Database - Feature #3813

misc DB features part deux

11/25/2018 09:25 AM - Greg Shah

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

Description

Related issues:

Related to Database - Feature #3758: misc database features	**Closed**	
Related to Database - Feature #4175: add support for -1 (single user) option ...	**Closed**	
Related to Database - Bug #4535: CONNECT statement with VALUE expression quirks	**New**	**02/11/2020**
Related to Database - Bug #4537: CONNECT statement parameters parsing quirks	**New**	**02/11/2020**
Related to Database - Feature #4375: add support for -pf <profile_filename> o...	**Closed**	
Related to Database - Feature #4613: add missing support for database user de...	**New**	
Related to Base Language - Feature #4392: add -pf support for appserver CONNE...	**Test**	

History

#1 - 11/25/2018 09:25 AM - Greg Shah

- Related to Feature #3758: misc database features added

#2 - 11/25/2018 09:28 AM - Greg Shah

Language Statements

- ~~BUFFER-COMPARE~~ - implement NO-LOBS (already done for the POC?)
- ~~BUFFER-COPY~~ - implement NO-LOBS (already done for the POC?)
- ~~COPY-LOB~~ (will be worked on in #2135 phase 3)

Built-In Function

- DBPARAM - there is a limited implementation in ConnectionManager; the limitation is that we only report the connection options we support, currently: -db (physical database), -dt (database type - always "PROGRESS"), -ld (logical database), -H (host), -S (socket/service), -U (userid), -P (password; actual value not reported), -1 (single client, reported but not enforced currently).

Attributes

- ~~BUFFER:AFTER-ROWID~~ (part of ProDataSet support #3809)
- ~~BUFFER:BEFORE-ROWID~~ (part of ProDataSet support #3809)
- ~~BUFFER:ORIGIN-ROWID~~ (part of ProDataSet support #3809)
- ~~BUFFER:AFTER-BUFFER~~ (part of ProDataSet support #3809)
- ~~BUFFER:BEFORE-BUFFER~~ (part of ProDataSet support #3809)
- ~~BUFFER:RECORD-LENGTH~~ (probably just uses the RECORD-LENGTH() function implementation, see #3858)

- BUFFER-FIELD:DEFAULT-STRING (needs implementation - looks like this is just an unformatted string representation of a field's INITIAL value)
- BUFFER-FIELD:DECIMALS (needs implementation)

- ~~QUERY:BASIC-LOGGING~~ (will be worked with LOG-MANAGER support, see #3853)

Methods

- ~~BUFFER:BUFFER-COMPARE~~ - implement NO-LOBS (already done for the POC?)
- ~~BUFFER:BUFFER-COPY~~ - implement NO-LOBS (already done for the POC?)
- ~~BUFFER:BUFFER-VALIDATE~~
- BUFFER:RAW-TRANSFER - finish implementation

- ~~QUERY:CREATE-RESULT-LIST-ENTRY~~ (this is already being worked in #3762)
- ~~QUERY:GET-BUFFER-HANDLE~~

- TEMP-TABLE:CREATE-LIKE
- ~~TEMP-TABLE:READ-XML~~ (this should already be complete from #3809 and #3574)
- ~~TEMP-TABLE:WRITE-XML~~ (this should already be complete from #3809 and #3574)

Built-In Function Usage in WHERE Clauses

New use cases such as ROW-STATE() and VALID-OBJECT() need to be handled. Reports need to be reviewed to determine the use cases that need support in PL/Java and which are safe for substitution parameter usage.

#3 - 05/14/2019 05:34 PM - Constantin Asofiei

DEFAULT-STRING conversion and runtime stub was added in 3751a rev 11322

#4 - 05/24/2019 01:25 PM - Constantin Asofiei

BUFFER-VALIDATE method is required for the current large app work.

#5 - 01/13/2020 08:21 AM - Greg Shah

- Status changed from New to WIP

- Assignee set to Igor Skornyakov

From Eric:

> Pending Ovidiu's review, I believe the rest of #3813-2 is now pretty accurate, though I don't know how heavily some of the recently implemented features have been tested. Looks like the main work left is the 2 BUFFER-FIELD attributes (doesn't look like a lot of work); BUFFER:RAW-TRANSFER (don't think that has been touched since you started the implementation); and catching the UDF support up with some more recently implemented built-ins. As your comment in Redmine regarding the latter suggests, it may not be possible/practical to implement some built-ins as UDFs, depending on whether they need access to state from the persistence runtime.
>
> Ovidiu: Could you please review #3813-2 and provide your feedback on the status of these features? I just went through and updated the status as I understand it, but there are some features you've worked on/around recently and I'm not sure where they stand ATM. Specifically:
>
> - TEMP-TABLE:CREATE-LIKE method: it looks like you did some refactoring recently for your dataset work. What is the current status of this method's implementation, both for temp-table and dataset?
> - Please comment on the "Built-In Function Usage in Where Clauses" section, to the extent you are aware of new features implemented in the course of the dataset work which may not yet be reflected in our UDF support (assuming such support is sensible/feasible).

From Ovidiu:

> The TEMP-TABLE:CREATE-LIKE was renamed (the former name was "createTableLike") because the same method name is used by PDS so they share the same interface. The Dataset variant should be fully functional. I do not recall changing the TempTable actual implementation.
>
> Regarding the 'Built-In Function Usage in WHERE Clauses'. In 4124a there was added support for ROW-STATE, but in 3809e (based on 4124a) the BEFORE-ROWID, AFTER-ROWID, ERROR, ERROR-STRING functions/attributes (where applicable) was also added. The current implementation converts these functions/attributes into the appropriate fields/properties, so no support is needed in UDF.

4124a was merged to trunk, but 3809e is not yet in trunk. I expect it in trunk this week.

#6 - 01/13/2020 08:52 AM - Igor Skornyakov

Which branch should I use for this task?
Thank you.

