Database - Feature #4397

add database attrs, methods and options

11/09/2019 10:40 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 #3809: ProDataSet support	**Closed**
Related to Database - Feature #1587: implement full support for word indexes	**Closed**

History

#1 - 11/09/2019 10:56 AM - Greg Shah

The following work is needed:

- BUFFER:LAST-BATCH attribute
  - Although this is related to PDS, it has no conversion/runtime support because it was not in the 3809 list.
  - Taking into account that the BATCH implementation seems fine, we just need to expose the internal state of buffer after the FILL operation. 2-3 hours should be enough.
- BUFFER:DATA-SOURCE-MODIFIED attribute (partial runtime)
  - The getter and manual setter are already implemented.
  - We need to add an automatic set when SAVE-ROW-CHANGES encounters an error.
  - The flag is also documented as marshalled between the client and the AppServer. Ovidiu and need to discuss with Constantin this part of the implementation. Estimation: 2-3 hours.
- BUFFER:MERGE-ROW-CHANGES() method (runtime stub)
  - The runtime has the state validations at the beginning but not actual implementation.
  - Estimation: 1 DoW.
- RELATION:REPOSITION attribute
  - Incomplete business logic implementation.
  - Estimation 1/2 DoW.
- TEMP-TABLE SERIALIZE-NAME option (partial runtime)
  - This attribute is only implemented for STATIC TEMP-TABLEs, which we think is wrong.
  - Ovidiu reports that the code should be moved up the hierarchy so that the dynamic TEMP-TABLE benefit from it.
  - Estimate: 2-3h
- TEMP-TABLE:ADD-NEW-INDEX() method
  - It is currently marked as partial.
  - I believe that the current restriction that it doesn't handle word indexes.
  - This will be dependent upon completing #1587.
- QUERY:SET-CALLBACK() method
  - The BUFFER/DATASET support for this currently exists in trunk.
  - Query support is missing right now.
- record phrase options
  - FIELDS phrase
  - NO-PREFETCH
    - It is not clear if this is needed.
    - Eric notes: "We do our own optimization of prefetching, so NO-PREFETCH probably has no use for us, but I can look at this a bit deeper... put in up to 1 day, but it probably will amount to nothing."
    - I guess we only need to implement something here if there is some real application behavior that can be different with this flag.

#2 - 01/22/2020 05:44 PM - Greg Shah

- Related to Feature #3809: ProDataSet support added

#3 - 04/27/2020 06:39 PM - Constantin Asofiei

Just a note, there is a DATASET:FILL-MODE attribute which I stubbed in 4231b. FWD now has runtime only for BUFFER:FILL-MODE.

#4 - 04/27/2020 06:44 PM - Constantin Asofiei

I recall way back that we had a talk about FIELDS/EXCEPT with a record phrase, and with our current FWD, it doesn't make sense as Hibernate doesn't allow you to load specific fields (or something like that, don't recall the exact reason); the idea was, we always ended up doing the same (or more work) if we handled these options at runtime.

So, unless something changes in the FWD persistence layer with #4011, I don't see how we can benefit from these.

#5 - 04/28/2020 07:01 AM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> Just a note, there is a DATASET:FILL-MODE attribute which I stubbed in 4231b. FWD now has runtime only for BUFFER:FILL-MODE.

I was not aware of it. The API reference mentions only:

> **Applies to:** Buffer object handle

Where did you find from of the existence of the attribute for dataset objects? Customer code maybe? I wonder what is its semantics?

#6 - 04/28/2020 07:32 AM - Constantin Asofiei

Ovidiu Maxiniuc wrote:

> Constantin Asofiei wrote:
>
> > Just a note, there is a DATASET:FILL-MODE attribute which I stubbed in 4231b. FWD now has runtime only for BUFFER:FILL-MODE.
>
> I was not aware of it. The API reference mentions only:
>
> > **Applies to:** Buffer object handle
>
> Where did you find from of the existence of the attribute for dataset objects? Customer code maybe? I wonder what is its semantics?

One of Marian's tests, see dataset/static/events/after_before_no_fill.p

#7 - 04/28/2020 08:12 AM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> > Where did you find from of the existence of the attribute for dataset objects? Customer code maybe? I wonder what is its semantics?
>
> One of Marian's tests, see dataset/static/events/after_before_no_fill.p

I cannot find this test in my repository. I am a bit more nervous because my VM is not working.
Could you please test whether this attribute really exist? Just create a very simple dataset and try to print it with message.

#8 - 04/28/2020 08:16 AM - Constantin Asofiei

Yes, it exists:

```
def temp-table tt1 field f1 as int.
def dataset ds1 for tt1.

dataset ds1:fill-mode = "no-fill".
```

Marian's tests are on xfer: https://proj.goldencode.com/projects/p2j/wiki/Writing_4GL_Testcases#Testcase-Project

#9 - 04/28/2020 09:15 AM - Greg Shah

Constantin Asofiei wrote:

> I recall way back that we had a talk about FIELDS/EXCEPT with a record phrase, and with our current FWD, it doesn't make sense as Hibernate doesn't allow you to load specific fields (or something like that, don't recall the exact reason); the idea was, we always ended up doing the same (or more work) if we handled these options at runtime.
>
> So, unless something changes in the FWD persistence layer with #4011, I don't see how we can benefit from these.

I don't know of any restriction. It was only for optimization purposes and it was going to require work to figure out, so it was deferred (see #2137) and no one ever got back to it.

We know:

- Many 4GL tables are very wide (they have MANY columns). Perhaps this is because they didn't have referential integrity and it became easier to just include everything in one table rather than factor things properly.
- The 4GL developers explicitly put these FIELDS clauses in for a performance reason since there is no functional reason.

I think it is likely that honoring this will be helpful to us.

#10 - 04/28/2020 09:24 AM - Constantin Asofiei

I understand the reasoning behind it, to limit the amount of data retrieved by a query; but, my point is that to make this work in FWD:

- Hibernate operations (flush, load, etc) are done on a DMO - AFAIK we can't tell hibernate to flush only a partial record. At some point, the entire record will need to be in memory. And I don't think the runtime changes are that trivial.
- I don't know what the approach would be in FQL and #4011

With the above, yes, I think I can add both FIELDS and EXCEPT (as they work together, no reason to leave the other behind). But the runtime is another beast to handle.

