where <T> is a buffer proxy assignable to the destination procedure's buffer. Within TB.associate, we determine whether the BY-REFERENCE matching rules are met. If so, we return the buffer proxy instance passed in by the caller. If not, we define a buffer proxy locally, as we do today. The associate return value is assigned to the local buffer proxy reference in the called procedure.

This approach relies on the assumption that the buffer declaration in the called procedure is scoped to that procedure, and not more broadly. Is this a valid assumption? My thinking is yes, because it should be a parameter defined at that scope, but I need confirmation.

Potential complication: since the field names can differ for the same fields (i.e., those whose type and order match), we will need some sort of ordinal mapping in the invocation handler, so that a reference to ttB::fB actually invokes ttA::fA, if those field references are determined to reference the same field.

Thoughts?

#81 - 08/01/2019 12:07 PM - Constantin Asofiei

Eric Faulhaber wrote:

> Instead of returning void, TemporaryBuffer.associate returns <T>, where <T> is a buffer proxy assignable to the destination procedure's buffer.

This is the main reason it will not work: the source and target DMOs may be different. And the Java-style assignment will fail. Something similar I'm doing right now (but via reflection), and found cases where the assignment fails.

#82 - 08/01/2019 12:08 PM - Constantin Asofiei

Eric Faulhaber wrote:

> Potential complication: since the field names can differ for the same fields (i.e., those whose type and order match), we will need some sort of ordinal mapping in the invocation handler, so that a reference to ttB::fB actually invokes ttA::fA, if those field references are determined to reference the same field.

This is not about the :: (dereference) operator - at conversion time, the destination table is usage is hard-coded to that field names, and they will be accessed like ttB.getFB(). But, if ttA has the field named fA and it was BY-REFERENCE'd to ttB, then ttB.getFB() must route this call and retrieve field fA.

#83 - 08/01/2019 01:28 PM - Eric Faulhaber

As I understand it, there are 3 cases of field dereferencing within the callee procedure, which need to be supported for the BY-REFERENCE case:

1. Static field references to the locally defined temp-table, such as ttB.fB. I assume the callee cannot make static references to the caller's temp-table's non-matching field names, since that should not compile.
2. Dynamic field references to the locally defined temp-table, such as ttB::fB or h-ttB:BUFFER-FIELD("fB").
3. Dynamic field references to the caller's temp-table, such as ttB::fA or h-ttB:BUFFER-FIELD("fA").

It seems like all these cases can be supported as follows.

TB.associate returns <T>, where <T> is a reference to the **locally** defined buffer (i.e., the callee's version). Within TB.associate, we determine whether the BY-REFERENCE rules are met. If not, we define and return a normal buffer proxy for the local buffer, as today.

If we determine we are in BY-REFERENCE mode, we create a special Buffer proxy. It is similar in nearly all respects to how we create the proxy today. It is a BufferImpl instance holding all the same instance variable state as today (including all instance variables of its ancestor classes). The key differences:

- The proxy's invocation handler maps to the caller's RecordBuffer (i.e., that passed within the TableParameter). Static getter/setter calls on ttB are delegated to the corresponding properties of ttA, using an additional mapping layer to deal with the possibility of the method names being different.
- The way we manage the BufferFieldImpl look up for case 3 above (dynamic field references to the caller's temp-table) would change slightly. Case 2 would be handled as it is today.

Thoughts?

#84 - 08/01/2019 02:39 PM - Marian Edu

I don't know much about how you have that implemented but a temp-table is a data structure in 4gl, it has a default buffer but can have more buffers attached either static or dynamic. If you pass the table by-reference you send the table reference/handle... aka, the table inside the callee will point to the same instance as the caller. If you define or create a buffer in the callee will be on the same temp-table instance during the call. If you change the position on the default buffer handle the same position will be in the caller after the call. Whatever changes are done on a by-reference table will be reflected in the caller as it really is the same object instance, this is true even if the routine returns an error.

The check on the table structure match is more restrictive when BY-REFERENCE is used where indexes matters while for BY-VALUE it doesn't matter if the indexes are different or not present at all.

IMHO, BY-REFERENCE should be easier than BY-VALUE as you really just need to pass the the structure of the caller matches the one of the callee and if so pass the object instance, much like for TABLE-HANDLE.

#85 - 08/01/2019 02:51 PM - Eric Faulhaber

Thanks, Marian. I think my design will handle the BY-REFERENCE case, which is complicated a bit by the fact that FWD has different object instances representing the temp-table in the caller and the callee. Since these are proxies, we can play some games to make sure the correct object is referenced under the covers.

Marian Edu wrote:

> Whatever changes are done on a by-reference table will be reflected in the caller as it really is the same object instance, this is true even if the routine returns an error.

Since this is the same object, is my dynamic field dereference scenario 2 described in #3816-83 even legal, or would that produce a runtime error?