#7 - 01/13/2020 08:58 AM - Greg Shah

4335a

#8 - 01/21/2020 12:17 PM - Igor Skornyakov

The following statement from the Progress documentation regarding DBPARAM functions (see https://docs.progress.com/bundle/abl-reference/page/DBPARAM-function.html) looks incorrect or at least misleading:

```
If the CONNECT statement contains the -pf parameter, which refers to a parameter file, the string DBPARAM retu
rns includes the parameters in the file without "-pf" or any reference to the file.
```

For example, I have a .pf file with multiple parameters, but only two of them are included in the DBPARAM result in addition to the parameters already supported by FWD: -Bt (TEMP-TABLE-BUFFERS) and -fc (SCHEMA-FLD-CACHE-SZ).
I understand that these parameters have meaningful counterparts in FWD.
Of course, I can try to figure the exact list of the parameter file parameters which are considered by DBPARAM. However, I suspect that most of them will have no FWD counterparts as well.
My question is. How the result of the DBPARAM function is supposed to be used, or how it is used in the application(s) of our customers?
Thank you.

#9 - 01/21/2020 03:17 PM - Greg Shah

> For example, I have a .pf file with multiple parameters, but only two of them are included in the DBPARAM result in addition to the parameters already supported by FWD: -Bt (TEMP-TABLE-BUFFERS) and -fc (SCHEMA-FLD-CACHE-SZ).

Are you saying that the 4GL includes some but not all of the parameters listed in the .pf file in its DBPARAM result?

> I understand that these parameters have meaningful counterparts in FWD.

Not always. Internally, the FWD implementation is very different from the 4GL. For example, the -Bt (TEMP-TABLE-BUFFERS) and -fc (SCHEMA-FLD-CACHE-SZ) seem unlikely to map to anything useful in FWD.

> I can try to figure the exact list of the parameter file parameters which are considered by DBPARAM.

Yes, I think this is needed.

> However, I suspect that most of them will have no FWD counterparts as well.

Correct. But even a meaningless parameter probably should be included if specified by the user.

> How the result of the DBPARAM function is supposed to be used, or how it is used in the application(s) of our customers?

I've seen it used in 2 ways:

- Output in logfiles or debug information for support purposes.
- Parsed to answer specific questions (e.g. to see whether a particular value is being passed).

#10 - 01/21/2020 03:33 PM - Igor Skornyakov

Greg Shah wrote:

> > For example, I have a .pf file with multiple parameters, but only two of them are included in the DBPARAM result in addition to the parameters already supported by FWD: -Bt (TEMP-TABLE-BUFFERS) and -fc (SCHEMA-FLD-CACHE-SZ).
>
> Are you saying that the 4GL includes some but not all of the parameters listed in the .pf file in its DBPARAM result?

Exactly.

> Yes, I think this is needed.

OK. Thank you.

> > How the result of the DBPARAM function is supposed to be used, or how it is used in the application(s) of our customers?
>
> I've seen it used in 2 ways:
>
> - Output in logfiles or debug information for support purposes.
> - Parsed to answer specific questions (e.g. to see whether a particular value is being passed).

I see. Thank you.

#11 - 01/23/2020 05:02 PM - Igor Skornyakov

Here is a (hopefully) complete list of database connection parameters shown by DBPARAM function.

Parameter	Description
-1	Single-user Mode
-adminport *n*	AdminServer Port
-aibufs *n*	Specifies the number of after-image buffers when running AIW
-aistall	Suspends database activity when an empty after-image (AI) file is unavailable
-B *n*	Specifies the number of blocks in the database buffers
-bibufs *n*	Specifies the number of before-image buffers when running BIW
-bistall	Sends a message to the log file when the recovery log threshold is reached. Use with *-bithold*
-bithold *n*	Specifies the maximum size of the recovery log files
-Bp *n*	Requests a number of private read-only buffers
-Bpmax *n*	Requests a maximum number of private buffers
-c *n*	Specifies the number of index place holders or cursors
-cache *filename*	Reads the database schema from a local file instead of the database
-classpath *pathname*	Identifies the Java classpath to use when starting an SQL server
-cs *n*	Changes the maximum number of index levels
-ct *n*	Max number of connection retries
-DataService *ds-name*	The name of the DataService to connect through a NameServer to an ODBC, ORACLE, or SQL Server
-db *name*	Physical Database Name.
-directio	Use Direct I/O to open all files in unbuffered mode
-Dsrv *params*	DataServer parameters
-dt *db-type*	Identifies the database type

-fc *n*	Changes the number (soft limit) of entries in the schema field cache
-G *n*	Specifies the number of seconds before OpenEdge reuses a before-image cluster
-groupdelay *n*	Specifies the number of milliseconds a transaction waits before committing
-H *host-name*	Identifies a remote host
-hash *n*	Specifies the number of hash table entries for the buffer pool
-i	No Crash Protection
-L *n*	Lock Table Entries
-ld *logical-dnname*	Assigns the logical database name
-lkwtmo *n*	Lock Timeout in seconds
-Mm *n*	Specifies the message buffer size for network client/server protocols
-N *network-type*	Network type
-n *n*	Number of Users
-P *value not shown*	password
-pinshm	Pin Shared Memory
-properties *filename*	Identifies the properties file an AdminServer uses when File starting a database server or servergroup
-RO	Open a database for read-only access
-S *service-name* or *port-number*	Service name or port number
-servergroup *name*	Identifies a logical collection of server processes to start
-spin *n*	Identifies a logical collection of server processes to start
-trig *dir-name* or *lib-name*	Identifies the directory or library containing the ABL triggers for a database
-U *userid*	Specifies the user ID

#12 - 01/23/2020 05:14 PM - Igor Skornyakov

Can anybody help me to figure which of the database connection parameters described in the previous note have direct meaningful counterparts in FWD?
Thank you.

#13 - 01/23/2020 06:15 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Can anybody help me to figure which of the database connection parameters described in the previous note have direct meaningful counterparts in FWD?
> Thank you.