#11 - 04/28/2020 09:41 AM - Greg Shah

The runtime would be up to Eric/Ovidiu. After #4011, they have plenty of control all the way down to the JDBC driver.

#12 - 04/28/2020 09:43 AM - Constantin Asofiei

Looks like FIELDS and EXCEPT are already supported for this case:

```
for each tt1 fields (f1 f2 f3):
   message tt1.f1.
end.

for each tt1 except (f4 f5 f6):
   message tt1.f1.
end.
```

which generates:

```
            query0.include(tt1, "f1", "f2", "f3");
```

and

```
            query1.exclude(tt1, "f4", "f5", "f6");
```

There's a todo for static temp-table usage, I'll fix that. But otherwise, conversion works.

#13 - 04/28/2020 10:08 AM - Constantin Asofiei

4231b rev 11493 :

- added BUFFER:LAST-BUFFER and QUERY's callback API support
- fixed EXCEPT/FIELDS option with a record phrase for a static temp-table/buffer.

#14 - 04/28/2020 12:16 PM - Ovidiu Maxiniuc

Greg Shah wrote:

> The runtime would be up to Eric/Ovidiu. After #4011, they have plenty of control all the way down to the JDBC driver.

Indeed, with 4011 we are able to generate customized queries, to fetch only a subset of the fields. For the moment, we compose queries which get the full record from db. Also, we still use full records as entities in cache and these options might get it a bit more complicated (meaning, when cache is queried it will respond: *I don't have the whole record, only 63.8% of it from a previous query*).
So the 4GL programmer must be really careful using them, otherwise the same record will need multiple SQL accesses to be fetched.

#15 - 06/20/2020 11:57 AM - Greg Shah

Task branch 4231b has been merged to trunk as revision 11347.

Is there anything left to do in this task?

#16 - 10/14/2020 12:45 PM - Greg Shah

- Assignee set to Ovidiu Maxiniuc

#17 - 10/30/2020 11:53 AM - Ovidiu Maxiniuc

Status of issues in 4397a/11776.

- BUFFER:LAST-BATCH attribute

Done 99%. There is one edge-case where a "look-ahead" query is required.

- BUFFER:DATA-SOURCE-MODIFIED attribute (partial runtime)

Done.
Note: the attribute/function name suggests that the data source buffer was altered but, from my tests, changing the source buffers does not affect this attribute. Having an modified or deleted before-image loaded in before-buffer(this) will set the attribute, instead.

- BUFFER:MERGE-ROW-CHANGES() method (runtime stub)

Done.

- RELATION:REPOSITION attribute

Done.

- TEMP-TABLE SERIALIZE-NAME option (partial runtime)

Done.

- TEMP-TABLE:ADD-NEW-INDEX() method

Not touched. (It is currently marked as partial because it doesn't handle word indexes. This will be dependent upon completing #1587).

- QUERY:SET-CALLBACK() method

Done.

- FIELDS / EXCEPT phrase

Not implemented.
Supported in conversion but the runtime silently ignores the configured fields. The implementation would need an additional array/bitset member in Record class to flag the working fields, and all accesses to the record to check this flag. Of course, at least Loader, Persister and FqlToSqlConverter classes must be adjusted to take into consideration the missing properties. This will invalidate the fast-find algorithm at least, and it might make possible to validate records if the missing properties are part of indexes. So, the risk are high and we gain some benefits only when big data is not fetched. If this is mandatory, a new dedicated task should be open.

- NO-PREFETCH

Not implemented.
Eric is right. We incrementally fetch the results from database starting with one record bucket. Implicitly, in FWD all queries are NO-PREFETCH if not really iterated.

- DATASET:FILL-MODE

I tested this with OE 11.6.3. In ABL you can assign it any value, but reading it will always return ?. In FWD the conversion honours the current ABL documentation (FILL-MODE on progress.com) and unwraps the handle to a buffer object, which result in error 4052 to be printed. Since the error is not present in ABL I will add the NO-OP implementation to dataset objects, too.

#18 - 10/30/2020 01:05 PM - Greg Shah

TEMP-TABLE:ADD-NEW-INDEX method

Not touched. (It is currently marked as partial because it doesn't handle word indexes. This will be dependent upon completing #1587).

Please make a note in #1587 so that this feature is implemented in that task.

FIELDS / EXCEPT phrase

Not implemented.
Supported in conversion but the runtime silently ignores the configured fields. The implementation would need an additional array/bitset member in Record class to flag the working fields, and all accesses to the record to check this flag. Of course, at least Loader, Persister and FqlToSqlConverter classes must be adjusted to take into consideration the missing properties. This will invalidate the fast-find algorithm at least, and it might make possible to validate records if the missing properties are part of indexes. So, the risk are high and we gain some benefits only when big data is not fetched. If this is mandatory, a new dedicated task should be open.

As far as we know, this would only be needed for performance reasons. If in fact there is a difference in validation behavior, then we would need this for functional reasons as well.

We had added conversion support in #2095. We have #2137 for the runtime support. I don't object to working this in that task, BUT I'd like to know if this is purely performance related or if it has any functional impact. Please see if you can find any difference in behavior when this clause is present.

Please do update all gap marking to match up. Leave behind any notes in comments. I'm OK with things like NO-PREFETCH being unimplemented so long as the gap marking is correct. By correct, I mean we need it to be encoded so it is clear that there is no gap there. We probably can mark it as full restricted with a comment that says that explains it.

#19 - 10/30/2020 01:14 PM - Ovidiu Maxiniuc

- Related to Feature #1587: implement full support for word indexes added

#20 - 11/10/2020 06:47 AM - Greg Shah

- % Done changed from 0 to 100

- Status changed from New to Test

#21 - 11/20/2020 10:49 AM - Eric Faulhaber

Ovidiu, based on the previous discussion in this task's history, my understanding is that the work is functionally complete. However, there are many files without header updates; some code has been commented out, but not removed; and a number of TODOs have been added. What is the actual status of the branch? It cannot be merged as is.