Constantin, Ovidiu, if you think the above plan is feasible, can one or both of you work on the conversion changes and the TB.associate method changes, while I work on the new TemporaryBuffer API to define the special buffer proxy?

#86 - 08/01/2019 03:13 PM - Constantin Asofiei

Eric Faulhaber wrote:

> TB.associate returns <T>, where <T> is a reference to the **locally** defined buffer (i.e., the callee's version). Within TB.associate, we determine whether the BY-REFERENCE rules are met. If not, we define and return a normal buffer proxy for the local buffer, as today.

Here, you plan to update the DMO Java field, for the default buffer in FWD, right? I mean, you don't plan to hide the Java class field with some method-level variable. Also, note that we need to restore the field's value back, once the method ends.

> If we determine we are in BY-REFERENCE mode, we create a special Buffer proxy. It is similar in nearly all respects to how we create the proxy today. It is a BufferImpl instance holding all the same instance variable state as today (including all instance variables of its ancestor classes). The key differences:

Will this BufferImpl reflect any changes back to the caller's buffer (bufA)? Also, while bufA is BY-REFERENCE'd in bufB, any and all changes made to bufB (TEMP-TABLE or default BUFFER) are visible in both bufA (at the caller) and bufB (at the callee). They really are the same resource; you can end up back in procA while bufA is BY-REFERENCE'd in bufB, and bufA (and ttA) will see any and all changes made in procB.

I just want to make sure that this is clear: in 4GL, when you BY-REFERENCE a temp-table into another temp-table (and also its default buffer), then both have the same resource, and every change is seen immediately in both (and not just when the method ends or something else). Also, other explicit buffers defined in program B for ttB, will see the ttA temp-table, if this was BY-REFERENCE'd into ttB.

- The way we manage the BufferFieldImpl look up for case 3 above (dynamic field references to the caller's temp-table) would change slightly. Case 2 would be handled as it is today.

While bufA is BY-REFERENCE'd in bufB, you can't dereference via :: the fields using bufB's field names - only bufA fields work. So, ttB::fB is not valid, only ttB::fA is valid.

#87 - 08/01/2019 03:15 PM - Ovidiu Maxiniuc

I think it might work. The new local BUFFER will keep the reference of the passed-in BUFFER and will delegate all methods to it. The field access must be done through this map created at the moment the BUFFER is associated.
There are a few issues with the implementation I see at this moment:

- is how we distinct the field access (DMO interface ?) form other method calls?
- building the redirection map: iterate fields in (which? probably position) order and pair setters and getters.
- will all trigger/events fire? will they fire twice (once for each buffer)?

#88 - 08/01/2019 03:16 PM - Constantin Asofiei

Eric Faulhaber wrote:

> Since this is the same object, is my dynamic field dereference scenario 2 described in #3816-83 even legal, or would that produce a runtime error?

This is not legal, the field will not be found, if ttB actually is ttA.

> Constantin, Ovidiu, if you think the above plan is feasible, can one or both of you work on the conversion changes and the TB.associate method changes, while I work on the new TemporaryBuffer API to define the special buffer proxy?

Will this work if the source and target DMO's are not the same? Consider how FWD converts two temp-tables which match, but different field names. The defined type of these Java fields are not compatible with each other.

#89 - 08/01/2019 03:27 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> I think it might work. The new local BUFFER will keep the reference of the passed-in BUFFER and will delegate all methods to it. The field access must be done through this map created at the moment the BUFFER is associated.
> There are a few issues with the implementation I see at this moment:
>
> - is how we distinct the field access (DMO interface ?) form other method calls?

This happens naturally, in that the DMO interface calls go through the invocation handler and the other method calls are made directly on the BufferImpl instance (they are not proxied today, nor would they be in this design). The BufferImpl instance is still created in the context of the called procedure, so it would work the same way as it does today.

> - building the redirection map: iterate fields in (which? probably position) order and pair setters and getters.

I think we iterate in fieldId order. The TableMapper$LegacyTableInfo.ip2j map already provides the mapping by ordinal, but I'll have to double-check whether this is the correct order for this purpose.

- will all trigger/events fire? will they fire twice (once for each buffer)?

They will fire only for the buffer defined by the caller, not for the buffer defined by the callee. We would not be changing the bulk of the logic in the invocation handler, only handling the extra mapping of the getter/setter methods. Everything else continues to operate in the context of the caller's buffer and operates on that buffer's currentRecord.

I guess any "reverse" references to RecordBuffer.dmoProxy within the caller's buffer would have to be fixed up as well, to refer to the callee's proxy temporarily, until the called procedure returns.

#90 - 08/01/2019 03:31 PM - Eric Faulhaber

Constantin Asofiei wrote:

> Eric Faulhaber wrote:
>
> > Since this is the same object, is my dynamic field dereference scenario 2 described in #3816-83 even legal, or would that produce a runtime error?
>
> This is not legal, the field will not be found, if ttB actually is ttA.

OK, good, that actually simplifies things.

