

Database - Bug #5030

xfer dataset test issues

12/07/2020 04:00 PM - Ovidiu Maxiniuc

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

History

#1 - 12/07/2020 04:23 PM - Ovidiu Maxiniuc

I have started recently to run the testsets stored on xref repository in order to fine-tune the dataset-related attributes, methods and events. The first iteration of my implementation of these was done using results from executing different pieces of code on a OE 11.6 test machine. There were some issues which were incorrectly interpreted from the OE online manual and my tests and the xfer test repository was a great source of fixes, improvements and increase of FWD stability in regard to dataset features.

However, running them I encountered a set of inconsistencies: in fact, the xfer tests were failing on my OE test machine. The only reasonable cause for this was that they were developed on different OE version, presumably newer than my 11.6. Starting from that assumption I went on adjusting the code biasing the xfer testset. At this moment the attributes and events test have no failures for all static, dynamic and mixed cases. Yet, probably there are some fine and hidden differences that make me harder to advance with methods testsets. I am unable to test these locally because my OE lacks some of functionality (for example: EMPTY-TEMP-TABLE for a TEMP-TABLE handle object which makes the expected dataset configurations impossible to reach at the moment a specific method is tested). Since the internal dataset is different at the moment of a method call the result/after-call state will be evidently wrong.

The role of this task is for me to acquire more insights of issues I encounter and behaviour of dataset-related methods/attributes which are not supported in OE11.6 and little or not documented in (online) reference/manual.

#2 - 12/07/2020 04:40 PM - Ovidiu Maxiniuc

A first question is about the dataset relations. Normally, there can be at most $n - 1$ instances of them, but if some of them are marked as not-active (and here is a funny thing, that the parameter for add-relation is called not-active but the documentation specifies that using *"FALSE causes the data-relation to be inactive"*) more can be defined. The buffers that are not designed child of an active relation are named top-buffers.

However, the REPOSITION has a very similar behaviour, breaking the recursivity of FILL method if the attribute is set for a relation and also making child buffers of the relation to become top-buffers even if the relation is active.

My problem is, I do not have a clear understanding of how is this attribute (REPOSITION) set for a parent-id-relation. Why the reposition attribute of a parent-id-relation is yes (usually), but sometimes no? Does the other attribute (ACTIVE) have some role? Does it matter whether the relation, dataset or buffers are created statically or dynamically?

#3 - 12/07/2020 05:21 PM - Ovidiu Maxiniuc

A second issue is related to recently added AUTO-DELETE attribute. I am see why is an auto-delete dynamic buffer not deleted when its dataset calls clear() even if the attribute was not reset?

#4 - 12/08/2020 02:17 AM - Marian Edu

Ovidiu Maxiniuc wrote:

A first question is about the dataset relations. Normally, there can be at most $n - 1$ instances of them, but if some of them are marked as not-active (and here is a funny thing, that the parameter for add-relation is called not-active but the documentation specifies that using *"FALSE causes the data-relation to be inactive"*) more can be defined. The buffers that are not designed child of an active relation are named top-buffers.

Indeed an unfortunate wording in the doc but other than that is correct than a child buffer of the inactive relation is 'promoted' as a top level buffer - the test on active property of data relation should have made that clear but it looks like this was overlooked :(

However, the REPOSITION has a very similar behaviour, breaking the recursivity of FILL method if the attribute is set for a relation and also making child buffers of the relation to become top-buffers even if the relation is active.

The behavior on FILL is somehow similar, the buffers do not show in the top buffers list though and the reposition during parent navigation still happens only different - 'SELECTION', this actually means the query for child buffer does not filter out records to only include the ones for the selected parent but will include all records and only reposition itself on the corresponding record. If you will the normal REPOSITION mode (duh, REPOSITION) is used for parent-child (1-n) relations while SELECTION is the inverse child-parent (n-1) ones.