Currently, ConnectionManager.connect supports:

```
-1 (reported, but not yet enforced)
-db
-H
-ld
-P
-S
-U
```

We've considered -RO, but it is not yet implemented.

#14 - 01/24/2020 03:31 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
>
> > Can anybody help me to figure which of the database connection parameters described in the previous note have direct meaningful counterparts in FWD?
> > Thank you.
>
> Currently, ConnectionManager.connect supports:
>
> [...]
>
> We've considered -RO, but it is not yet implemented.

I see. Thank you.
I understand that I should add support for the remaining options in the CONNECT statement by just storing values and adding them to the DBPARAM.
Is my understanding correct?
Thank you.

#15 - 01/24/2020 08:11 AM - Greg Shah

- Related to Feature #4175: add support for -1 (single user) option for CONNECT statement added

#16 - 01/24/2020 08:12 AM - Greg Shah

- Related to Feature #4392: add -pf support for appserver CONNECT() method added

#17 - 01/24/2020 08:17 AM - Greg Shah

> I understand that I should add support for the remaining options in the CONNECT statement by just storing values and adding them to the DBPARAM.
>
> Is my understanding correct?

Yes.

> For example, I have a .pf file with multiple parameters, but only two of them are included in the DBPARAM result in addition to the parameters already supported by FWD: -Bt (TEMP-TABLE-BUFFERS) and -fc (SCHEMA-FLD-CACHE-SZ).

One thing that was not clear from #3813-11: which of these items can be specified in the CONNECT but do not appear in DBPARAM?

We also have 2 other highly related tasks (#4175 and #4392). I think it makes sense to go ahead and implement them at this same time. This is the runtime support for -1 and the -pf support.

#18 - 01/24/2020 08:22 AM - Igor Skornyakov

Greg Shah wrote:

> One thing that was not clear from #3813-11: which of these items can be specified in the CONNECT but do not appear in DBPARAM?

I've tested these items only with the configuration (.pf) file. According to the Progress documentation, all of them can be specified in the CONNECT statement. I expect that they will appear in DBPARAM in this case as well. However, I will double-check.

> We also have 2 other highly related tasks (#4175 and #4392). I think it makes sense to go ahead and implement them at this same time. This is the runtime support for -1 and the -pf support.

Yes, I saw it. Thank you.

#19 - 01/24/2020 11:10 AM - Igor Skornyakov

All connection parameters from #3813-11 are accepted in the CONNECT statement and are shown by DBPARAM in this case.

#20 - 01/24/2020 11:34 AM - Greg Shah

Igor Skornyakov wrote:

> All connection parameters from #3813-11 are accepted in the CONNECT statement and are shown by DBPARAM in this case.

Did the original pf file you tested have parameters that are not related to CONNECT? Perhaps that is why they don't show up (unrecognized parms are probably ignored by CONNECT).

#21 - 01/24/2020 11:55 AM - Igor Skornyakov

Greg Shah wrote:

> Igor Skornyakov wrote:
> Did the original pf file you tested have parameters that are not related to CONNECT? Perhaps that is why they don't show up (unrecognized parms are probably ignored by CONNECT).

The test .pf files contained parameters which are not db related. 4GL complains about them (I'm going to reproduce this in FWD). I've commented them out. I understand that the very first .pf file I've used for testing also contained ignored parameters.

#22 - 01/27/2020 02:40 PM - Igor Skornyakov

The conversion of the CONNECT statement looks a little bit strange.
For example. The statement

```
CONNECT -1 -adminport 8000 -aibufs 1 -aistall -B 10 -bibufs 20 -bistall -bithold 10 -Bp 1 -Bpmax 1 -c 199 -cac
he E:\testcases\fwd1.bin -classpath c:\ -cs 11 -DataService aaaaaaaaaaa -db E:\testcases\uast\security\fwd1.db
 -directio -Dsrv aaaa -dt PROGRESS -fc 1000 -G 1 -groupdelay 10 -hash 10 -i -L 1000 -ld fwd1 -lkwtmo 1000 -Mm
400 -n 1 -P secret -pinshm -properties ddd -RO -servergroup grp -spin 100 -trig c:\ -U fwdb@domain.
```

Is converted as

```
        ConnectionManager.connect("-1", "-ld", "fwd1", "-P", "-P secret", "-RO", "-U", "-U fwdb@domain");
```

Apart from missed parameters we see that -P and -U appear twice in the converted statement.
Is it done on purpose?
Thank you.

#23 - 01/27/2020 03:34 PM - Eric Faulhaber

Hm, you're correct, that conversion looks completely wrong. I think the conversion may have been broken along the way, compared to the original implementation. Please see the javadoc for ConnectionManager.connect(Object...) and the implementation of that method for the expected input.

Essentially, there are two alternatives:

- a single string with the physical database name OR a space delimited set of options which are parsed by downstream code; OR
- multiple parameters in an expected order (without the option qualifiers), each representing a supported option.

In either case, the duplication of the -P and -U options is incorrect. However, the dropping of unsupported parameters is expected.

#24 - 01/27/2020 03:47 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Hm, you're correct, that conversion looks completely wrong. I think the conversion may have been broken along the way, compared to the original implementation. Please see the javadoc for ConnectionManager.connect(Object...) and the implementation of that method for the expected input.
>
> Essentially, there are two alternatives:
>
> - a single string with the physical database name OR a space delimited set of options which are parsed by downstream code; OR
> - multiple parameters in an expected order (without the option qualifiers), each representing a supported option.

I see. Thank you. Working on the conversion support of additional parameters.

> In either case, the duplication of the -P and -U options is incorrect. However, the dropping of unsupported parameters is expected.

I understand this.

#25 - 01/27/2020 03:52 PM - Eric Faulhaber

The original, ordered list approach was based on a very limited set of connection options, which were basically the minimum required to connect. As we add support for new options, that approach may no longer be feasible, or may continue to be limited to that original set of options.

