Database - Feature #3574

finish implementation of temp-table XML support

05/17/2018 08:38 AM - Greg Shah

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

Description

Related issues:

Related to Database - Feature #3294: implement temp-table XML support from #3...	**Closed**
Related to Database - Feature #3809: ProDataSet support	**Closed**
Related to Database - Bug #5176: Buffer field attribute leak	**Closed**

History

#1 - 05/17/2018 08:39 AM - Greg Shah

From Eric:

The following is the full list of (non-ProDataSet) features that are currently unimplemented for temp-table XML support.

This does not take into account whatever changes Constantin has made in 3507a.

Methods:

read-xmlschema() method
- file | memptr | handle | longchar
- override-default-mapping (string --> {character | clob}; binary --> {raw | blob})
- field-type-mapping (override field/type pairs)
- verify-schema-mode (loose --> match name, type | strict --> match name, type, extend)

write-xmlschema() method
- file | stream | stream-handle | memptr | handle | longchar
- formatted
- encoding (?)
- min-xmlschema
- omit-initial-values

read-xml() method (have basic implementation; the following options/features are not implemented)
- memptr | handle | longchar
- read-mode (append | merge | replace)
- schema-location (absolute or relative path or URL (file or http protocol))
- override-default-mapping (string --> {character | clob}; binary --> {raw | blob}) -- only when temp-table schema not predefined?
- field-type-mapping (override field/type pairs)
- verify-schema-mode (ignore | loose | strict) -- same as read-xmlschema() method, except for ignore

write-xml() method (have basic implementation; the following options/features are not implemented)
- stream | stream-handle | memptr | handle | longchar
- formatted
- encoding (?)
- schema-location
- write-xmlschema
- min-xmlschema
- omit-initial-values

serialize-row() method?
- looks like a lot of overlap with write-xml() internals

Attributes:

namespace-prefix
namespace-uri
schema-marshal?

#2 - 05/17/2018 08:39 AM - Greg Shah

- Related to Feature #3294: implement temp-table XML support from #3257-34 added

#3 - 11/19/2019 09:24 AM - Greg Shah

How much of this task is complete as of branch 3809e?

#4 - 11/19/2019 10:35 AM - Ovidiu Maxiniuc

Methods:

- read-xmlschema() method
  - file | memptr | handle | longchar OM: only file and longchar
  - override-default-mapping (string --> {character | clob}; binary --> {raw | blob}) OM: estimated difficulty: probably easy
  - field-type-mapping (override field/type pairs) OM: estimated difficulty: probably easy
  - verify-schema-mode (loose --> match name, type | strict --> match name, type, extend) OM: partial, need testing

- write-xmlschema() method
  - file | stream | stream-handle | memptr | handle | longchar OM: only file and longchar
  - formatted OM: done
  - encoding (?) OM: done
  - min-xmlschema OM: done, but needs more testing
  - omit-initial-values OM: done, but needs more testing

- read-xml() method (have basic implementation; the following options/features are not implemented)
  - memptr | handle | longchar OM: only file and longchar
  - read-mode (empty | append | merge | replace) OM: only *empty* and *append*. *merge* and *replace* probably similar to FILL modes
  - schema-location (absolute or relative path or URL (file or http protocol)) OM: estimated difficulty: medium
  - override-default-mapping (string --> {character | clob}; binary --> {raw | blob}) -- only when temp-table schema not predefined? OM: estimated difficulty: probably easy
  - field-type-mapping (override field/type pairs) OM: estimated difficulty: probably easy
  - verify-schema-mode (ignore | loose | strict) -- same as read-xmlschema() method, except for ignore OM: partial, need testing

- write-xml() method (have basic implementation; the following options/features are not implemented)
  - stream | stream-handle | memptr | handle | longchar OM: only file and longchar
  - formatted OM: done
  - encoding (?) OM: done
  - schema-location OM: done
  - write-xmlschema OM: done
  - min-xmlschema OM: done, but needs more testing
  - omit-initial-values OM: done, but needs more testing

- serialize-row() method?
  - looks like a lot of overlap with write-xml() internals OM: missing both runtime and conversion support

Attributes:

- namespace-prefix OM: missing support in Temp-Table/Buffer
- namespace-uri OM: done, but needs more testing
- schema-marshal? OM: missing both runtime and conversion support

#5 - 01/22/2020 05:44 PM - Greg Shah

- Related to Feature #3809: ProDataSet support added

#6 - 09/10/2020 05:58 AM - Constantin Asofiei

Conversion and runtime stubs for SERIALIZE-ROW() method are in 3821c rev 11512

#7 - 10/14/2020 12:47 PM - Greg Shah

- Assignee set to Ovidiu Maxiniuc

#8 - 11/10/2020 06:49 AM - Greg Shah

- Status changed from New to WIP

- % Done changed from 0 to 90

From Ovidiu:

> Here are the remaining known issues:
>
> - At this moment I am trying to find a solution for the following issue (it looks more like an unnoticed regression from #4011): I am creating a dynamic table based on .xsd information. The DMO interface is built in memory with fields denormalized (default). The indexed access methods exist, but they are not annotated so not present in DmoMeta. The problem is, when reading an extent field from XML the indexed setter must be used. Since there is none, the read will fail. I do not have a solution here yet;
> - After fixing this issue I need some time for testing "schema-location (absolute or relative path or URL (file or http protocol))" (a couple of).