My problem is, I do not have a clear understanding of how is this attribute (REPOSITION) set for a parent-id-relation. Why the reposition attribute of a parent-id-relation is yes (usually), but sometimes no? Does the other attribute (ACTIVE) have some role? Does it matter whether the relation, dataset or buffers are created statically or dynamically?

It does not matter if is static or dynamic, the default value of REPOSITION is probably because is more natural to define a relation as parent child. But say if you want to show all orders with the items sold on each of them then it might make sense to define the orders as top buffer and attach that to the browse and define the relation to items as a reverse one (child-parent) and set the reposition to be SELECTION (false). That way on fill the relation is ignored and all items will be loaded regardless or not being sold in one of the orders, after the fill when the navigation occurs in the order query the query on the child (well parent) table is not reopen filtering out unrelated records as it will happen for REPOSITION but simply it will be repositioned on the corresponding item sold on the current order.

I'll look if there is already an example for this and craft out something if not.

#5 - 12/08/2020 02:35 AM - Marian Edu

Ovidiu Maxiniuc wrote:

A second issue is related to recently added AUTO-DELETE attribute. I am see why is an auto-delete dynamic buffer not deleted when its dataset calls clear() even if the attribute was not reset?

The AUTO-DELETE attribute only has a role when a dynamic dataset is deleted, nothing to do with CLEAR method that merely resets it's structure - the object is still very much alive and valid. In fact there is a note on the documentation for CLEAR method that states that calling this on a dynamic dataset will leave all it's 'inner' objects hanging around as orphans which could lead to memory leaks is those are not deleted individually. The default value (true) means when a dataset is deleted (DELETE OBJECT hDataset) then all it's components are deleted as well. If one wants (some of) those components to outlive the dataset then the AUTO-DELETE can be set to false and those objects are not deleted. One use case will be to temporary create a dataset for existing buffers then when not needed leave the initials objects alone when we delete the dataset. The only use case for the CLEAR method imho is when one want to re-arrange the buffers in different dataset structure since one buffer can't be part of different datasets at the same time.

#6 - 12/08/2020 07:37 PM - Ovidiu Maxiniuc

Here is one example I used for observing the REPOSITION attribute:

```
define temp-table tt1
  field f1 as integer.

define temp-table tt2
  field f2 as integer
  field tt1_id as recid.

define variable hds as handle.
create dataset hds.

message "set-buffers" hds:set-buffers(buffer tt1:handle, buffer tt2:handle).
message "add-parent-id-relation 1:" hds:add-parent-id-relation(buffer tt1:handle, buffer tt2:handle).
message "relation" hds:NUM-RELATIONS
  hds:get-relation(1)
  hds:get-relation(1):name
  hds:get-relation(1):active
  hds:get-relation(1):reposition.

message "add-parent-id-relation 2:" hds:add-parent-id-relation(buffer tt1:handle, buffer tt2:handle, "").
message "relation" hds:NUM-RELATIONS
  hds:get-relation(2)
  hds:get-relation(2):name
  hds:get-relation(2):active
  hds:get-relation(2):reposition.

message hds:num-top-buffers.
```

This will output:

```
set-buffers yes
add-parent-id-relation 1: 34466
relation 1 34466 RELATION1 yes yes
```

```
add-parent-id-relation 2: 34467
relation 2 34467 RELATION2 yes yes
2
```

Meaning that, two ACTIVE PARENT-ID-relations are created for same two TOP-buffers.

However, in `xfer:dataset/dynamic/methods/add_parent_id_relation.p` a very similar dataset is constructed, with only buffers and a single relation. Yet, the code expects the relation to have the reposition attribute set to false:

```
43:     relHdl = dsOrder:add-parent-id-relation(ttOrder:default-buffer-handle, ttOrderLine:default-buffer-handle) no-error.
...
56:     assert-true(relHdl:reposition = false, substitute({&msg_attr_test_value}, 'reposition', false), {&line-number}).
```

#7 - 12/08/2020 07:40 PM - Ovidiu Maxiniuc

Marian Edu wrote:

The AUTO-DELETE attribute only has a role when a dynamic dataset is deleted, nothing to do with CLEAR method ..