Also, there are cases where a connect option string is built up at runtime, or comes from a *.pf file. In these cases, we have to deal with legacy options which are inappropriate for FWD's design, or which at least need some additional attention to work with FWD's design.

Of course, legacy, internal implementation detail options should just be ignored.

Options having to do with file system or network resources, if relevant in the new environment, are mapped to the new environment. For instance, when we encounter the file path for a physical database, we attempt to map this to a database configuration in the directory. When we encounter a port service name, this is likewise mapped to a resource in the directory.

If any of the newly supported options have similar characteristics, they should be handled in a similar fashion.

#26 - 01/27/2020 04:09 PM - Igor Skornyakov

Eric Faulhaber wrote:

> The original, ordered list approach was based on a very limited set of connection options, which were basically the minimum required to connect. As we add support for new options, that approach may no longer be feasible, or may continue to be limited to that original set of options.
>
> Also, there are cases where a connect option string is built up at runtime, or comes from a *.pf file. In these cases, we have to deal with legacy options which are inappropriate for FWD's design, or which at least need some additional attention to work with FWD's design.
>
> Of course, legacy, internal implementation detail options should just be ignored.
>
> Options having to do with file system or network resources, if relevant in the new environment, are mapped to the new environment. For instance, when we encounter the file path for a physical database, we attempt to map this to a database configuration in the directory. When we encounter a port service name, this is likewise mapped to a resource in the directory.
>
> If any of the newly supported options have similar characteristics, they should be handled in a similar fashion.

I see my current task as following. We extend CONNECT statement conversion and runtime support so that DBPARAM will return a result that is compatible with 4GL. This does not include support of the semantics of additional options (apart from the -pf option which should be supported in a sense that parameters from the configuration file will be shown by DBPARAM).
Is my understanding correct?
Thank you.

#27 - 01/27/2020 05:09 PM - Eric Faulhaber

For implementation of the DBPARAM builtin function, correct. For those options which we intend to add support, the semantics will be implemented later, in separate tasks.

#28 - 01/27/2020 05:15 PM - Igor Skornyakov

Eric Faulhaber wrote:

> For implementation of the DBPARAM builtin function, correct. For those options which we intend to add support, the semantics will be implemented later, in separate tasks.

Thank you.

#29 - 02/07/2020 03:05 PM - Igor Skornyakov

I've essentially re-worked the CONNECT statement conversion. The only remaining problem I see now is about the parameters with values containing a backslash (e.g. "-cache E:\testcases\fwd1.bin"). The backslash and (in some cases) the next character is "eaten" somewhere. Trying to figure where this happens and how to fix it.

#30 - 02/07/2020 03:48 PM - Greg Shah

Isn't this just the fact that the \ is an escape char and will always be eaten? And when used with some characters (e.g. t, n...) the next character is eaten as well because the combination has special meaning (e.g. \t is a tab character)

#31 - 02/07/2020 04:04 PM - Igor Skornyakov

Greg Shah wrote:

> Isn't this just the fact that the \ is an escape char and will always be eaten? And when used with some characters (e.g. t, n...) the next character is eaten as well because the combination has special meaning (e.g. \t is a tab character)

I think so. However, this means that the conversion will be incorrect in many situations (and, in fact, it is incorrect now for the "CONNECT C:\some\path\somedb." statement. My question is about the processing step when the escape characters are processed.

#32 - 02/07/2020 04:17 PM - Igor Skornyakov

It seems that a backslash is processed at a very early stage. I see eaten chars in a .cache file. Not sure how to deal with this.

#33 - 02/07/2020 04:20 PM - Greg Shah

This means your project is not configured as a Windows project. See p2j.cfg.xml and the <parameter name="opsys" value="WIN32" /> should be set there if it is a windows project.

String literals are normally processed in convert/literals.rules and they should be emitted using progressToJavaString() which in TRPL is backed by

ExpressionConversionWorker and the actual code is in character.progressToJavaString().

#34 - 02/07/2020 04:41 PM - Igor Skornyakov

Greg Shah wrote:

> This means your project is not configured as a Windows project. See p2j.cfg.xml and the <parameter name="opsys" value="WIN32" /> should be set there if it is a windows project.

The "opsys" doesn't help but <parameter name="unix-escapes" value="false" /> resolves the issue.

> String literals are normally processed in convert/literals.rules and they should be emitted using progressToJavaString() which in TRPL is backed by ExpressionConversionWorker and the actual code is in character.progressToJavaString().

I see. Thank you.

#35 - 02/07/2020 04:55 PM - Greg Shah

Setting opsys should have implicitly set unix-escapes, so that sounds like a bug. Please figure that out now, because it will hit someone else at some point. The code that implements this is in Configuration.java.

#36 - 02/08/2020 03:41 AM - Igor Skornyakov

Greg Shah wrote:

> Setting opsys should have implicitly set unix-escapes, so that sounds like a bug. Please figure that out now, because it will hit someone else at some point. The code that implements this is in Configuration.java.

The problem is in line 538 of the Configuration.java:

```
      parmMap.computeIfAbsent("unix-escapes",   x -> String.valueOf(!isWin));
```

This means that even if the incorrect value of the "unix-escapes" was specified before "opsys" (as in my case) it will not be overriden. This seems to be done intentionally, but can be consusing indeed.
From the other side, I do not see a reason for using computeIfAbsent method with a lambda expression with depends only on a local variable of the computeDerivedValues function.
I suggest using

```
      parmMap.put("unix-escapes",  String.valueOf(!isWin));
```

instead.
Is it OK?

#37 - 02/10/2020 10:55 AM - Greg Shah

> This means that even if the incorrect value of the "unix-escapes" was specified before "opsys" (as in my case) it will not be overriden. This seems to be done intentionally, but can be consusing indeed.

Correct. This is intentional. The idea is that it is not appropriate to override an explicit value specified by the user. So, if it is present we will accept whatever value you explicitly specified.