#9 - 11/14/2020 06:19 AM - Greg Shah

- % Done changed from 90 to 100

- Status changed from WIP to Review

From Ovidiu:

> I am now committing a stable revision (r11811) with all issues in #3574 fixed. Of course there might be some edge-cases I could not imagine but this is clearly unavoidable.
>
> However, some things might be slightly changed with 4397a before merging to 3821c (its parent branch) or the trunk because I have touched some areas where Eric (changes in Record class and changed access level for some methods to be accessible for XML importer) or Constantin (changes related to appserver data marshalling) might not fully agree.

#10 - 01/07/2021 02:01 PM - Greg Shah

From #4397-27:

> so I would say they should be handled as part of #3574. However, the separation line is rather thin, which means I might spot issues with dataset

table content while fixing issues in serialization code.

Is there more work to be done to test the serialization code or is it complete (except for the 2 bugs noted in #4397-2)?

#11 - 01/25/2021 04:29 PM - Ovidiu Maxiniuc

I have encountered the following issue: the serialize-hidden attribute steals the value from another temp-table field. In fact, the attribute is now common for each field of all temp-tables with same structure. I noticed this for dynamic temp-tables, but this may also happen in the static case also.

In details, if we construct two temp tables with same structure (schema), the DynamicTablesHelper in its createDynamicDMO will create equals CacheKey containing the tables' meta information. They are equals because the table are equals. So they will share the same DMO (DMOInfo), interface and implementation class. This looks fine as optimization, but there are other issues: these shared classes are used as key for other informations, like DmoMeta in DmoMetadataManager and LegacyTableInfo in TableMapper's data structures. And this causes some problems. The keys are indeed immutable, but their associated values in these maps are. This means that the programmer can change some table and field attributes (ex: serialize-hidden) in different tables independently in 4GL. In FWD this is not possible. Changing an attribute for a table will have the effect that the change is visible in all tables which share the same DMO interface/impl-class.

I am thinking of a solution here...

#12 - 01/25/2021 05:53 PM - Ovidiu Maxiniuc

While thinking of the above issue I encountered the next one. Consider the 4GL code (ignore missing declarations):

```
define variable dsM    as handle    no-undo.
dsM:read-json("longchar", longCharVar, ?) no-error.
run CreateDS-M.p(output dsM).
lOk =  dsM:read-json("file", "dsM.json", 'append') no-error.
```

It is correctly converted to Java. The first call to read-json will evidently fail. Here is the last line:

```
    silent(() -> lOk.assign(dsM.unwrapJsonData().readJson(new SourceData("file", "dsM.json"), new character("
append"))));
```

but at the execution lOk is set to no as the second read-json java call detects that dsM was invalid at the first call (3135:Invalid handle. Not initialized or points to a deleted object). The "dsM.json" is valid. The readJson() method tests the first record after being validated with ErrorManager.error() and for some unknown cause, this was not reset before calling the second read-json method so it returns the 3135 error.

My question is: where/when should the error flag be reset?

#13 - 01/26/2021 08:25 AM - Greg Shah

> My question is: where/when should the error flag be reset?

1. A NO-ERROR statement has its errors collected into the ErrorManager.WorkArea.pendingErrorStatus. At the end of the statement's processing, successful or not, the current ErrorManager.WorkArea.errorStatus is reset from the ErrorManager.WorkArea.pendingErrorStatus and the ErrorManager.WorkArea.pendingErrorStatus is cleared. The ErrorManager.WorkArea.errorStatus is what is read by any 4GL access to the ERROR-STATUS system handle. This means that the previous statement's NO-ERROR results are available until the end of the next NO-ERROR statement.

2. This behavior is modified slightly when one enters a nested top-level block during silent mode. In TransactionManager.pushScope(), the silent flag is remembered and cleared. In the corresponding TransactionManager.popScope(), if the silent flag was true on entry it will be restored. This isn't related to your case since the RUN statement did NOT have NO-ERROR set. If it was set, then the silent mode would be set aside for the duration of the nested call, which would change how ERROR conditions are processed.

#14 - 01/26/2021 09:45 AM - Ovidiu Maxiniuc

Greg Shah wrote:

> 1. A NO-ERROR statement has its errors collected into the ErrorManager.WorkArea.pendingErrorStatus. At the end of the statement's processing, successful or not, the current ErrorManager.WorkArea.errorStatus is reset from the ErrorManager.WorkArea.pendingErrorStatus and the ErrorManager.WorkArea.pendingErrorStatus is cleared. The ErrorManager.WorkArea.errorStatus is what is read by any 4GL access to the ERROR-STATUS system handle. This means that the previous statement's NO-ERROR results are available until the end of the next NO-ERROR statement.