Sorry, that was my mistake. I assumed the CLEAR method is called before destroying the dataset object. Apparently this does not happen. I fixed the code and now the testcase passes.

#8 - 12/09/2020 03:48 AM - Marian Edu

Ovidiu Maxiniuc wrote:

Here is one example I used for observing the REPOSITION attribute:
[...]

This will output:
[...]

Meaning that, two ACTIVE PARENT-ID-relations are created for same two TOP-buffers.

Well, nothing is perfect... in practice no one is doing that though so I wouldn't make it such a big deal :)

However, in `xfer:dataset/dynamic/methods/add_parent_id_relation.p` a very similar dataset is constructed, with only buffers and a single relation. Yet, the code expects the relation to have the reposition attribute set to false:

[...]

Yes, this is correct - we do not have an older version (you mentioned 11.6) but on both 11.7 and 12.2 the reposition attribute is set to false as it should since that is the default. This is a very special kind of relationship, with the slow deprecation of RECID this is probably hardly used in real life anyway. Because the RECID of temp-table records are really just valid in memory one might say having REPOSITION set to true make sense as this can't affect the FILL behavior but then when navigating parent query one expects to only see filtered records in child query so SELECTION not REPOSITION. Can you check what is the relation's query PREPARE-STRING to see if child records are indeed filtered out using parent's RECID?

```
hds:get-relation(2):query:prepare-string
```

Imho a parent-id-relationship is to be ignored on FILL anyway, as said there is no way to have those relations persisted in a database for later retrieval - the RECID of temp-table records has nothing to do with that of a database table record. Only used for in-memory data structures where the developer is lazy enough not to add a real foreign-key and use this RECID based relations.

#9 - 12/09/2020 06:09 PM - Ovidiu Maxiniuc

Marian Edu wrote:

Well, nothing is perfect... in practice no one is doing that though so I wouldn't make it such a big deal :)

The problem is, even so, if this is the way it works in OE, FWD must do the same.

Yes, this is correct - we do not have an older version (you mentioned 11.6) but on both 11.7 and 12.2 the reposition attribute is set to false as it should since that is the default.

Sorry, I lost you here. The xref code expects the REPOSITION to be FALSE at line 56: `assert-true(relHdl:reposition = false,` Could you test my first example from note-6 on OE 11.7 and/or 12.2? I do not have access to these versions.

BTW, is there a public documentation on OE 12.X? I could not find it, I am always redirected to 11.7.

This is a very special kind of relationship, with the slow deprecation of RECID this is probably hardly used in real life anyway. Because the RECID of temp-table records are really just valid in memory one might say having REPOSITION set to true make sense as this can't affect the FILL behavior but then when navigating parent query one expects to only see filtered records in child query so SELECTION not REPOSITION. Can you check what is the relation's query PREPARE-STRING to see if child records are indeed filtered out using parent's RECID?

Yes, it is. On my test machine, the query gives me: `FOR EACH tt2 WHERE tt2.tt1_id=RECID(tt1)` for both relations.

BTW, what is SELECTION ? You mentioned it earlier, but I could not find any documentation about such attribute. Is it new in 12.X?

Imho a parent-id-relationship is to be ignored on FILL anyway, as said there is no way to have those relations persisted in a database for later retrieval - the RECID of temp-table records has nothing to do with that of a database table record. Only used for in-memory data structures where the developer is lazy enough not to add a real foreign-key and use this RECID based relations.

Yes, I noticed this recently. The PARENT-ID relations are not used for FILL-ing, even if active. Really, I am not sure what is their use.

Greg,

I am a bit puzzled now. What is our OE version target to match? Will we support multiple versions? I know there was a conversion/runtime flag for then-new features in OE10 some times ago, they are obsolete now, anyway.