#22 - 11/20/2020 12:23 PM - Eric Faulhaber

Some more comments/questions on 4397a:

- Are there any performance implications to the PropertyDefinition changes?
- Are there any changes to TemporaryBuffer which might conflict with Adrian's ongoing work in #4055?
- Why was the default scale for decimal fields changed from 10 to 0 in Property?
- There are debug changes to a number of files (e.g., TemporaryBuffer, Window, rowid, etc.) which look transient.

#23 - 11/20/2020 03:59 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Ovidiu, based on the previous discussion in this task's history, my understanding is that the work is functionally complete. However, there are many files without header updates; some code has been commented out, but not removed; and a number of TODOs have been added. What is the actual status of the branch? It cannot be merged as is.

I reviewed all affected files in 4397a and:

- added missing H entries. The was the most encountered issue, but I let these intentionally. This is because I planned (and did) several rebase operation and this is the place where the most conflicts occur during operation. As this branch will be merged soon, the number of rebase operation will be reduced;
- reverted back some files which were only touched. They were bulk-saved to bzr.
- reverted back changes related to temporary code used in debugging and profiling. Although I will still need these changes, I will manually exclude them from further commits;
- there is only one known TODO left behind (related to FILL-WHERE-STRING). The attribute works in vast majority of cases, except for some edge-cases when the 4GL programmer will issue a series of changes in DataSource and linked DataRelation state. I need to think more to fix this optimally.

> Some more comments/questions on 4397a:
>
> - Are there any performance implications to the PropertyDefinition changes?

The changes are not meant to gain some performance directly (like those from #4785) but to implement the SCHEMA-MARSHAL attribute. So the idea is to skip externalizing some attributes based on schemaMarshalLevel value. Ultimately, the amount of data transfer can be decreased at the will of 4GL programmer which is what this attribute's intent.

> - Are there any changes to TemporaryBuffer which might conflict with Adrian's ongoing work in #4055?

I just rebased to latest 3821c. As expected, no conflict occurred. Adrian's working on extent fields - as far as I can say, my changes are related to BEFORE-BUFFER related fields, which were not always correctly saved.

> - Why was the default scale for decimal fields changed from 10 to 0 in Property?

In some places the knowledge that the decimals were not specified is needed, as opposed to its actual value. This is the same issue Stanislav encountered in #4930, but here the type allows a special token (0) to be used as "not specified".

- There are debug change to a number of files (e.g., TemporaryBuffer, Window, rowid, etc.) which look transient.

As noted above, I rolled back these changes.

#24 - 11/30/2020 04:15 PM - Ovidiu Maxiniuc

I encountered the following issue with conversion. It is not mandatory related to datasets but I encountered it while working for this task and decided to post is here.
Consider the following code fragment:

```
DEFINE VARIABLE numTest AS INTEGER NO-UNDO.
DEFINE VARIABLE ds1 AS HANDLE NO-UNDO.

FUNCTION getNumTest RETURNS INTEGER () :
    numTest = numTest + 1.
    RETURN numTest.
END FUNCTION.

CREATE DATASET ds1.
ds1:NUM-BUFFERS = ?.
ds2:NUM-BUFFERS = getNumTest().
```

Note that NUM-BUFFERS is a read-only of DataSet.

The conversion, as we expect it now, will be for the last three lines:

```
    ds1.unwrap().readOnlyError("NUM-BUFFERS");
    ds2.unwrap().readOnlyError("NUM-BUFFERS");
    message(getNumTest());
```

The output in FWD is:

```
**NUM-BUFFERS is not a setable attribute for DATASET  widget. (4052)
**NUM-BUFFERS is not a setable attribute for DATASET  widget. (4052)
1
```

There are not less than two issues here:

1. (a bit more visible) the side-effect of the first call to getNumTest() is lost. That is, the replacement with readOnlyError() of the call to getNumTest() function will cause the numTest variable not to be incremented, so the output will be 1 in FWD instead of 2 which is on P4GL.
2. a more hidden issue is that, before checking whether this attribute (possible all of them) is "setable", P4GL will test if the value of the argument is unknown and whether this value is valid for the attribute. In which case, another exception (4083) is thrown.

The result of running this piece of code on P4GL is instead:

```
**Unable to assign UNKNOWN value to attribute NUM-BUFFERS on DATASET widget. (4083)
**NUM-BUFFERS is not a setable attribute for DATASET widget. (4052)
2
```

To fix this I am going to change a bit the conversion so that the readOnlyError() method to accept a second parameter, do not drop the parameter AST subtree and actually evaluate that expression at runtime. This way the side effect will be kept. After we have the value of the argument it can be checked whether it is indeed, ? / unknown. There is a problem here: I do not know which attributes allow ? as valid argument. We probably need a hardcoded list of them but, OTOH, the attributes might be widget-dependent. Anyway, for the moment I will put all of them in same handling and add an exclusion list, initially empty.

#25 - 12/07/2020 07:42 PM - Ovidiu Maxiniuc

I encountered two more issues which were probably added with 4011 and slipped unnoticed until now.

- the first is that, for dynamic temp-tables, the names of components of any index were always the index's name: DynamicConversionHelper.buildTempTable() (at *build index components*). The legacyField is (String) idxCol.getAnnotation("historical"); not child annotation. What is strange here is that this is a CTRL+C/V bug old as the dynamic conversion support in FWD. I wonder how we never ever encountered an issue because of this. I would miss this too, if there was not...

- the second issue, caused by an optimization which went wrong. That is, in DynamicConversionHelper.prepareTempTable(), after calling buildDataModelClass the model is cached in RecordBuffer. Which is nice, since the model, as the schema AST below, are not needed to be rebuilt again as the table structure will never change. However, since we are using the BaseAST super class to obtain the tree linking, once any of these trees was grafted to a schema (data model, respectively) parent node, it will keep the reference to its former parent (BaseAST.pareent) and immediate sibling (BaseAST.right). As result, if these subtrees are not "unbound", they will keep these reference when re-grafted to another (temporary) schema, meaning an old "immediate sibling" will leak to a new schema/data model tree. In my case the next sibling was also added to the structure, making it a recursive link to second table. This caused a situation when the tree navigation went endlessly causing 100% CPU usage and, what was worse, refusing the debugger to connect to server process.
  The solution here is simple, I think: just unbind the cached schema/model subtree once the processing is done. Or better, if the buffer is used in a second dynamic query, make sure any links to an old structure are gone before grafting to the new parent trees.