I agree but with a small note: shouldn't error flag be cleared at the start of the next NO-ERROR?

In my case, the second statement is dsM:read-json(...) no-error. Its runtime iteratively reads the fields and populates a record in memory in a batch-assign -like construct. When this is done RecordBuffer.endBatch() is called and the ErrorManager.error() is used to check whether the record was successfully validated. The record is fine, but the ErrorManager still returns the cached error from the first NO-ERROR statement.

I think I will manually reset the error flag at the start of readJson() method. This will fix the issue for this testcase but there might be other places which have the same issue. I an thinking whether we could fix this globally.

#15 - 01/26/2021 10:20 AM - Greg Shah

> shouldn't error flag be cleared at the start of the next NO-ERROR?

No. Although we initially thought this was the case, it causes problems when you are referencing the ERROR-STATUS:ERROR in an expression inside the next statement. The 4GL indeed allows reference to the previous value while the next NO-ERROR statement is processes.

> When this is done RecordBuffer.endBatch() is called and the ErrorManager.error() is used to check whether the record was successfully validated. The record is fine, but the ErrorManager still returns the cached error from the first NO-ERROR statement.

I think you can use isPendingError() instead and it is a simple fix. While processing in the runtime, this is the correct way to check for problems in the current NO-ERROR statement.

#16 - 01/27/2021 08:31 PM - Ovidiu Maxiniuc

I started implementing a solution for the issue in note-11. As noted there, the problem in the fact that TableMapper uses the DMO interface as key for permanent, static and dynamic temp-tables. As two temp-tables may share the same DMO interface, we cannot distinct between the two tables at this level.

My solution is to use the TempTable (or better, the AbstractTempTable) as a key instead. Normally this should be easy to switch the key but the TempTableMapper share the same interface with its sibling, PermanentTableMapper. So we need a common key. Unfortunately, the permanent tabled do not have such concept. So I added a new super interface GenericTable which is inherited by TempTable (and hence implemented by AbstractTempTable) and another new class PermanentTable. The new interface has only one method public DmoMeta getDmoMeta(); pulled up from TempTable. Another useful common properties/attributes might also be extracted, but once having the DmoMeta the user of this object can obtain a lot of data. I see two disadvantages with this approach:

- the duplication of information. While the current solution tries to store common information only once, and being to aggresive so forcing the mutable table properties to be stored only once, the new solution will duplicate the common, immutable table properties. Not very good as this includes the fields, indexes, etc.
- adding the PermanentTable reference to buffers and generally obtaining the reference to the GenericTable object used as key when accessing TableMapper static methods.

But, at the same time, this will unify the permanent and temp-tables. Now there are a lot of places where specific processing is done conditionally in TableMapper: when accessing temp-tables use a specific API, for permanent tables - a different one. From my quick tests I noticed that some of these mutable attributes are valid even for permanent tables. For example you can attempt to set and retrieve the serialize-hidden or XML-NODE-NAME without getting errors for permanent tables, too.

There is another (secondary) solution I considered: storing these attributes in AbstractTempTable. When access is needed, the parent TempTable object is queried.

#17 - 01/28/2021 04:34 PM - Eric Faulhaber

I really don't like the TableMapper construct at all. I tried to rationalize its internals at one point long ago, without changing the API used by other classes, but it's still a mess. And now with all the changes we introduced with #4011 (DmoMeta and RecordMeta, which already are largely redundant with one another), there is so much redundancy in the information it stores. At some point, we need to rationalize all this in a holistic way.

Sorry, that was more of a rant than a constructive comment, but the point w.r.t. your previous post was that I support anything you do to rationalize/simplify this set of classes, and I am opposed to anything you do to make the mess worse ;)

I honestly cannot judge based on your comments which way you have gone, though I agree the key problem had to be solved, since the keys were not one-to-one with the tables. However, the keys sound significantly complex now. Is there an impact on performance? I did not have this in mind when I reviewed the code.

#18 - 01/28/2021 05:11 PM - Ovidiu Maxiniuc

That is exactly what I feel in regard to TableMapper, too. It only have the advantage of being decoupled, but access is awful. In r11991 I cut some static calls to it, so replacing the double or triple map access with direct information available usually locally, in dmoMeta object. (I had some problems with this update from last night: after previous rebase I forgot to restore the binding so the commit was local. I did an update and everything was a mess :(. I re-did the rebase and re-commit earlier so you might want to review these changes too).

#19 - 01/28/2021 07:12 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> I re-did the rebase and re-commit earlier so you might want to review these changes too).

See #4397-46.

#20 - 03/03/2021 07:09 PM - Ovidiu Maxiniuc

- Related to Bug #5176: Buffer field attribute leak added

#21 - 03/17/2021 02:30 PM - Greg Shah

Can we close this?

#22 - 03/17/2021 04:26 PM - Ovidiu Maxiniuc

Yes, please do. I am not aware of any remaining issue.

#23 - 03/17/2021 04:36 PM - Greg Shah

- Status changed from Review to Closed