#10 - 12/09/2020 06:39 PM - Ovidiu Maxiniuc

- Status changed from New to WIP

I have another issue. In dataset/dynamic/methods/copy_dataset.p we have:

```
25: run dataset/dynamic/create/dsMaster.p(false, false, false, output dsMaster).
202: run dataset/dynamic/create/dsOrder.p(true, false, false, output dsOrder).
```

Note the first parameter, noDefaults is different. This means the ttOrderCustomer table will have a some additional fields and indexes (compared to dsMaster 's ttCustomer) hence a different structure, as seen in dataset/dynamic/create/ttCustomer.p:

```
if noDefaults then
do:
    ttCustomer:add-new-field ("active", "logical").
    ttCustomer:add-new-field ("turnover", "decimal").
    ttCustomer:add-new-field ("employees", "int64").
    ttCustomer:add-new-field ("established", "date").
    ttCustomer:add-new-field ("created", "datetime").
    ttCustomer:add-new-field ("updated", "datetime-tz").
    ttCustomer:add-new-field ("logo", "blob").
    ttCustomer:add-new-field ("description", "clob").
    ttCustomer:add-new-field ("customHandle", "handle").
    ttCustomer:add-new-field ("customObject", "Progress.Lang.Object").

    ttCustomer:add-new-index("pk", true, true).
    ttCustomer:add-index-field("pk", "customerNum").
end.
```

At line 205 in same copy_dataset.p we have:

```
dsOrder:copy-dataset ( dsMaster, ?, true ).
```

Note that there is no 4th parameter (loose-copy-mode), so its value is false. Here is what docs say about this:

When FALSE, the metaschema for the source and target temp-tables must be the same or the AVM generates a run-time error. The default value is FALSE.

So, the tables have different schema because of different noDefaults, loose-copy-mode is false, yet the calls ends with success. No error is generated and, as the test follows on next lines, the ttOrderCustomer in destination dataset should be populated with 3 records.

My test machine refuses to run the testcase. It complains about error 11885 (lack of a unique primary index is required in the target table) and quits silently.

Ovidiu Maxiniuc wrote:

Marian Edu wrote:

Well, nothing is perfect... in practice no one is doing that though so I wouldn't make it such a big deal :)

The problem is, even so, if this is the way it works in OE, FWD must do the same.

Agreed, just saying finding unexpected behavior should be a surprise... it simply shows no one did that before to report it or it was not seen as critical, in this case the easiest way to heal the pain is stop doing whatever was causing it :)

Funny enough you can even set a different recid field for second relation and it won't complain - not sure how this will work afterwards, have to extend the tests for this case :(

Yes, this is correct - we do not have an older version (you mentioned 11.6) but on both 11.7 and 12.2 the reposition attribute is set to false as it should since that is the default.

Sorry, I lost you here. The xref code expects the REPOSITION to be FALSE at line 56: `assert-true(relHdl.reposition = false,` Could you test my first example from note-6 on OE 11.7 and/or 12.2? I do not have access to these versions.

Your example works on new versions but REPOSITION is false.

BTW, is there a public documentation on OE 12.X? I could not find it, I am always redirected to 11.7.

[\[\[https://documentation.progress.com/output/ua/OpenEdge_latest\]\]](https://documentation.progress.com/output/ua/OpenEdge_latest)

This is a very special kind of relationship, with the slow deprecation of RECID this is probably hardly used in real life anyway. Because the RECID of temp-table records are really just valid in memory one might say having REPOSITION set to true make sense as this can't affect the FILL behavior but then when navigating parent query one expects to only see filtered records in child query so SELECTION not REPOSITION. Can you check what is the relation's query PREPARE-STRING to see if child records are indeed filtered out using parent's RECID?

Yes, it is. On my test machine, the query gives me: `FOR EACH tt2 WHERE tt2.tt1_id=RECID(tt1)` for both relations.

BTW, what is SELECTION ? You mentioned it earlier, but I could not find any documentation about such attribute. Is it new in 12.X?