#26 - 01/07/2021 11:06 AM - Greg Shah

Is it correct that all of the functionality needed for this task is complete in 4397a?

When do you estimate completion for the effort to fix dataset issues with the testcases project?

#27 - 01/07/2021 11:46 AM - Ovidiu Maxiniuc

Greg Shah wrote:

> Is it correct that all of the functionality needed for this task is complete in 4397a?

> When do you estimate completion for the effort to fix dataset issues with the testcases project?

At this moment all of the functionality specified in the first notes of this task is present in branch 4397a (latest revision is 11924, built on 3821c/11897). When running dataset related tests (dynamic/mixed/static attributes/events/methods) from xfer there are only two known issues:

- a quirk in add-relation(). If same relation is added twice, the method return false, but the relation is actually added. It result in a kind of illegal state of the dataset. This is apparently a known issue, documented in tests code. The code to handle it wouldn't be trivial (all existing relations need to be scanned to detect the duplicate) and I am not sure the 'feature' is worthy of the extra CPU clocks. I decided not to implement it yet;
- a known issue discovered relatively recent (more details in #5056) which causes records with default values to be validated even if they are break the unique constraint of an index.

The remaining failing issues are related to XML/JSON external serialization, 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. But overall, I think this task is over now.

#28 - 01/07/2021 02:00 PM - Greg Shah

> At this moment all of the functionality specified in the first notes of this task is present in branch 4397a (latest revision is 11924, built on 3821c/11897). When running dataset related tests (dynamic/mixed/static attributes/events/methods) from xfer there are only two known issues:
>
> - a quirk in add-relation(). If same relation is added twice, the method return false, but the relation is actually added. It result in a kind of illegal state of the dataset. This is apparently a known issue, documented in tests code. The code to handle it wouldn't be trivial (all existing relations need to be scanned to detect the duplicate) and I am not sure the 'feature' is worthy of the extra CPU clocks. I decided not to implement it yet;

My initial reaction is that we probably should fix this. Can we potentially use a Set to detect duplicates quickly?

Marian: What is your opinion of this bug?

> - a known issue discovered relatively recent (more details in #5056) which causes records with default values to be validated even if they are break the unique constraint of an index.

You're working on this one now?

> The remaining failing issues are related to XML/JSON external serialization,

You mean the 2 items above? Or are there some other issues?

#29 - 01/07/2021 03:03 PM - Ovidiu Maxiniuc

Greg Shah wrote:

> My initial reaction is that we probably should fix this. Can we potentially use a Set to detect duplicates quickly?

Probably yes. I will try this way, although it is a bit more complicated. Depending on the state of the active attribute of other instances, the newly added relation can be legal or not. I also lack other informations here: how is the fill method affected? what happens when one these tangled relation changes it active state? what is other does?

> > - a known issue discovered relatively recent (more details in #5056) which causes records with default values to be validated even if they are break the unique constraint of an index.
>
> You're working on this one now?

No.

> > The remaining failing issues are related to XML/JSON external serialization,
>
> You mean the 2 items above? Or are there some other issues?

Yes, there are some other issues. Sorry for misleading phrase. I am focused on these now.

#30 - 01/07/2021 03:33 PM - Eric Faulhaber

Ovidiu, I've updated then reverted (to adjust for an apparent rebase since my last update) my older checkout of 4397a. After, it shows an unknown file:

```
src/com/goldencode/util/OperationTimer.java
```

I'm not sure how this file ended up as unknown to bzr, since it's not my file (I haven't made any changes to this branch). What is the intended status of this file as of rev 11924?

#31 - 01/07/2021 03:34 PM - Eric Faulhaber

Actually, never mind. I'll just do a fresh checkout.

#32 - 01/11/2021 02:11 AM - Marian Edu

Greg Shah wrote:

> Marian: What is your opinion of this bug?

That is definitively an undiscovered bug, no one probably though this is even possible... they are mostly dealing with developers not end users that can try everything without thinking too much about it :)

If you want we can write some more tests to see how the dataset behaves when this odd situation occurs but imho this should not concern you much, I would simply throw an error when a duplicate relation is being created. I do think they do detect circular relations (A->B, B->A) and throw an error. This is not how the 4GL works but as said this is just an unknown bug for them right now and will certainly be fixed when found - fail fast is what I would do here.

> - a known issue discovered relatively recent (more details in #5056) which causes records with default values to be validated even if they are break the unique constraint of an index.

Unique indexes in 4GL (and same is in SQL standard) can allow duplicate records but only if the field part of the index is not mandatory and hence accept NULL as value - multiple rows with NULL value are accepted, non-NULL values must be unique though.

#33 - 01/11/2021 04:10 AM - Eric Faulhaber

I'm reviewing 4397a/11924. Since it's so big, I'm not going to present a single review of all the changes, but rather I'll discuss items that I discover as I go...

First, while I'm reviewing the changes for all files in 4397a, a number of files are not persistence related. I'll catch what I can, but someone more familiar with those non-persistence areas should also review.

Considering that this branch is based on 3821c and will be merged back into it, please remove the revision numbers from the file headers of individual files which already have been assigned revision numbers for 3821c. Eventually, the entire 3821c branch will be merged to trunk, and each file should have a single revision number for that merge, regardless of how many individual updates were made to that file.

Please help me understand the no-undo-tt attribute assigned to some temp-table ASTs in p2o.xml. These ASTs eventually will drive the creation of temp-table DMOs, which could be common across programs. Is this annotation being used as a further distinguishing characteristic to differentiate DMO definitions which should be separate instead of common? In other words, will we now generate different DMO interfaces for temp-tables which are otherwise structurally identical, but differ only by UNDO-ability?

#34 - 01/11/2021 07:31 AM - Greg Shah

> If you want we can write some more tests to see how the dataset behaves when this odd situation occurs but imho this should not concern you much, I would simply throw an error when a duplicate relation is being created. I do think they do detect circular relations (A->B, B->A) and throw an error. This is not how the 4GL works but as said this is just an unknown bug for them right now and will certainly be fixed when found - fail fast is what I would do here.