> Constantin, Ovidiu, if you think the above plan is feasible, can one or both of you work on the conversion changes and the TB.associate method changes, while I work on the new TemporaryBuffer API to define the special buffer proxy?

> Will this work if the source and target DMO's are not the same? Consider how FWD converts two temp-tables which match, but different field names. The defined type of these Java fields are not compatible with each other.

Sorry, what precisely are you asking about? The remapped getter/setter calls in the invocation handler, or something else? Can you provide an example of your concern in terms of temp-table definitions?

#91 - 08/01/2019 03:32 PM - Ovidiu Maxiniuc

I think the dynamic dereference can be handled separately. In this case, both ttB and ttA will only have the access to the field fA/fB using ttB::fA and ttA::fA. As Constantin noted ttB::fB and ttA::fB are both invalid, once the association took place.

Because the all calls are 'live' redirected to ttA buffer, the changes done on ttB will simultaneously be seen in ttA and on any other ttC where ttA might have been passed a BIND/BY-REFERENCE parameter.

#92 - 08/01/2019 03:42 PM - Eric Faulhaber

Constantin Asofiei wrote:

> Will this BufferImpl reflect any changes back to the caller's buffer (bufA)? Also, while bufA is BY-REFERENCE'd in bufB, any and all changes made to bufB (TEMP-TABLE or default BUFFER) are visible in both bufA (at the caller) and bufB (at the callee). They really are the same resource; you can end up back in procA while bufA is BY-REFERENCE'd in bufB, and bufA (and ttA) will see any and all changes made in procB.
>
> I just want to make sure that this is clear: in 4GL, when you BY-REFERENCE a temp-table into another temp-table (and also its default buffer), then both have the same resource, and every change is seen immediately in both (and not just when the method ends or something else). Also, other explicit buffers defined in program B for ttB, will see the ttA temp-table, if this was BY-REFERENCE'd into ttB.

OK, this is where the plan may break down.

The proxy (BufferImpl) instance created in the callee procedure would still be an object scoped (in Java terms) to that method. So, any changes to the 4GL resource state (for lack of a better term) -- things like name, dataSet, etc. -- would all still be instance variables of that local object and NOT of the caller's BufferImpl instance. Sounds like this would be a problem :-(

- The way we manage the BufferFieldImpl look up for case 3 above (dynamic field references to the caller's temp-table) would change slightly. Case 2 would be handled as it is today.

> While bufA is BY-REFERENCE'd in bufB, you can't dereference via :: the fields using bufB's field names - only bufA fields work. So, ttB::fB is not valid, only ttB::fA is valid.

Same restriction for h-ttB:BUFFER-FIELD("fB")?

#93 - 08/01/2019 03:43 PM - Ovidiu Maxiniuc

Marian, what is exactly the difference between passing a structure (I guess DATASETs and BUFFERs are similar here) with BIND and BY-REFERENCE?

From my experience, it feels like the same thing, except that when calling BY-REFERENCE the callee's structure is restored, while in the case of BIND, the association remains persistent until either the (persistent) procedure is deleted or the structure that was passed as parameter is deleted. Am I right?

#94 - 08/01/2019 03:45 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> Because the all calls are 'live' redirected to ttA buffer, the changes done on ttB will simultaneously be seen in ttA and on any other ttC where ttA might have been passed a BIND/BY-REFERENCE parameter.

Yes, but in the proposal I've outlined so far, only changes to the underlying RecordBuffer instance and its currentRecord. As noted above, changes to the callee's BufferImpl instance state would not. We would have to proxy **way** more to cover all that state.

#95 - 08/01/2019 03:51 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Ovidiu Maxiniuc wrote:
>
> > Because the all calls are 'live' redirected to ttA buffer, the changes done on ttB will simultaneously be seen in ttA and on any other ttC where ttA might have been passed a BIND/BY-REFERENCE parameter.
>
> Yes, but in the proposal I've outlined so far, only changes to the underlying RecordBuffer instance and its currentRecord. As noted above, changes to the callee's BufferImpl instance state would not. We would have to proxy **way** more to cover all that state.

This is the way I understood it. The '*way* more' would cover your plan from breaking down, because the new proxy will not contain any actual private data (well, except for ttA reference).

#96 - 08/01/2019 03:53 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Same restriction for h-ttB:BUFFER-FIELD("fB")?

Yes, this throws error 7351. However, h-ttB:BUFFER-FIELD("fA") is valid and returns the same handle as h-ttA:BUFFER-FIELD("fA").

#97 - 08/01/2019 04:08 PM - Eric Faulhaber

I suppose we could create a proxy for all the interfaces which BufferImpl statically implements (including those of its ancestor classes) and have a separate invocation handler which simply redirects all those calls to a different (the caller's) BufferImpl instance. The rest of the plan would stay the same. DMO getter/setter calls would still be redirected using the RecordBuffer$Handler.