Is not an attribute, if REPOSITION is set to false then it's said to be SELECTION - the doc is very brief about this, there is however some explanation here [\[\[https://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/dvpds/using-the-reposition-data-relation.html\]\]](https://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/dvpds/using-the-reposition-data-relation.html).

Imho a parent-id-relationship is to be ignored on FILL anyway, as said there is no way to have those relations persisted in a database for later retrieval - the RECID of temp-table records has nothing to do with that of a database table record. Only used for in-memory data structures where the developer is lazy enough not to add a real foreign-key and use this RECID based relations.

Yes, I noticed this recently. The PARENT-ID relations are not used for FILL-ing, even if active. Really, I am not sure what is their use.

As said, the only used case for those (that I could think of) is for in-memory parent-child data structures where the developer doesn't bother to define a proper foreign-key and for 'NESTED' relations when loading XML/JSON data where there isn't a natural primary key in parent that is present in child

table.

Greg,

I am a bit puzzled now. What is our OE version target to match? Will we support multiple versions? I know there was a conversion/runtime flag for then-new features in OE10 some times ago, they are obsolete now, anyway.

#12 - 12/10/2020 08:53 AM - Greg Shah

I am a bit puzzled now. What is our OE version target to match? Will we support multiple versions? I know there was a conversion/runtime flag for then-new features in OE10 some times ago, they are obsolete now, anyway.

None of our current customers use 11.6. And as far as I know 11.7 was "alive" longer than 11.6. I think there will be many customers on that and some on 12.x. I also understand that these newer versions are fixing bugs in older versions, so by focusing on those newer versions we avoid the "implement the bug for compatibility" requirement. If we ever get a customer that requires compatibility with 11.6 or who has an application which is dependent on bugs/quirks of 11.6, then we will implement those at that time.

#13 - 12/10/2020 12:16 PM - Ovidiu Maxiniuc

Marian Edu wrote:

BTW, is there a public documentation on OE 12.X? I could not find it, I am always redirected to 11.7.

[[https://documentation.progress.com/output/ua/OpenEdge_latest]]

Yes, that's it. I think you might see a different page. At the bottom of the page I can read:

© 2017 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.
Progress® OpenEdge® Release 11.7

Yes, I noticed this recently. The PARENT-ID relations are not used for FILL-ing, even if active. Really, I am not sure what is their use.

As said, the only used case for those (that I could think of) is for in-memory parent-child data structures where the developer doesn't bother to define a proper foreign-key and for 'NESTED' relations when loading XML/JSON data where there isn't a natural primary key in parent that is present in child table.

This means I will have to double-check the JSON/XML serialization implementation.

#14 - 12/10/2020 12:57 PM - Marian Edu

Ovidiu Maxiniuc wrote:

[[https://documentation.progress.com/output/ua/OpenEdge_latest]]

Yes, that's it. I think you might see a different page. At the bottom of the page I can read:

Not really, just not paying attention I guess... docs for all version (pdf) can be found here [[<https://docs.progress.com/category/openedge-archives>]]. The latest seems to be 11.7 indeed, time to wake them up I guess :)

#15 - 01/05/2021 12:33 PM - Ovidiu Maxiniuc

I have identified a OE bug that bothers me for some time. The copy-temp-table's 2 parameter (replace-mode) is leaking over the 3rd (loose-copy-mode). This makes incompatible tables defined like in code below

```
DEFINE TEMP-TABLE tt1
  FIELD f1 AS INTEGER
  INDEX idx1 IS UNIQUE f1.
```

```
DEFINE TEMP-TABLE tt2
  FIELD f1 AS INTEGER
  FIELD f2 AS CHARACTER
  INDEX idx1 IS UNIQUE f1.
```

```
CREATE tt1. tt1.f1 = 1.
CREATE tt1. tt1.f1 = 2.
```