I'm fine with raising an ERROR condition when this is encountered. Considerations:

- We will need to create a new Redmine issue for this quirk, with links to this task (and any others?).
- We should know (and document in Redmine) the extent of the difference with some tests, if they can be written quickly.
- We need to be able to map the raised ERROR condition to that Redmine quirk issue, so that we spend no extra time in diagnosing this deviation.

#35 - 01/11/2021 07:37 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> I'm reviewing 4397a/11924. Since it's so big, I'm not going to present a single review of all the changes, but rather I'll discuss items that I discover as I go...

That's perfectly fine for me.

> First, while I'm reviewing the changes for all files in 4397a, a number of files are not persistence related. I'll catch what I can, but someone more familiar with those non-persistence areas should also review.

This happened because they share the same interfaces. One example is Errorable. In trunk/3821c, the accessors use get/set prefixes. But the BufferImpl also implements it and this naming will collide with eventual accessors of an error field.

> Considering that this branch is based on 3821c and will be merged back into it, please remove the revision numbers from the file headers of individual files which already have been assigned revision numbers for 3821c. Eventually, the entire 3821c branch will be merged to trunk, and each file should have a single revision number for that merge, regardless of how many individual updates were made to that file.

That requires a bit of work since I do not know which are the entries specific to 3821c. I will do it with my next commit.

> Please help me understand the no-undo-tt attribute assigned to some temp-table ASTs in p2o.xml. These ASTs eventually will drive the creation of temp-table DMOs, which could be common across programs. Is this annotation being used as a further distinguishing characteristic to differentiate DMO definitions which should be separate instead of common? In other words, will we now generate different DMO interfaces for temp-tables which are otherwise structurally identical, but differ only by UNDO-ability?

The no-undo-tt TRPL annotation is used just to generate the noUndo attribute in @Table Java annotation. It is used just for legacy marking. I am not sure if this should also be combined with the directory switch we use at runtime to mark temp-tables globally as no-undo. If so, this is a simple runtime check, and we can overwrite the hardcoded value from the annotation of DMO interface.

#36 - 01/12/2021 03:07 AM - Eric Faulhaber

4397a/11924 code review (cont):

SSL.java:328 contains the following comment:

```
      // OM : got these messages: cause under investigation
```

Was this resolved, or an issue opened about it?

BufferField and BufferFieldImpl: typos in the header entry.

BufferFieldImpl.setDecimals: how does a change in this setting flow back to the backing table, whose data type includes the number of decimals?

BufferImpl:

- Comments for FindMode enum seem wrong through cut and paste
- error() and rejected(): is there any condition/event that resets the flags once set, or does an error condition remain permanently, unless changed by further calls to the corresponding setters?
- There is a lot of new logic around saving and merging changes that I cannot vet with a simple review; it must be tested

DataSet:

- There is a lot of change here, and I have to assume it is correct, because I just don't know the expected behavior. I assume these changes (those that are not adding new functionality, at least) were made based on test results.
- Two of the addRelation variants (the ones with the "stupid manual" comments) have a notActive parameter, while the other has an active parameter. This is confusing. Is it intentional and correct?
- In addParentIdRelation, there is a block commented out that starts with @if (!allowsActiveChild(bufChild)). Is this a final change, or something that was experimental?
- The javadoc for the active parameter (previously notActive) of addRelationWorker now looks wrong.

DataSetParameter.createDataSet: a TODO was added. How important is it to implement this?

QueryWrapper.offEnd is possibly set at query open and in maybeFireCallback. Should it ever be unset? Why do we track this state in the wrapper, separately from the off-end state of the delegate query?

RecordBuffer:

- Can maybeFireRowUpdate throw any runtime exception? If so, please double check the places where state is set and reverted around this call (e.g., commit(boolean)) to make sure the state is correct in the event of an exceptional exit.
- In create, there is new code for temporary buffers, which invokes BufferImpl.dataSet() on every call. I am concerned this is an expensive operation for the >99% of calls to this method, for which this block does not apply.
- I am concerned about setCurrentRecord being non-private. This is a very sensitive method, which I made a point of not exposing outside the class, and which I didn't want to be proxied. It is hard to tell by looking only at diffs which external code now invokes it. Please explain the rationale behind the change.
- Some new methods are missing javadoc.

TempTable: please javadoc each of the SCHEMA_MARSHAL_* constants.

TemporaryBuffer:

- Honestly, I'm a little lost in many of the changes in this class (and some of the related dataset features), and I'm relying on your expertise and on testing to flush out problems.
- Adrian, could you please review the changes where they intersect with your recent updates for fast-copy? Honestly, I'm not familiar enough with this code to review properly.
- Some rogue indents at 4811-4813.
- In copyAllRows at line 3527, you added a second parameter (true) to the RecordBuffer.validate call. This parameter was only meant to be true in a very specific case when called from TransactionManager, to emulate a 4GL quirk. Are you sure it is appropriate here?

Property annotation: I don't think the change to the default from 10 to 0 for the scale is correct. The default number of decimal places to the right of the decimal point for a 4GL decimal type is 10, AFAIK. On what did you base this change?

Table annotation: please see my question in a previous post about the no-undo-tt annotation for temp-table ASTs. Same question for the new noUndo property here. I guess we will produce different DMOs for temp-tables which differ only by NO-UNDO now?

The following need header entries:

- DsRelationDefinition
- P2JQuery
- Record

I have some concerns about the BaseRecord.copy state flag changes and UNDO, but I need to look at it when I'm less tired...

#37 - 01/12/2021 02:10 PM - Adrian Lungu

Eric, the changes on TemporaryBuffer are OK in regard with the temp-table copy. I saw that now there is support for the replace-mode. I was thinking to add replace-mode support for the fast-copy as well, but I need to get a little bit more documented on the subject (I added this matter in #4055-63).

#38 - 01/12/2021 05:58 PM - Eric Faulhaber

4397a/11924 code review (cont)...

DmoMeta.getPrimaryIndex: in implicit primary index case, what constitutes the "first" index? What is the significance of this; i.e., how is the implicit primary index used?

Session.invalidateRecords: under what circumstances is this used?

JsonExport.checkSupportedTypes has this code:

```
      String unsupportedType = Util.checkSupportedTypes(schema);
      if (unsupportedType != null)
      {
         // NOTE: this test is executed for each serialized row! Error 15391 is printed multiple
         //       times in message line, one for each record.
         ...
```

Does that refer to how the 4GL does it, or is that a FWD inefficiency? Can the schema change between rows? If not, can't we check only once and cache the result per export?