To do this without reworking the GCD proxy implementation, I guess we would need two levels of invocation handlers. The first would redirect all interface methods implemented by BufferImpl and its ancestors to the caller's BufferImpl object, and all DMO interface calls to the usual (RecordBuffer$Handler) invocation handler. Yuck.

This is becoming very meta...

#98 - 08/01/2019 04:14 PM - Eric Faulhaber

Constantin, Ovidiu, do either of you know of any "4GL resource state" that is managed in BufferImpl which does NOT go through a well-defined interface? If so, we would need to make sure any calls which can change that state go through an interface, so we can proxy that interface and delegate to the caller's BufferImpl instance. Anything perhaps that is computed internally and would affect the implementation of a converted method or attribute?

#99 - 08/01/2019 04:22 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Constantin, Ovidiu, do either of you know of any "4GL resource state" that is managed in BufferImpl which does NOT go through a well-defined interface?

You mean like dataSet member? There is no ABL method/attribute to set it, but the setDataSet() is called when a DataSet is defined / created.

#100 - 08/01/2019 04:30 PM - Constantin Asofiei

Eric Faulhaber wrote:

> Constantin, Ovidiu, do either of you know of any "4GL resource state" that is managed in BufferImpl which does NOT go through a well-defined

> interface?

See at least BufferImpl.lastQuery, which is executed by QueryWrapper.addBuffer and sets the BufferImpl.lastQuery field; there may be more.

#101 - 08/01/2019 04:51 PM - Constantin Asofiei

Eric Faulhaber wrote:

> Sorry, what precisely are you asking about? The remapped getter/setter calls in the invocation handler, or something else? Can you provide an example of your concern in terms of temp-table definitions?

Say you have these two tables:

```
def temp-table tt2 field fx as int.
def temp-table tt2 field f2 as int.
```

and these convert as these:

```
Tt2_2_1.Buf tt2 = TemporaryBuffer.define(Tt2_2_1.Buf.class, "tt2", "tt2", false);
Tt2_3_1.Buf tt2 = TemporaryBuffer.define(Tt2_3_1.Buf.class, "tt2", "tt2", false);
```

Note how the field type is different - Tt2_2_1.Buf vs Tt2_3_1.Buf, even if their definitions are compatible and can be passed as BY-REFERENCE to eachtoerh; the idea is, when the switch occurs at program B, whatever value TemporaryBuffer.associate returns must be of the same type as the field defining the bufB in program B. Otherwise, the assignment will fail.

Also, how do you plan to restore the Java field's value (e.g. bufB field in program B), back to its prior value, when the internal procedure ends? I think it can be done via a lambda, emitted as an argument to the associate call. OTOH, reflection can be used to restore the field, but its kind of ugly (as to identify the Java field my current approach is to walk the entire class hierarchy and find a declared field having the exact reference).

#102 - 08/01/2019 04:54 PM - Constantin Asofiei

Eric, I'm backtracking a little on your "TemporaryBuffer.associate will return a new proxy idea". The reason are the DATASETs - when you pass a DATASET via BY-REFERENCE, you must switch all the buffers in the target dataset with the buffers in the source dataset. If we don't make these mutable, then I need to use reflection to update the Java fields with the new values (again, same type compatibility constraint).

#103 - 08/01/2019 05:18 PM - Eric Faulhaber

Constantin Asofiei wrote:

> Eric Faulhaber wrote:
>
> > Sorry, what precisely are you asking about? The remapped getter/setter calls in the invocation handler, or something else? Can you provide an example of your concern in terms of temp-table definitions?
>
> Say you have these two tables:
> [...]
> and these convert as these:
> [...]
> Note how the field type is different - Tt2_2_1.Buf vs Tt2_3_1.Buf, even if their definitions are compatible and can be passed as BY-REFERENCE to eachtoerh; the idea is, when the switch occurs at program B, whatever value TemporaryBuffer.associate returns must be of the same type as the field defining the bufB in program B. Otherwise, the assignment will fail.

Yes, it will return a proxy of type bufB. That requirement was the reason behind the changes in the proposal between #3816-80 and #3016-83, based on your earlier feedback.

> Also, how do you plan to restore the Java field's value (e.g. bufB field in program B), back to its prior value, when the internal procedure ends? I think it can be done via a lambda, emitted as an argument to the associate call. OTOH, reflection can be used to restore the field, but its kind of ugly (as to identify the Java field my current approach is to walk the entire class hierarchy and find a declared field having the exact reference).

bufB is a parameter scoped to the called procedure, correct? Doesn't it naturally go out of scope when the procedure returns?

#104 - 08/01/2019 05:22 PM - Eric Faulhaber

Constantin Asofiei wrote:

> Eric, I'm backtracking a little on your "TemporaryBuffer.associate will return a new proxy idea". The reason are the DATASETs - when you pass a DATASET via BY-REFERENCE, you must switch all the buffers in the target dataset with the buffers in the source dataset. If we don't make these mutable, then I need to use reflection to update the Java fields with the new values (again, same type compatibility constraint).