message

```
TEMP-TABLE tt1:COPY-TEMP-TABLE(TEMP-TABLE tt2:HANDLE)
TEMP-TABLE tt2:COPY-TEMP-TABLE(TEMP-TABLE tt1:HANDLE)
```

```
TEMP-TABLE tt1:COPY-TEMP-TABLE(TEMP-TABLE tt2:HANDLE, ?, NO)
TEMP-TABLE tt2:COPY-TEMP-TABLE(TEMP-TABLE tt1:HANDLE, ?, NO, YES)
```

```
TEMP-TABLE tt1:COPY-TEMP-TABLE(TEMP-TABLE tt2:HANDLE, ?, YES, ?)
TEMP-TABLE tt2:COPY-TEMP-TABLE(TEMP-TABLE tt1:HANDLE, ?, YES, NO).
```

to be able to be copied.

To output

```
COPY-TEMP-TABLE requires tt2 and tt1 columns to match exactly unless the fourth loose-mode parameter is passed
as TRUE. (12303)
COPY-TEMP-TABLE requires tt1 and tt2 columns to match exactly unless the fourth loose-mode parameter is passed
as TRUE. (12303)
COPY-TEMP-TABLE requires tt2 and tt1 columns to match exactly unless the fourth loose-mode parameter is passed
as TRUE. (12303)
```

no no no yes yes yes

Notice that, if the 2nd parameter is on, it will allow the copy operation, even if loose-copy-mode (3rd) parameter is unknown/? (implicitly) or NO explicitly.

#16 - 01/05/2021 07:29 PM - Ovidiu Maxiniuc

According to my old and new ABL reference manual and OpenEdge [online documentation](#), the TEMP-TABLE-PREPARE() method has a 2nd, logical, optional parameter, before-table-exp, which would *allows [...] to explicitly generate a before-table for a dynamic temp-table*.

Well, there is no such parameter. The truth is there is none. The before table is always created. The attempt to use this signature will throw an error. The problem is not that parameter per-se, but the fact that its value is NO by default. FWD was not creating the before-table and since these tables are not directly visible, this caused several apparently unrelated regressions with xfer tests :(.

#17 - 01/07/2021 05:41 PM - Ovidiu Maxiniuc

Marian,
I was re-running the test-sets again when I encountered the following issue: the same method responds in apparently different ways in different places. I am talking about dataset version of merge-changes method when it receives as parameter a different dataset handle than the one it got the changes using get-changes. In all cases, the changes were obtained from dsMaster dataset, be it static reference or a handle to a dynamic one.

directory	source file	line	method calls for getting and merging the changes and expected outcome for the latest
static	merge_changes_changeDs_applyOnDiffrentDs.p	28	dataset dsCopyMaster:handle:get-changes (dataset dsMaster:handle). dataset dsCopyMaster:handle:merge-changes (dataset dsTestMaster:handle) no-error. assert-err-num(11923...
static	merge_changes_pk_changeDs_applyOnDiffrentDs.p	28	dataset dsCopyMaster:handle:get-changes (dataset dsMaster:handle). dataset dsCopyMaster:handle:merge-changes (dataset dsTestMaster:handle) no-error. assert-err-num(11923...
dynamic	merge_changes_changeDs_applyOnDiffrentDs.p	54	dsCopyMaster:get-changes (dsMaster). dsCopyMaster:merge-changes (dsTestMaster) no-error .assert-not-err(...
dynamic	merge_changes_pk_changeDs_applyOnDiffrentDs.	54	dsCopyMaster:get-changes (dsMaster). dsCopyMaster:merge-changes (dsTestMaster) no-error .assert-not-err(...

Aside the fact that the 11923 error (MERGE-CHANGES run on original table <name> that does not match the ORIGIN-HANDLE in the GET-CHANGES table <name>.) was expected only when working with static dataset reference and none when working with handles to dynamically constructed datasets I cannot see any syntactic difference. FWD tests now the original table which provided the changes on get-changes. If they differ the error is raised. Evidently, this is not always the case. I refuse to think this based on static/dynamic attribute and implement it that way.

Do you have any guidance here?