XmlImport (possibly other places as well): please do not add new Throwable.printStackTrace calls (and remove/replace old ones when encountered). Better to log these exceptions properly, if failure information is needed.

P2OAccessWorker: the addition of NameConverter.resolvePossibleKeywordConflict calls in the various name getters after the original names are fetched seems wrong. These conflicts should be resolved before the names are stored in the P2OLookup objects, not adjusted on the way out. This worker is just a pass-through access layer for TRPL code to access the P2OLookup objects. All the meaningful work should be done in those objects, not in the access worker.

ControlEntity needs a header entry.

Whew! That's it. Nice work, Ovidiu. That was a huge update.

#39 - 01/12/2021 06:49 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> SSL.java:328 contains the following comment: [...] Was this resolved, or an issue opened about it?

I still have a breakpoint at that location. The execution flow did not reach that point ever since.

> BufferField and BufferFieldImpl: typos in the header entry.

Fixed.

> BufferFieldImpl.setDecimals: how does a change in this setting flow back to the backing table, whose data type includes the number of decimals?

The new value is updated in decimals map of TemporaryBuffer. However, at the second look at the code, there might be an issue because this attribute is stored only for current buffer. It seems logical that once this attribute is changed, it would be seen in all buffers of same table. I need to recheck this.

> BufferImpl:
>
> - Comments for FindMode enum seem wrong through cut and paste

Done.

> - error() and rejected(): is there any condition/event that resets the flags once set, or does an error condition remain permanently, unless changed by further calls to the corresponding setters?

These flags can be manually reset by the programmer and are taken into consideration when processing MERGE-CHANGES and MERGE-ROW-CHANGES. In the later case, the attribute is reset or the records are completely dropped.

> - There is a lot of new logic around saving and merging changes that I cannot vet with a simple review; it must be tested

Indeed, these tests out these methods in a new light and allow me to fix the initial solution. The methods work as expected by xfer tests. I am not aware where/if are they used in client applications.

> DataSet:
>
> - There is a lot of change here, and I have to assume it is correct, because I just don't know the expected behavior. I assume these changes (those that are not adding new functionality, at least) were made based on test results.

Indeed.

> - Two of the addRelation variants (the ones with the "stupid manual" comments) have a notActive parameter, while the other has an active parameter. This is confusing. Is it intentional and correct?

The static definition of the dataset has a NOT-ACTIVE option. If this is present the newly created relation is inactive.
The 5th (optional) logical parameter of the dynamic method (ADD-RELATION) is also called not-active. But the semantic is reversed. I quote the reference manual: *not-active - An optional LOGICAL expression where FALSE causes the data-relation to be inactive.* I just tried to keep the parameters named the same. Probably I am too aggressive in this regard. I will adjust that.

- In addParentIdRelation, there is a block commented out that starts with @if (!allowsActiveChild(bufChild)). Is this a final change, or something that was experimental?

You will encounter similar blocks in other places. They occurred because of the differences in P4GL between my initial implementation (based on my local OE 11.6) and OE 12.x which was used to write the xfer testcases. Usually they are common sense blocks and I think they were not completely dropped in newer versions, but I do not have access to OE 12.x to test directly.

- The javadoc for the active parameter (previously notActive) of addRelationWorker now looks wrong.

Indeed. I fixed it.

> DataSetParameter.createDataSet: a TODO was added. How important is it to implement this?

Logically, it is a must. However, since the old implementation was based on a wrong data, I doubt the code ever actually worked. I do not have an environment to test it. I preferred the TODO instead of a 'blind' solution.

> QueryWrapper.offEnd is possibly set at query open and in maybeFireCallback. Should it ever be unset? Why do we track this state in the wrapper, separately from the off-end state of the delegate query?

The offEnd is initialized in open(). Normally, if the delegate opens correctly, offEnd is reset to false. There are tens of places where the delegate's offEnd changes during complex operations. I preferred to let these operations end and then analyze the result to decide whether the callback needs to be called instead of getting the notification when internal data of the delegate was not yet finished processing. I think it is safer this way.

> RecordBuffer:

- Can maybeFireRowUpdate throw any runtime exception? If so, please double check the places where state is set and reverted around this call (e.g., commit(boolean)) to make sure the state is correct in the event of an exceptional exit.

The maybeFireRowUpdateEvent() method launches converted ABL code which can raise some conditions. But they should be dropped by ErrorManager.clearPending(); as this is what happens in P4GL. I am not aware of other exceptions. But if a NPE occurs, shouldn't it be fixed locally?

- In create, there is new code for temporary buffers, which invokes BufferImpl.dataSet() on every call. I am concerned this is an expensive operation for the >99% of calls to this method, for which this block does not apply.

The public dataSet() and the method it calls use handle wrappings heavily. I think I can add a faster version using package-wide members instead. It should be faster.

- I am concerned about setCurrentRecord being non-private. This is a very sensitive method, which I made a point of not exposing outside the class, and which I didn't want to be proxied. It is hard to tell by looking only at diffs which external code now invokes it. Please explain the rationale behind the change.

I am concerned, too, that's why I just commented out and not completely drop the private access modifier. The problem is that there are a few ABL method which process batches of records identified by specific criteria. I am able to obtain those DMO-s really fast by using SQL queries, but they need to be loaded in the RecordBuffer. At this moment there are only four cases: BufferImpl:9162, 9179, 9191, TemporaryBuffer:3473. I am afraid that there will be more in some places like serial package, which will make things even worse. I am still thinking of a solution here.

- Some new methods are missing javadoc.

I think I already added those in r11925.

> TempTable: please javadoc each of the SCHEMA_MARSHAL_* constants.

Done.

> TemporaryBuffer:
>
> - Honestly, I'm a little lost in many of the changes in this class (and some of the related dataset features), and I'm relying on your expertise and on testing to flush out problems.
> - Adrian, could you please review the changes where they intersect with your recent updates for fast-copy? Honestly, I'm not familiar enough with this code to review properly.
> - Some rogue indents at 4811-4813.

They occurred with rebase operations. Removed.