Hm, I don't know enough about the current DATASET implementation to comment or adjust my proposed design. Can you point me at your current implementation so I can get a better idea of what needs to happen?

#105 - 08/01/2019 05:24 PM - Eric Faulhaber

Constantin Asofiei wrote:

> If we don't make these mutable...

What exactly would be mutable? The current buffer proxy? Some new wrapper class? Please help me understand how you envision this.

#106 - 08/01/2019 05:25 PM - Constantin Asofiei

Eric Faulhaber wrote:

> bufB is a parameter scoped to the called procedure, correct? Doesn't it naturally go out of scope when the procedure returns?

No, this is not correct. In this code:

```
def temp-table ttB field fB as int. // here

procedure proc0.
   def input parameter table for ttB.

   create ttB. // here
   ttB.fB = 10. // here
   buffer ttB::fA = 20. // here
end.
```

The def parameter table for ttB doesn't hide the ttB table. What I've marked above uses the same Java field in the converted code.

Also, the signature looks like public void proc0(final TableParameter) - and the parameter is just used to map the source table with the target table. But at proc0, this still uses the Java field we emit for the default buffer.

So, what you noted in #3816-92:

> The proxy (BufferImpl) instance created in the callee procedure would still be an object scoped (in Java terms) to that method

can't work - the real Java field for this default buffer must be updated. Why we need this - because procB can call other methods/procedures in program B, and work with bufB, while this actually references bufA.

#107 - 08/01/2019 05:34 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Hm, I don't know enough about the current DATASET implementation to comment or adjust my proposed design. Can you point me at your current implementation so I can get a better idea of what needs to happen?

A DataSet keeps reference to its buffers. As another is passed as BY-REFERENCE or BIND parameter, the buffers will be assigned at the same time with the DATASET. This does not happens yet because I did not know how this is implemented for BUFFERS. But the idea is that the original BUFFERS and any internal state of the DATASET are hidden while the association is active.

Constantin, do you see any issue here, once the Buffer's association is completed?

#108 - 08/01/2019 05:42 PM - Constantin Asofiei

Eric Faulhaber wrote:

> What exactly would be mutable? The current buffer proxy? Some new wrapper class? Please help me understand how you envision this.

You can save the original ttB (either temp-table and buffer) in a handle var; after you switch ttB to ttA, the handle remains set to the ttB resource. This tells us that the 'hidden' temp-table ttB still exists, it just got switched for the duration of executing that procedure.

The current code which creates our DMO proxy is this:

```
   protected static <T> T createProxy(Class<T> dmoBufIface,
                                      RecordBuffer buffer,
                                      String legacyName)
   {
      Class<?>[] interfaces =
      {
         dmoBufIface,
         BufferReference.class
      };

      T dmoProxy = (T) ProxyFactory.getProxy(BufferImpl.class,
                                             interfaces,
                                             false,
                                             doNotProxy,
                                             new DmoProxyPlugin(dmoBufIface, buffer.dmoClass),
                                             buffer.handler);

      buffer.setDMOProxy((BufferReference) dmoProxy);
      ((BufferImpl) dmoProxy).doSetName(legacyName);

      ProcedureManager.processResource((WrappedResource) dmoProxy, false);

      return dmoProxy;
   }
```

We need to wrap this into another proxy of type T (so you have access to the original, conversion-time temp-table fields), which will be mutable, and

allow you to switch the wrapped DMO proxy to another one. This proxy will know:

- the original, conversion-time DMO proxy (bufB)
- the currently active DMO proxy (bufA)

#109 - 08/01/2019 05:58 PM - Ovidiu Maxiniuc

Something similar to this I added for StaticDataSet. There is the bound Dataset which can be null for REFERENCE-ONLY objects. However, a copy to the original DataSet is kept in dsDef When the object is associated, the bound is reassigned, so now all DataSet interface methods are redirected there. When the BIND/BY-REFERENCE is over, the bound is restored to dsDef for normal DataSet s and null for REFERENCE-ONLY objects.

#110 - 08/01/2019 06:02 PM - Constantin Asofiei

Ovidiu, do you know if we currently have a registry of explicitly defined buffers for a table? I ask because 4GL switches the temp-table to ttA in all defined buffers (explicit or implicit) for ttB, in program B. So we need to process all these, not just the buffer mentioned at def parameter table for.

#111 - 08/01/2019 06:16 PM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> Ovidiu, do you know if we currently have a registry of explicitly defined buffers for a table? I ask because 4GL switches the temp-table to ttA in all defined buffers (explicit or implicit) for ttB, in program B.

No, we do not have, yet. I also encountered the need of such repository, but I suspended work as no occurrences were found yet in customer's code.

> So we need to process all these, not just the buffer mentioned at def parameter table for.

This does not sound too good. Do you mean buffer defined before or after the association?

I tried to run some tests and my procedure stopped because of indexes names were different :-O. (Before defining the new buffer everything was fine.)