#18 - 01/11/2021 02:27 AM - Marian Edu

Ovidiu Maxiniuc wrote:

According to my old and new ABL reference manual and OpenEdge [online documentation](#), the TEMP-TABLE-PREPARE() method has a 2nd, logical, optional parameter, before-table-exp, which would *allows [...] to explicitly generate a before-table for a dynamic temp-table*.

Well, there is no such parameter. The truth is there is none. The before table is always created. The attempt to use this signature will throw an error. The problem is not that parameter per-se, but the fact that its value is NO by default. FWD was not creating the before-table and since these tables are not directly visible, this caused several apparently unrelated regressions with xfer tests :(.

Ovidiu, you're right on this one. The parameter is there alright, it does very much exist and works as expected see my new test on the method in `table/dynamic/methods/temp_table_prepare.p`.

I will go over the get/merge changes tests, I assume what you're saying is that those tests might not work because of missing before-table (at least for dynamic).

#19 - 01/14/2021 04:50 PM - Ovidiu Maxiniuc

Marian Edu wrote:

Ovidiu Maxiniuc wrote:

Ovidiu, you're right on this one. The parameter is there alright, it does very much exist and works as expected see my new test on the method in `table/dynamic/methods/temp_table_prepare.p`.

Based on your integrated testcase I tried to run the following simplified version on both OE 11.6.3 and OE 11.7.5 I have access to:

```
define variable ttCust as handle no-undo.

create temp-table ttCust no-error.
ttCust:add-new-field ("customerNum", "integer") no-error.
message "1 param:" ttCust:temp-table-prepare ("ttCustomer") ttCust:before-table.
delete object ttCust.

create temp-table ttCust no-error.
ttCust:add-new-field ("customerNum", "integer") no-error.
message "2 param, false:" ttCust:temp-table-prepare ("ttCustomer", false) no-error.
delete object ttCust.

create temp-table ttCust no-error.
ttCust:add-new-field ("customerNum", "integer") no-error.
message "2 param, true:" ttCust:temp-table-prepare ("ttCustomer", true) no-error.
```

Each time I get the following **compiler messages**:

```
** Unable to understand after -- "ttCustomer, false)". (247)
** Could not understand line 10. (198)
```

```
** Unable to understand after -- "ttCustomer, true)". (247)
** Could not understand line 15. (198)
```

There is one more thing: up to xfer revno 968, the only temp-table-prepare method used was the one with a single parameter. If the default value for the second parameter is false, it means the before tables were never created. At least for dynamic tests. But they were created otherwise a lot of testcases would work incorrectly.

So I am guessing that:

- the second parameter **was added in OE 12.X** (even if the documentation expressly mentions it from 11.5) and
- its **default value is true**, not false as documented.

#20 - 01/15/2021 02:57 AM - Marian Edu

Ovidiu Maxiniuc wrote:

Each time I get the following **compiler messages**:

Ovidiu, this is probably because of the `message xxx no-error.` statement that you use there which I doubt is valid syntax. Just take out the message or the no-error part and it should work just fine (11.7 at least).

There is one more thing: up to xfer revno 968, the only temp-table-prepare method used was the one with a single parameter. If the default value for the second parameter is false, it means the before tables were never created. At least for dynamic tests. But they were created otherwise a lot of testcases would work incorrectly.

That was something I was worried about as well, it turns out this only applies for current state and the AVM is smart enough to create the before table automatically when needed - as in when you enable tracking-changes for the temp-table the before table become valid and everything works.

```
define variable ttCust    as handle no-undo.
define variable dsCust    as handle no-undo.

create temp-table ttCust no-error.
ttCust:add-new-field ("customerNum", "integer") no-error.
message ttCust:temp-table-prepare ("ttCustomer") valid-handle(ttCust:before-table).

create dataset dsCust.
dsCust:add-buffer(ttCust).

ttCust:tracking-changes = true.
message valid-handle(ttCust:before-table).
```

So I am guessing that:

- the second parameter **was added in OE 12.X** (even if the documentation expressly mentions it from 11.5) and

I'm sure it was added before, just try to fix the syntax error as mentioned before and it should work.