If not present, then we set it to a sensible default.

> I suggest using
> ...
> instead.
> Is it OK?

No, I think it is working as designed.

In your case, were you explicitly setting unix-escapes to true even though opsys was set to win32?

#38 - 02/10/2020 11:10 AM - Igor Skornyakov

Greg Shah wrote:

> In your case, were you explicitly setting unix-escapes to true even though opsys was set to win32?

Yes, unix-escapes was set to true and it was **before** the opsys set. It is a general problem - how the multiple settings of the same parameter are processed.

#39 - 02/10/2020 11:35 AM - Greg Shah

I agree that we need to improve our documentation to explain this. For now, I think you can ignore it.

#40 - 02/10/2020 01:21 PM - Igor Skornyakov

Using VALUE() in the CONNECT statement looks a little tricky. Consider the following code:

```
DEF VAR c1 AS CHAR NO-UNDO.
c1 = " -db E:\testcases\uast\security\fwd1.db".
DEF VAR c2 AS CHAR NO-UNDO.
c2 = " -P secret".
CONNECT VALUE("-1" + c1)  -ld VALUE("fwd1" + c2)  -U fwdb@domain.
```

The first VALUE expression is OK, while the second one results in the ** The character is not permitted in name fwd1 -P secret. (274) error.

#41 - 02/10/2020 02:08 PM - Greg Shah

Is this the 4GL or FWD?

#42 - 02/10/2020 02:10 PM - Igor Skornyakov

Greg Shah wrote:

> Is this the 4GL or FWD?

It is 4GL. The current implementation of FWD doesn't support any of these use cases. Working on it.

#43 - 02/10/2020 05:21 PM - Igor Skornyakov

The VALUE usage in the CONNECT statement looks really tricky. For example

```
CONNECT VALUE("E:\testcases\uast\security\fwd1.db")  -P secret -U fwdb@domain  -bibufs 12 -1 -ld fwd1.
```

works fine

```
CONNECT VALUE("-db E:\testcases\uast\security\fwd1.db  -P secret")  -P secret -U fwdb@domain  -bibufs 12 -1 -l
d fwd1.
```

works too. However

```
CONNECT VALUE("E:\testcases\uast\security\fwd1.db  -P secret")  -P secret -U fwdb@domain  -bibufs 13 -1 -ld fw
d1.
```

fails with Warning: -ld is not a database parameter and was ignored. (1402) error

#44 - 02/10/2020 05:22 PM - Constantin Asofiei

Does CONNECT E:\testcases\uast\security\fwd1.db -P secret -P secret -U fwdb@domain -bibufs 13 -1 -ld fwd1. work in 4GL? (I mean, without VALUE).

#45 - 02/10/2020 05:31 PM - Igor Skornyakov

Constantin Asofiei wrote:

> Does CONNECT E:\testcases\uast\security\fwd1.db -P secret -P secret -U fwdb@domain -bibufs 13 -1 -ld fwd1. work in 4GL? (I mean, without VALUE).

Yes, it works.

#46 - 02/10/2020 05:36 PM - Greg Shah

For now, I don't know of a case where customer code uses this approach. Please create a new task (under the Database project) for this strange behavior. Link it here as a related task and make me a watcher.

We will defer that work.

#47 - 02/10/2020 05:44 PM - Igor Skornyakov

Greg Shah wrote:

> For now, I don't know of a case where customer code uses this approach. Please create a new task (under the Database project) for this strange behavior. Link it here as a related task and make me a watcher.
>
> We will defer that work.

OK. I will do it tomorrow. Regarding the scope of this task. Should I skip support of the VALUE expression containing more than one option? Thank you.

#48 - 02/10/2020 05:51 PM - Greg Shah

#3813 does need to handle the case where there are more than 1 option inside a single VALUE() expression. That is a common thing.

#49 - 02/10/2020 05:55 PM - Igor Skornyakov

Greg Shah wrote:

> #3813 does need to handle the case where there are more than 1 option inside a single VALUE() expression. That is a common thing.

I see. Thank you.

#50 - 02/11/2020 06:18 AM - Igor Skornyakov

- Related to Bug #4535: CONNECT statement with VALUE expression quirks added

#51 - 02/11/2020 07:25 AM - Igor Skornyakov

Some important notes:

1. The CONNECT statement can create more than one connection. This is mentioned in the Progress doc, but I've overlooked this before.
2. The CONNECT statement can have multiple instances of the same key for the same database. Only the latest found affects the actual connection **but** all of them are included in the DBPARAM string.
3. There can be only one parameter not preceded by the key (name started with '-'). It is treated as the **first** database name (as if it was preceded by '-db' key) and it is not necessary the first argument. See however #4535 as the situation becomes more complicated if the VALUE clause is used.

#52 - 02/11/2020 10:31 AM - Greg Shah

Ignoring the quirks in #4535 (which will be deferred), what is the effort to resolve these other issues (from #3813-51)?

#53 - 02/11/2020 11:03 AM - Igor Skornyakov

Greg Shah wrote:

> Ignoring the quirks in #4535 (which will be deferred), what is the effort to resolve these other issues (from #3813-51)?

I hope to do it today.

#54 - 02/11/2020 12:21 PM - Igor Skornyakov

4GL uses some weird logic when parsing CONNECT statement arguments which I cannot understand.
For example, if one uses an incorrect key '-xdb' instead of '-db' the error message is:

```
Warning: -x is not a database parameter and was ignored. (1402)
```

In some situations two errors are generated:

```
Warning: -xd is not a database parameter and was ignored. (1402)
Warning: -b is not a database parameter and was ignored. (1402)
```

Is it OK if FWD will complain about a complete key, e.g.

```
Warning: -xdb is not a database parameter and was ignored. (1402)
```

?
Thank you.

#55 - 02/11/2020 01:36 PM - Greg Shah

> Is it OK if FWD will complain about a complete key, e.g.