#112 - 08/01/2019 06:50 PM - Constantin Asofiei

Ovidiu, I think the private-data issue I've mentioned is because, even if you have something like def input parameter for table btt1., where btt1 is an explicit buffer, the 4GL associates the default-buffer at the source with the default-buffer at the target table (plus switching the temp-tables themselves), regardless of what buffer name you use at the parameter definition.

This explains why I see the private-data set at the default buffer, at the target.

#113 - 08/02/2019 05:18 PM - Constantin Asofiei

There's another part related to the field mapping between the source and target temp-tables: when they are used in queries. I don't have a solution for this yet. Anyway, I expect to see abends if this is in use.

#114 - 08/02/2019 05:20 PM - Eric Faulhaber

Constantin Asofiei wrote:

> There's another part related to the field mapping between the source and target temp-tables: when they are used in queries. I don't have a solution for this yet. Anyway, I expect to see abends if this is in use.

Do you mean dynamically converted queries? Or statically converted queries, too?

#115 - 08/02/2019 05:20 PM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> There's another part related to the field mapping between the source and target temp-tables: when they are used in queries. I don't have a solution for this yet. Anyway, I expect to see abends if this is in use.

We talked about calling an update notification on parent dataset once a buffer was "redirected". Could we do the same for queries... ?

#116 - 08/02/2019 05:26 PM - Constantin Asofiei

Eric Faulhaber wrote:

> Constantin Asofiei wrote:
>
> > There's another part related to the field mapping between the source and target temp-tables: when they are used in queries. I don't have a solution for this yet. Anyway, I expect to see abends if this is in use.
>
> Do you mean dynamically converted queries? Or statically converted queries, too?

For the statically converter queries, this is needed for sure. I don't know about the dynamic queries. I would expect this to use the runtime buffer, and not the compile-time buffer.

#117 - 08/02/2019 05:26 PM - Constantin Asofiei

Ovidiu Maxiniuc wrote:

> We talked about calling an update notification on parent dataset once a buffer was "redirected". Could we do the same for queries... ?

There's nothing to notify as the static queries don't exist when the buffer association is performed.

#118 - 08/02/2019 08:15 PM - Constantin Asofiei

The main changes related to making the Buffer mutable are working, same with the Dataset. But there is some cleanup to do:

- this 'mutable proxy' must be kept only at the Java field, with the 'original definition' for the dataset/buffer
- any internal storage (like dataset and relation buffers) must keep the actual buffer resource, non-mutable
- any lookup of these Buffer/Dataset Java objects must be done keeping in mind if the reference is the mutable proxy or not. Otherwise, even if the semantic of the two objects is the same (as the proxy relays the call to the same resource), they are actually different Java proxy instances, so equals/hashCode will not match.

#119 - 08/02/2019 08:25 PM - Constantin Asofiei

Constantin Asofiei wrote:

> - this 'mutable proxy' must be kept only at the Java field, with the 'original definition' for the dataset/buffer

In other words, I need to determine where the mutable proxy is allowed to be saved and where not. For example, if you have an active QUERY resource for tt1, you invoke something and 'circle back' to the same program where you have this query open (plus having a by-reference argument and tt1 was switched to tt2), what does the QUERY say? I need to check, but I would assume that as the QUERY is already opened/iterating, then it remains on the original buffer, and not the new one received as by-reference.

#120 - 08/05/2019 08:33 AM - Stanislav Lomany

- File param.png added

I'll drop here some starting spec on TABLE parameters.

- When a temp-table is defined as REFERENCE-ONLY, it cannot be used until it is initialized by a RUN call with a source table passed with BY-REFERENCE or BIND.

```
Attempt to reference uninitialized temp-table. (12378)
```

is thrown if we try to query temp-table or its buffers or access their methods/attributes.

- INPUT/OUTPUT specifications are ignored on calling and called sides when a TABLE parameter is passed BY-REFERENCE. Or if it is called by BIND and the table in the calling procedure is REFERENCE-ONLY. If it is called by BIND and the table in the called procedure is REFERENCE-ONLY, then only INPUT or INPUT-OUTPUT parameters are allowed (no actual difference between them). Otherwise the following error is thrown:

```
<external procedure name> INPUT or INPUT-OUPUT is required with BIND if caller parameter is bound and call
ed parameter <parameter name> is REFERENCE-ONLY and not bound yet. (13010)
```

- This table shows results of calling TABLE parameters with different options and different types of tables (normal/REFERENCE-ONLY):

Calling / Callee	By value	BIND	REF-ONLY, by value	REF-ONLY, BIND
By value	By value	By value	Error 13012	Error 13012
BIND	Error 13009	Error 13009	Error 13009	Binding
BY-REF	By ref	By ref	By ref	By ref
REF-ONLY, by value	Error 13015	Error 13015	Error 13015	Error 13015
REF-ONLY, BIND	Error 13161	Binding	Error 13161	Error 12378
REF-ONLY, BY-REF	Error 13013	Error 13013	Error 13013	Error 13013