> - In copyAllRows at line 3527, you added a second parameter (true) to the RecordBuffer.validate call. This parameter was only meant to be true in a very specific case when called from TransactionManager, to emulate a 4GL quirk. Are you sure it is appropriate here?

I noticed that the last record which was validated remained in buffer but not present in database, so it was not visible to low level SQL queries further needed by copyAllRows caller. I really thought about this and I think it is possible to flush it (maybe release the buffer) after the method ends. I will try do to that and re-run the test sets. It it works, I will drop the new extra parameter to validate.

> Property annotation: I don't think the change to the default from 10 to 0 for the scale is correct. The default number of decimal places to the right of the decimal point for a 4GL decimal type is 10, AFAIK. On what did you base this change?

Please look at PropertyMeta.getDecimals(). Indeed, the returned value is 10 if the annotation is 0. The null value is needed as a marker that the attribute is not specified. There are some places whe this information is needed. But when a decimal value is instantiated, the PropertyMeta.getDecimals() value is to be used.

> Table annotation: please see my question in a previous post about the no-undo-tt annotation for temp-table ASTs. Same question for the new noUndo property here. I guess we will produce different DMOs for temp-tables which differ only by NO-UNDO now?

So, the question is: if we have two otherwise identical TT but with different NO-UNDO we should have the same DMOs? If we want to keep full compatibility with P4GL we need to be able to access the UNDO attribute which is also writable (in some conditions). For example, copy-temp-table or create-like will keep this attribute.

> The following need header entries:
>
> - DsRelationDefinition
> - P2JQuery
> - Record

Will be added with next commit.

> I have some concerns about the BaseRecord.copy state flag changes and UNDO, but I need to look at it when I'm less tired...

Thank you very much. There a lot of changes. I also got a bit dizzy searching for answers to your questions/notes.

#40 - 01/12/2021 06:56 PM - Ovidiu Maxiniuc

Adrian Lungu wrote:

> Eric, the changes on TemporaryBuffer are OK in regard with the temp-table copy. I saw that now there is support for the replace-mode. I was thinking to add replace-mode support for the fast-copy as well, but I need to get a little bit more documented on the subject (I added this matter in #4055-63).

Yes, the replace-mode was not properly implemented at first time. And there are a few other places where it is still not correctly implemented or even missing (JSON file read). I am working on them right now.
But I am thinking the code should be extracted in a global single point. OTOH, each place now needs it is probably locally optimized somehow.

#41 - 01/12/2021 07:14 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> 4397a/11924 code review (cont)...
> DmoMeta.getPrimaryIndex: in implicit primary index case, what constitutes the "first" index? What is the significance of this; i.e., how is the implicit primary index used?

This is the first declared index. From my tests, order seems important. I am not sure I fully understand your question.

> Session.invalidateRecords: under what circumstances is this used?

At this moment this is used twice, only from BufferImpl.acceptChanges(). The method uses a low level SQL query to bulk alter possibly all the records from the table (by accepting the changes, all reserved properties are reset and the before-table is cleared). This does not changes the actual client data in after-table, but the Session is unaware of these changes and will provide invalid 'cached' records for queries involving reserved properties.

> JsonExport.checkSupportedTypes has this code: [...]
> Does that refer to how the 4GL does it, or is that a FWD inefficiency? Can the schema change between rows? If not, can't we check only once and cache the result per export?

From my tests, this is how 4GL does it. A bit strange if you ask me. I would expect the method to stop at first error.

> XmlImport (possibly other places as well): please do not add new Throwable.printStackTrace calls (and remove/replace old ones when encountered). Better to log these exceptions properly, if failure information is needed.

OK. Will be done.

> P2OAccessWorker: the addition of NameConverter.resolvePossibleKeywordConflict calls in the various name getters after the original names are fetched seems wrong. These conflicts should be resolved before the names are stored in the P2OLookup objects, not adjusted on the way out. This worker is just a pass-through access layer for TRPL code to access the P2OLookup objects. All the meaningful work should be done in those objects, not in the access worker.

I understand. I will move the check upstream.

> ControlEntity needs a header entry.

Will be added with next commit.

> Whew! That's it. Nice work, Ovidiu. That was a huge update.

Thank you. This is the result of multiple weeks of work. And there are still some changes, but the big ones are already in.

#42 - 01/18/2021 03:14 PM - Ovidiu Maxiniuc

Ovidiu Maxiniuc wrote:

> > P2OAccessWorker: the addition of NameConverter.resolvePossibleKeywordConflict calls in the various name getters after the original names are fetched seems wrong. These conflicts should be resolved before the names are stored in the P2OLookup objects, not adjusted on the way out. This worker is just a pass-through access layer for TRPL code to access the P2OLookup objects. All the meaningful work should be done in those objects, not in the access worker.
>
> I understand. I will move the check upstream.

I tried to do this but I encountered some unexpected aspects. There are two cases: the table and the field names. The latter are OK because their name is used unchanged, as computed initially. But for the buffer name things stay a bit different, in the sense that the DMO undecorated interface name is used (for temp-tables the _k_j suffix not yet appended). To obtain the buffer name that name is de-capitalized. To make sure that does not conflicts with SQL, NameConverter.resolvePossibleKeywordConflict() should already had testes with de-capitalized possible parameter. OTOH, this might be too generic.

But, OTOH, the P2OAccessWorker.javaBufferName() is not returning a stored value from a P2OLookup data structure, it always does a local processing. And, as long as it is consistent across calls, it should be fine. What bothers me now is the inconsistency between computing the name of the buffer and fields: one checks the SQL name collisions when constructing the P2OLookup data structure, the other processes it after the value is extracted.

#43 - 01/21/2021 12:50 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> But, OTOH, the P2OAccessWorker.javaBufferName() is not returning a stored value from a P2OLookup data structure, it always does a local processing. And, as long as it is consistent across calls, it should be fine. What bothers me now is the inconsistency between computing the name of the buffer and fields: one checks the SQL name collisions when constructing the P2OLookup data structure, the other processes it after the value is extracted.

You are right. Both checks should be done when constructing the P2OLookup data structure and the returned values should be deterministic/consistent. I should not have done the lowercasing in the access layer; this layer should just be about returning pre-computed values.