- its **default value is true**, not false as documented.

Nope, it's value is definitively false but be aware of the fact that the before table (for dynamic temp-table) gets automatically created as needed so it's more like an optimization for when this is not needed, sort of lazy-loading. I will try to update the tests to see exactly what methods/properties can trigger the automatic before-table creation.

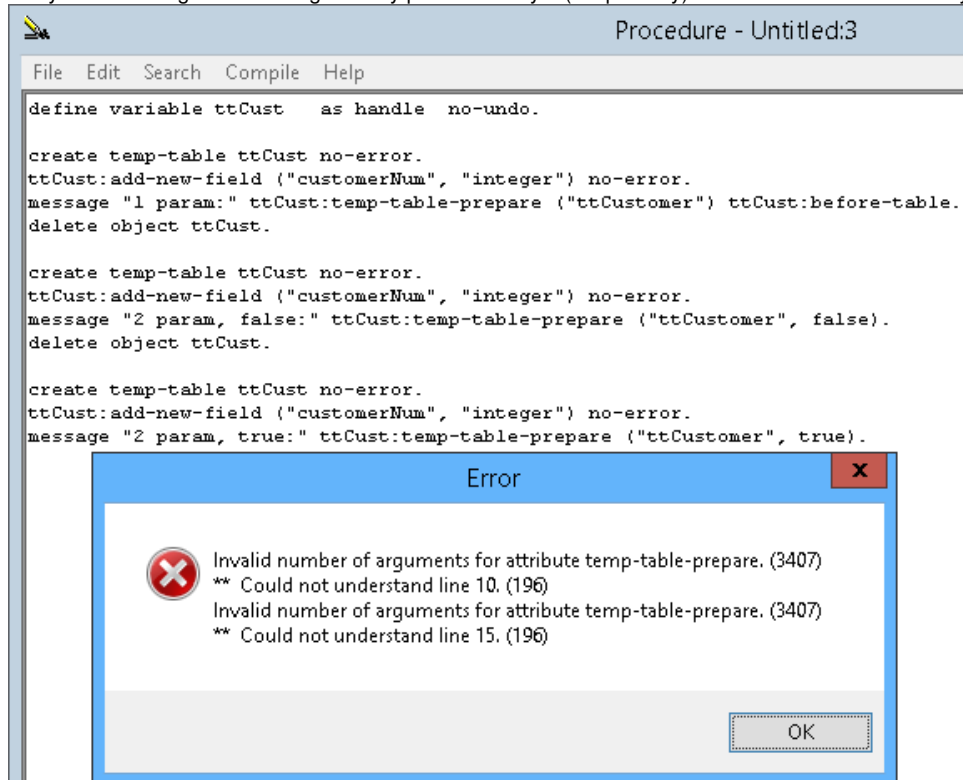
#21 - 01/15/2021 11:59 AM - Ovidiu Maxiniuc

- File Error3407x2.png added

Marian Edu wrote:

Ovidiu, this is probably because of the `message xxx no-error.` statement that you use there which I doubt is valid syntax. Just take out the message or the no-error part and it should work just fine (11.7 at least).

Sorry for the wrong error messages in my previous entry. I (desperately) tried to force it work. With my local 11.6 I get the following:



LE: I tested the code on OE 11.7.5 and, indeed, the parameter is present.

Nope, it's value is definitively false but be aware of the fact that the before table (for dynamic temp-table) gets automatically created as needed so it's more like an optimization for when this is not needed, sort of lazy-loading. I will try to update the tests to see exactly what methods/properties can trigger the automatic before-table creation.

It make sense what you say about this lazy before-table creation.
If you can detect the moment when these are created, that would be really useful.

Files

Error3407x2.png	13.5 KB	01/15/2021	Ovidiu Maxiniuc
-----------------	---------	------------	-----------------