All errors from the table:

```
<internal procedure name> <external procedure name> BIND or BY-REFERENCE modifier required on RUN, FUNCTION or
 METHOD invocation parameter when called TABLE or DATASET parameter <parameter name> is REFERENCE-ONLY and is
still not bound. (13012)
```

```
<internal procedure name> <external procedure name> BIND modifier not allowed for cases where neigher the call
ed nor the called TABLE or DATASET parameter <parameter name> has been defined REFERENCE-ONLY and neither call
er nor called parameter is a TABLE-HANDLE or DATASET-HANDLE. (13009)
```

```
<external procedure name> REFERENCE-ONLY caller TABLE or DATASET parameter <parameter name> requires BIND keyw
ord. (13015)
```

```
<external procedure name> REFERENCE-ONLY caller TABLE or DATASET parameter <parameter name> requires BIND keyw
ord instead of BY-REFERENCE. (13013)
```

```
<external procedure name> If a caller or called REFERENCE-ONLY parameter <parameter name> will become BOUND to
 its opposite as a result of the call, then the BIND keyword must be supplied on both the caller invocation an
d on the called parameter definition. (13161)
```

```
Attempt to reference uninitialized temp-table. (12378)
```

- There is a nice Progress article describing passing a temp-table parameter by binding: https://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/gsabl/passing-a-temp-table-parameter-by-binding.html

#121 - 08/05/2019 09:38 AM - Constantin Asofiei

There's another issue with BIND which I don't see how to solve yet: if the source and target tables have different names (and thus different DMO variables and maybe field names), but otherwise they are compatible, after BIND is done, a certain table will exist in two ways:

- with the original name, where it was defined originally.
- with the destination name (and field names), where it was bound. BTW, can you bind a table/dataset to multiple locations?

So, this table must know that it is used from within a 'wrapper' where other DMO variable and field names exist. We need some kind of notification for it to switch variable/field names, when another program is entered.

#122 - 08/07/2019 08:01 AM - Marian Edu

Constantin Asofiei wrote:

> There's another issue with BIND which I don't see how to solve yet: if the source and target tables have different names (and thus different DMO variables and maybe field names), but otherwise they are compatible, after BIND is done, a certain table will exist in two ways:
>
> - with the original name, where it was defined originally.
> - with the destination name (and field names), where it was bound. BTW, can you bind a table/dataset to multiple locations?
>
> So, this table must know that it is used from within a 'wrapper' where other DMO variable and field names exist. We need some kind of notification for it to switch variable/field names, when another program is entered.

Constantin, the way I see it each table has a meta-data (structure) which is sometimes more or less important when passing it around as parameter. The way Progress handle this most of the time is it will only look for the number of fields and data type/extents of each if that matches the names are not that important. But, if you pass the table by-value the callee will receive a table instance with the same meta-data as the one it expects even if the one in the caller has different names. If you pass it by-reference/bind it will get the same meta-data as in the caller. There is a catch here though, accessing the table's fields statically works even if the name in the caller is different - looks like the compiler is using the field position resolved at compile time instead of the field name. Trying to access the same field at runtime (dynamically) will result in an error (7351) because as said before the table's meta-data is the one from the caller so the field is not found because it has a different name.

Now, I don't think this is even documented and so far I haven't seen anyone (ab)using tables like this - in the usual scenario the table/dataset definition is in an include file so everyone that is using it have the exact same definition (the only thing that is different is that sometime the table is reference-only).

If you want to be completely compatible with Progress implementation I would say the only option would be to resolve the field name/position mapping at compile time for static data access and then at runtime use the table's meta-data/structure. For meta-data when you pass the table by value it needs to be a new instance of the caller table but the meta-data should point to the one of the callee. If you pass it by reference/bind it's really the same table instance that you need to pass around, it's just like when you pass the table handle - the callee will work on the same instance.

#123 - 08/13/2019 03:06 AM - Constantin Asofiei

The changes with the mutable proxy implementation are in 4124a rev 11449. Please review.

#124 - 08/26/2019 01:30 PM - Constantin Asofiei

There's a new issue to consider; if the temp-table has a before-table, when passing this table (or its dataset) around as a parameter, if it is by-reference, the before-table must always be 'glued' to its temp-table, and bound together; not just the temp-table.

I don't know what is happening with the by-value case.

#125 - 08/27/2019 07:12 AM - Marian Edu

Constantin, the before-table as well as the default buffer handle (and the before-table's default buffer handle) are all 'part' of the temp-table so it travels with it when passed as parameter. For BY-REFERENCE there is just the pointer that gets passed so the celled routine works on the exact same instance of the temp-table while when BY-VALUE is used the called routine receives a (deep) copy of the instance from the caller - it's a new temp-table instance, all records from the caller temp-table are copied over including the before table content if any. So, to answer your question, for BY-VALUE the temp-table received by the called routine will do have a before-table if the one in the caller had one.