Yes, for **now** it is OK. Please create a new task for this quirk and make it related. We will defer the work on that.

#56 - 02/11/2020 03:52 PM - Igor Skornyakov

- Related to Bug #4537: CONNECT statement parameters parsing quirks added

#57 - 02/12/2020 10:23 AM - Igor Skornyakov

Finished rework of CONNECT/DBPARAM.
Committed to 4335a revision 11416.

Working on the parameter file support (#4395). The support should not be complicated, but a number of use cases should be checked.

#58 - 02/13/2020 10:03 AM - Igor Skornyakov

4GL allows -pf option in the .pf file. However, this may result in the application hang due to infinite recursion. I do not think that we have to reproduce this and FWD will report an error if such recursion is discovered.
Is it correct?
Thank you.

#59 - 02/13/2020 10:04 AM - Greg Shah

Yes, failing on recursion is a good idea. Hangs don't have to be duplicated.

#60 - 02/13/2020 04:12 PM - Igor Skornyakov

Added configuration file (-pf option of the CONNECT statement).
Committed to 4335a revision 11419.

Please note that some additional testing is required.

#61 - 02/14/2020 08:20 AM - Igor Skornyakov

Additional tests of the CONNECT -pf option support passed. A minor typo was fixed.
4335a revision 11421 is ready for the code review (files ConnectionManager.java and database_general.rules).

#62 - 02/17/2020 03:19 PM - Igor Skornyakov

The conversion of the uast/raw_transfer/raw_transfer_basic_multi_column_mixed_data_type_encoding.p fails/

```
     [java] EXPRESSION EXECUTION ERROR:
     [java] --------------------------
     [java] throwException( sprintf("## unknown variable type %s/%s/%s", ref.parent.parent, ref.parent, ref))
     [java] ^  { ## unknown variable type FUNC_CHAR/COLON/KW_BUFFER [ATTR_INT id <347892352089> 125:39] }
     [java] --------------------------
     [java] EXPRESSION EXECUTION ERROR:
     [java] --------------------------
     [java] evalLib("gen_accessors", ref1, tmp)
     [java] ^  { Expression execution error @1:1 }
     [java] --------------------------
     [java] ERROR:
     [java] com.goldencode.p2j.pattern.TreeWalkException: ERROR!  Active Rule:
     [java] ----------------------
     [java]       RULE REPORT
     [java] ----------------------
     [java] Rule Type :   WALK
     [java] Source AST:  [ CRC-VALUE ] BLOCK/STATEMENT/KW_DISP/EXPRESSION/FUNC_CHAR/COLON/ATTR_INT/ @125:39 {3
47892352089}
     [java] Copy AST  :  [ CRC-VALUE ] BLOCK/STATEMENT/KW_DISP/EXPRESSION/FUNC_CHAR/COLON/ATTR_INT/ @125:39 {3
47892352089}
     [java] Condition :  throwException( sprintf("## unknown variable type %s/%s/%s", ref.parent.parent, ref.p
arent, ref))
     [java] Loop      :  false
     [java] --- END RULE REPORT ---
     [java]
     [java]
     [java]
     [java]     at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:1070)
     [java]     at com.goldencode.p2j.convert.TransformDriver.processTrees(TransformDriver.java:563)
     [java]     at com.goldencode.p2j.convert.ConversionDriver.back(ConversionDriver.java:572)
     [java]     at com.goldencode.p2j.convert.TransformDriver.executeJob(TransformDriver.java:906)
     [java]     at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1022)
     [java] Caused by: com.goldencode.expr.ExpressionException: Expression execution error @1:1 [ATTR_INT id=3
47892352089]
     [java]     at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:275)
     [java]     at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:210)
     [java]     at com.goldencode.p2j.pattern.PatternEngine.apply(PatternEngine.java:1633)
     [java]     at com.goldencode.p2j.pattern.PatternEngine.processAst(PatternEngine.java:1531)
     [java]     at com.goldencode.p2j.pattern.PatternEngine.processAst(PatternEngine.java:1479)
     [java]     at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:1034)
     [java]     ... 4 more
     [java] Caused by: com.goldencode.expr.ExpressionException: Expression execution error @1:1
     [java]     at com.goldencode.expr.Expression.execute(Expression.java:484)
     [java]     at com.goldencode.p2j.pattern.Rule.apply(Rule.java:497)
     [java]     at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:745)
     [java]     at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:712)
     [java]     at com.goldencode.p2j.pattern.Rule.apply(Rule.java:534)
     [java]     at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:745)
     [java]     at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:712)
     [java]     at com.goldencode.p2j.pattern.Rule.apply(Rule.java:534)
     [java]     at com.goldencode.p2j.pattern.RuleContainer.apply(RuleContainer.java:585)
     [java]     at com.goldencode.p2j.pattern.RuleSet.apply(RuleSet.java:1)
     [java]     at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:262)
     [java]     ... 9 more
     [java] Caused by: com.goldencode.expr.ExpressionException: Expression execution error @1:1
     [java]     at com.goldencode.expr.Expression.execute(Expression.java:484)
     [java]     at com.goldencode.p2j.pattern.Rule.apply(Rule.java:497)
     [java]     at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:745)
     [java]     at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:712)
     [java]     at com.goldencode.p2j.pattern.Rule.apply(Rule.java:534)
     [java]     at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:745)
     [java]     at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:712)
     [java]     at com.goldencode.p2j.pattern.Rule.apply(Rule.java:534)
     [java]     at com.goldencode.p2j.pattern.NamedFunction.execute(NamedFunction.java:450)
     [java]     at com.goldencode.p2j.pattern.AstSymbolResolver.execute(AstSymbolResolver.java:712)
     [java]     at com.goldencode.p2j.pattern.CommonAstSupport$Library.execLib(CommonAstSupport.java:1534)
     [java]     at com.goldencode.p2j.pattern.CommonAstSupport$Library.evalLib(CommonAstSupport.java:1510)
     [java]     at com.goldencode.expr.CE11579.execute(Unknown Source)
     [java]     at com.goldencode.expr.Expression.execute(Expression.java:391)
     [java]     ... 19 more
     [java] Caused by: com.goldencode.p2j.pattern.CommonAstSupport$UserGeneratedException: ## unknown variable
 type FUNC_CHAR/COLON/KW_BUFFER [ATTR_INT id <347892352089> 125:39]
     [java]     at com.goldencode.p2j.pattern.CommonAstSupport$Library.throwException(CommonAstSupport.java:28
17)
     [java]     at com.goldencode.p2j.pattern.CommonAstSupport$Library.throwException(CommonAstSupport.java:28
```

```
02)
     [java]     at com.goldencode.expr.CE11585.execute(Unknown Source)
     [java]     at com.goldencode.expr.Expression.execute(Expression.java:391)
     [java]     ... 32 more
```