#44 - 01/21/2021 05:16 PM - Ovidiu Maxiniuc

I tried to configure the buffer name initialization in P2OLookup.addJavaName() but there is a problem. The javaNameMap contains two kind of information: if the legacy name key (interned, in lowecase) is qualified the value is a field name and otherwise (the key is not qualified) the value is the java buffer's class name (with first letter capitalized, per Java recommendation). The buffer variable name is obtained dynamically, later, in P2OAccessWorker by decapitalizing it. The name of the buffer class does not pose any problems, but the (default) buffer name will, as it will be used in SQL statements.

I did spent some time thinking of a solution here. We could store the buffer variable name, too, but we have to add a special 'legacy' key for it. I almost started the implementation but then something stroke. Why would Order (class name) pass the SQL collision check but order (buffer name variable) not? They should both fail and be decorated, as if any of them reached the SQL tier, they are exactly the same, since SQL is a case insensitive language. It turned out the NameConverter.resolvePossibleKeywordConflict() was not checking for collision in case insensitive mode for TYPE_DATABASE, TYPE_TABLE and TYPE_COLUMN. After this change both field name and buffer/alias names should not collide with SQL keywords.

#45 - 01/28/2021 01:39 PM - Eric Faulhaber

Code review 4397a/11958:

RecordBuffer.setCurrentRecord being non-private is still a problem. See comments in #4397-36, #4397-39.

SourceData/TargetData: just a minor format issue, but for some reason, the file header and license text bodies have been indented by one character, unlike every other source file in the project. Please change this back for consistency.

TableMapper: is DECIMALS the only mutable, temp-table attribute? It looks like this is the only one implemented as mutable currently. Is IllegalStateException the correct action when an attempt is made to set a non-mutable attribute? What does the original environment do in response to such an attempt? Please set the one-parameter-per-line format back for the LegacyFieldInfo c'tor calls in TableMapper$LegacyTableInfo.loadFields(DmoMeta) method (lines 3072, 3102). It may be more verbose looking the previous way, but I think the compact form now is much harder to read, considering all the parameters to track.

JsonExport: exportRecord and serializeTempTable have commented supported type checks, which is confusing. Is this supposed to be there?

P2OAccessWorker: javaBufferName still changes the value returned by javaClassName, rather than just acting as a pass-through layer for information stored in P2OLookup objects. I know you did not write this code, but can we leave it this way?

TargetLob needs a header entry.

Please resolve these issues before merging to 3821c. We can work any unfinished issues in 3821c.

#46 - 01/28/2021 07:11 PM - Eric Faulhaber

Code review 4397a/11991:

Continuing the discussion from #5056-14, I'm still convinced the BufferImpl.validate() changes are not correct. **In the silent error case**, the same ValidationException instance will cause two calls to ErrorManager.recordOrThrowError. The first call to recordOrThrowError will not throw ErrorConditionException due to silent error mode. Then we will return normally to RB.validate(boolean) and the original ValidationException will be rethrown. This will be caught by BufferImpl.validate() and will cause a second call to ErrorManager.recordOrThrowException for the same ValidationException instance.

I like the changes to cut out TableMapper, except that in a number of those places (mostly in BufferImpl), we are now invoking the BufferImpl.buffer() method when we don't need to. This is an expensive method, because each call has to go through a proxy (or two in some TemporaryBuffer cases) just to get the proxy's RecordBuffer instance. Granted, it returns from the invocation handler methods quickly, but why do this work unnecessarily? Please call buffer() no more than once per method, store the RecordBuffer instance returned in a local variable, and reuse it.

I know this method already is called multiple times in some other BufferImpl methods. I try to clean this up every time I edit that file, but it seems to keep being added back.

Likewise, it seems we have an unnecessary call to buffer() in the TempTableSchema c'tor.

Please fix the header in AppServerHelper.

Not sure why the *Keyboard classes are in this update.

Everything else looks good.

#47 - 01/28/2021 07:41 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Code review 4397a/11958:
> RecordBuffer.setCurrentRecord being non-private is still a problem. See comments in #4397-36, #4397-39.

Fixed. Somewhat. Not entirely please with the solution but at least the problematic calls are not separated from normal private ones.

> SourceData/TargetData: just a minor format issue, but for some reason, the file header and license text bodies have been indented by one character, unlike every other source file in the project. Please change this back for consistency.

Strange. Probably the IDE helped me. Fixed back.

> TableMapper: is DECIMALS the only mutable, temp-table attribute? It looks like this is the only one implemented as mutable currently.

No, it is not the only one. The other known (to me, yet) are grouped into serializeOptions.

> Is IllegalStateException the correct action when an attempt is made to set a non-mutable attribute? What does the original environment do in response to such an attempt?

At that level, the exception is OK. The LegacyFieldInfo.setDecimals(int) is only called from BufferFieldImpl.setDecimals(long). At this level the error condition 17092 is thrown if the programmer attempts to modify the permanent schema.

> Please set the one-parameter-per-line format back for the LegacyFieldInfo c'tor calls in TableMapper$LegacyTableInfo.loadFields(DmoMeta) method (lines 3072, 3102). It may be more verbose looking the previous way, but I think the compact form now is much harder to read, considering all the parameters to track.

Done. However, this long set of parameters does not look fine; no mater how we stack them, it will be hard to identify them. I do not recall if we have some code guidelines for such thing but personally I prefer to have maximum of 5-6 parameters.

> JsonExport: exportRecord and serializeTempTable have commented supported type checks, which is confusing. Is this supposed to be there?

Initial code was implemented with OE11.6. In OE11.7 and 12.x things seem a bit different, but I do not like to throw away the code without proper testing.

> P2OAccessWorker: javaBufferName still changes the value returned by javaClassName, rather than just acting as a pass-through layer for information stored in P2OLookup objects. I know you did not write this code, but can we leave it this way?

According to javaBufferName's javadoc, the returned value is expected to be "simply the name of the appropriate DMO interface, with the first letter lowercased."

> TargetLob needs a header entry.

Done.

> Please resolve these issues before merging to 3821c. We can work any unfinished issues in 3821c.

I would prefer 3574a instead, if possible.

#48 - 02/19/2021 10:48 AM - Greg Shah

- Status changed from Test to Closed

Ovidiu has recently confirmed that testing passed for the features in this task.