#126 - 08/27/2019 08:26 AM - Constantin Asofiei

Marian Edu wrote:

> Constantin, the before-table as well as the default buffer handle (and the before-table's default buffer handle) are all 'part' of the temp-table

An addition here: not only the default-buffer, any explicit buffer defined (non-dynamic) gets to reference the temp-table passed as argument, in case of BY-REFERENCE (or REFERENCE-ONLY).

And I also found another case: the binding can be 'inverse', from the callee to the caller. I posted this to #3809, but it belongs here:

Found another issue: bind can be bi-directional, from the destination to source, if the source is BY-REFERENCE. I'm working on fixing it. The effects of this:

- a static dataset (and its associated temp-tables) can survive the defining external program if it was bound at the caller's source; I assume the delete relies on NUM-REFERENCES, even for static datasets, and it will prevent deleting it, even if the external program gets deleted, if this is greater than zero.
- I assume when a by-reference dataset gets unbound (because it was switched to another dataset or if its defining external program gets deleted), it will automatically see that the currently bound dataset has an unknown instantiating procedure, NUM-REFERENCES is zero, and it will delete it

#127 - 08/27/2019 09:00 AM - Marian Edu

BIND can be used to bind a reference only table in the caller only if the called routine has an OUTPUT parameter that also use BIND. In that case the

table in the caller will be bound to the one of the called procedure and that will survive the deletion of the procedure/class that instantiated it in the first place (if static) although that is not visible when walking the session buffers (first-buffer/next-sibling).

A table/dataset can't be unbound, if you try to bind it again it will only work if the instance you try to bind it to is the same as the one it's already bound to otherwise it will throw an error stating it's not reference-only anymore (was bound). The only exception is if that gets bound to a dynamic instance and that was deleted, then you can bind it again to another instance - if you try to access any of it's attributes while left unbound it will throw the same error as if it was reference-only.

Most probably Progress deletes those 'orphan' static datasets when NUM-REFERENCES reaches zero but there is no way to test this, as said those are not really visible through the session buffers list.

#128 - 08/28/2019 01:53 AM - Marian Edu

Constantin Asofiei wrote:

> An addition here: not only the default-buffer, any explicit buffer defined (non-dynamic) gets to reference the temp-table passed as argument, in case of BY-REFERENCE (or REFERENCE-ONLY).

The difference is that those explicitly defined buffers are not part of the temp-table instance, they just point to it - aka their 'table' property point to the temp-table instance. A reference-only temp-table can not be used until it's bound to something real so it's more like a placeholder that at some point could reference a real thing. You can define additional buffers to that reference-only temp-table, can write code that access it or any defined buffers on it in a static way but at runtime if the thing is not bound to a real thing it will throw error.

No idea how this is implemented in Progress nor FWD but the way I see it is a reference-only temp-table is a proxy, you know it's definition for static use but at runtime unless it has a valid temp-table instance set it will throw error - otherwise it will simply proxy everything to that bound instance. The twist here is that when the code access that reference-only table (or any defined buffers on it) it's own definition is used - aka, the fields names must match the one of the reference only table definition. If you want to work with it dynamically and use the default-buffer-handle or create a new dynamic buffer on it this will give you the definition of the table instance bound to it. The field names can be different as long as the number of fields and position, data type and extent for each of those match.

So, for reference-only tables I would use a different implementation that is like a proxy for whatever your temp-table implementation looks like. For the buffer that can point to either a database table or a temp-table, all it needs to work with is some kind of 'data buffer' - for a reference-only table the HANDLE property will return the same as the one of the bound temp-table so it will all be transparent for the buffer that is defined on a reference-only temp-table. Accessing the static buffer will try to access the HANDLE property which will throw if the table is not bound or it will point to a valid instance otherwise so you don't really need to 'switch' anything when the table is bind.

#129 - 11/14/2019 04:47 PM - Greg Shah

- Assignee deleted (Stanislav Lomany)

#130 - 11/27/2019 09:59 AM - Greg Shah

- Related to Bug #4424: Additional def var needed to receive table handle when keyword FOR included in def parameter added

#131 - 01/23/2020 11:59 AM - Greg Shah

- Related to Feature #4516: testing and completion of runtime compatibility for table/table-handle/dataset/dataset-handle parameter options added

#132 - 01/23/2020 12:23 PM - Greg Shah

- Status changed from WIP to Closed

- % Done changed from 0 to 100

The final work will be done in #4516.

Files

conv.log	118 KB	06/28/2019	Stanislav Lomany
test.p	71 Bytes	07/10/2019	Stanislav Lomany
test2.p	71 Bytes	07/10/2019	Stanislav Lomany
SvlTest2.java	698 Bytes	07/10/2019	Greg Shah
SvlTest.java	781 Bytes	07/10/2019	Greg Shah
tests.log	10.3 KB	07/15/2019	Stanislav Lomany
param.png	26.5 KB	08/05/2019	Stanislav Lomany