Is it a regression?
Thank you.

#63 - 02/17/2020 03:25 PM - Greg Shah

No, we don't have any support for the CRC-VALUE attribute.

#64 - 02/17/2020 03:27 PM - Igor Skornyakov

Greg Shah wrote:

> No, we don't have any support for the CRC-VALUE attribute.

I see. Thank you.

#65 - 02/17/2020 03:32 PM - Greg Shah

I don't think those testcases were used in FWD. We used them to explore RAW-TRANSFER in the 4GL.

For the purposes of this task, the focus should be:

- Confirm that the method version has the same behavior as the language statement. If not, understand the differences.
- Implement method support for this in buffer, using the existing language statement implementation as makes sense. Hopefully common code can be used.

#66 - 02/17/2020 03:37 PM - Igor Skornyakov

Greg Shah wrote:

> I don't think those testcases were used in FWD. We used them to explore RAW-TRANSFER in the 4GL.
>
> For the purposes of this task, the focus should be:
>
> - Confirm that the method version has the same behavior as the language statement. If not, understand the differences.
> - Implement method support for this in buffer, using the existing language statement implementation as makes sense. Hopefully common code can be used.

I see. Thank you.

#67 - 02/18/2020 02:07 PM - Ovidiu Maxiniuc

Review for 4335a revisions 11416, 11419, and 11421.

In ConnectionManager.java:

- javadocs: I think some methods should be describing their parameters and return value in more details, especially when they are complex structures like List<List<Map.Entry<...>
- List<Map.Entry<> is a very strange structure. Why not using Map directly? Also, if I understood it right, instead of List<List<... I believe the data structure should be a map having as keys the database names (as detected in parser) and the values other maps with detected CONNECT_OPTION as keys and ConnectOptionValue as values;
- line 4696: an extra new ArrayList<>();.
- line 4726: variable not used;
- there are multiple minor deviations from the GCD coding standard. Most of them are related to white-spaces (space after keywords or javadoc sections delimitations). And there is this enum class named CONNECT_OPTION. Its instances in uppercase are fine.

In database_general.rules:

- lines 433 and 439: the backslash in """%s\'" doesn't seem right;
- the type of supConnOpts is LinkedHashMap. I believe HashMap would be sufficient here as no iteration is performed over it;
- when optionType == "i" and the text cannot be parsed as a long, you decide to make it a string instead. Is this correct?
- some options are mapped to "l" type in supConnOpts. They don't seem to be processed.

#68 - 02/18/2020 03:02 PM - Igor Skornyakov

> Review for 4335a revisions 11416, 11419, and 11421.
>
> In ConnectionManager.java:
>
> - javadocs: I think some methods should be describing their parameters and return value in more details, especially when they are complex structures like List<List<Map.Entry<...>

OK. I will think about how to make it more clear.

> - List<Map.Entry<> is a very strange structure. Why not using Map directly? Also, if I understood it right, instead of List<List<... I believe the data structure should be a map having as keys the database names (as detected in parser) and the values other maps with detected CONNECT_OPTION as keys and ConnectOptionValue as values;

The problem is that DBPARAM shows **all** CONNECT options even if some of them are specified more than once. See #3813-51. Please note also that single CONNECT statement can include options for multiple databases. This is the reason for List<List<>>.

> - line 4696: an extra new ArrayList<>();.

Thank you. Fixed.

- line 4726: variable not used;

Thank you. Fixed.

- there are multiple minor deviations from the GCD coding standard. Most of them are related to white-spaces (space after keywords or javadoc sections delimitations). And there is this enum class named CONNECT_OPTION. Its instances in uppercase are fine.

CONNECT_OPTION was renamed. Will try to fix other formatting issues. My problem is that I do not see them (in most cases).

In database_general.rules:

- lines 433 and 439: the backslash in """%s\"' doesn't seem right;

I've just retained the code from the existing implementation.

- the type of supConnOpts is LinkedHashMap. I believe HashMap would be sufficient here as no iteration is performed over it;

This is correct, thank you. Fixed

- when optionType == "i" and the text cannot be parsed as a long, you decide to make it a string instead. Is this correct?

I've just retained the code from the existing implementation.

- some options are mapped to "l" type in supConnOpts. They don't seem to be processed.

The logical options do not have values

#69 - 02/20/2020 06:04 AM - Igor Skornyakov

I've found a number of incompatibilities in the behavior of the RAW-TRANSFER statement and BUFFER:RAW-TRANSFER method between 4GL and FWD. The most important (so far) is that FWD doesn't check that source and target buffers' structure are the same.
I understand that it is related to the RAW-TRANSFER of BUFFER to RAW. I've not yet analyzed the implementation in details but the TODO comment in the BufferImpl.rawCopyTo(raw r) method states that FWD is not binary compatible with 4GL on this. Maybe it makes sense to fix it based on my experience with CLIENT-PRINCIPAL (de)serialization?

#70 - 02/20/2020 08:51 AM - Greg Shah

> TODO comment in the BufferImpl.rawCopyTo(raw r) method states that FWD is not binary compatible with 4GL on this. Maybe it makes sense to fix it based on my experience with CLIENT-PRINCIPAL (de)serialization?

Yes, please take a look at it. If you feel that you understand the solution, do fix it. Please see my previous work in #3549. If it is going to take an extended amount of time, then we will defer it.

#71 - 02/20/2020 03:17 PM - Igor Skornyakov

BUFFER-FIELD:AVAILABLE attribute is not supported in FWD (raises ** AVAILABLE is not a queryable attribute for BUFFER-FIELD widget. (4052) error) and it is a problem for BUFFER:RAW-TRANSFER support.
Is there a simple workaround?
Thank you.

#72 - 02/21/2020 04:14 AM - Igor Skornyakov

FWD behavior of the BUFFER-FIELD:BUFFER-VALUE() in the situation when there is no data record is not the same as in 4GL:
FWD reports a single message ** No target-data record is available. (91) with an error-status:error = true while 4GL reports this message with error-status:error = false **and** additional message Unable to extract BUFFER-VALUE for field a-raw. (7366).

I've also noticed another incompatibility:
LENGTH(raw) when raw is UNDEFINED returns 0 in FWD but UNKNOWN in 4GL.

#73 - 02/21/2020 05:18 AM - Igor Skornyakov

I've added support for the BUFFER-FIELD:AVAILABLE attribute which is sufficient for BUFFER:RAW-TRANSER runtime support (see #3813-71). However, handle:available() is still converted to handle:unwrapBuffer().available() which results in ** AVAILABLE is not a queryable attribute for BUFFER-FIELD widget. (4052) error.
What is the right way to fix this?
Thank you.

#74 - 02/21/2020 07:19 AM - Igor Skornyakov

I've noticed a strange effect.

Let hFieldHandle is a handle for the RAW field of a TEMP-TABLE. After RAW-TRANSFER to this table the converted expression LENGTH(hFieldHandle:BUFFER-VALUE()) returns incorrect value 18. However, if we assign LENGTH(hFieldHandle:BUFFER-VALUE()) to a RAW variable, the length of this variable has a correct value 8.

#75 - 02/21/2020 08:30 AM - Ovidiu Maxiniuc

Igor Skornyakov wrote:

> I've added support for the BUFFER-FIELD:AVAILABLE attribute which is sufficient for BUFFER:RAW-TRANSER runtime support (see #3813-71).
> However, handle:available() is still converted to handle:unwrapBuffer().available() which results in ** AVAILABLE is not a queryable attribute for BUFFER-FIELD widget. (4052) error.
> What is the right way to fix this?
> Thank you.

I see that AVAILABLE is an attribute for *Buffer object handle, Buffer-field object handle*. In FWD conversion, the kw_avail is handled in methods_attributes.rules:2588. This is why you always get an unwrapBuffer. What you need to do:

- create a new interface Available (as example see the Rejectable). Don't forget to put the LegacyAttribute annotation;
- add a method (unwrapAvailable()) in handle;
- remove the block of code at methods_attributes.rules:2588 and add a line in load_descriptors with reference to the new interface and accessor (only getter since this is a read-only attribute);
- make Buffer and BufferField implement the interface.

That should do it.

#76 - 02/21/2020 08:39 AM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> Igor Skornyakov wrote:
>
> > I've added support for the BUFFER-FIELD:AVAILABLE attribute which is sufficient for BUFFER:RAW-TRANSER runtime support (see #3813-71).
> > However, handle:available() is still converted to handle:unwrapBuffer().available() which results in ** AVAILABLE is not a queryable attribute for BUFFER-FIELD widget. (4052) error.
> > What is the right way to fix this?
> > Thank you.
>
> I see that AVAILABLE is an attribute for *Buffer object handle, Buffer-field object handle*. In FWD conversion, the kw_avail is handled in methods_attributes.rules:2588. This is why you always get an unwrapBuffer. What you need to do:
>
> - create a new interface Available (as example see the Rejectable). Don't forget to put the LegacyAttribute annotation;
> - add a method (unwrapAvailable()) in handle;
> - remove the block of code at methods_attributes.rules:2588 and add a line in load_descriptors with reference to the new interface and accessor (only getter since this is a read-only attribute);
> - make Buffer and BufferField implement the interface.
>
> That should do it.

Got it. Thank s lot!

#77 - 02/21/2020 08:44 AM - Greg Shah

Also, the new interface must be added to HandleCommon.

#78 - 02/21/2020 01:22 PM - Igor Skornyakov

Fixed the issue described in #3813-72. The problem with LENGTH(raw) is also resolved as it was a result of the fact that the BUFFER-FIELD:BUFFER-VALUE() in the absence of record have thrown exception instead of return UNLNOWN.

#79 - 02/22/2020 06:03 AM - Igor Skornyakov

The reason for the quirk described in #3813-74 is following:
The 4GL expression LENGTH(hFieldHandle:BUFFER-VALUE()) os converted to DynamicOps.length(new character(hFieldHandle.unwrapBufferField().value())). This means that we calculate not the length of the RAW value but the length of its conversion to CHARACTER.
At this moment I do not understand how this can be fixed.

#80 - 02/24/2020 03:55 PM - Igor Skornyakov

Added correct error messages and fixed incompatibilities of the RAW-TRANSFER support for use cases I could imagine. See uas/raw-transfer/buffer_raw_transfer_test.p.
Committed to 4335a revision 11432.

#81 - 02/24/2020 04:00 PM - Igor Skornyakov

I see two things regarding RAW-TRANSFER to be done

- Rework BUFFER <-> RAW (de)serialization which is binary compatible with 4GL.
- Test the exact behavior of 4GL when the RAW to BUFFER deserialization is only partially successful and adjust FWD support accordingly (should be easy).

#82 - 03/13/2020 03:45 PM - Greg Shah

Task branch 4335a was merged to trunk as revision 11345.

The ConnectionManager change was removed before it was merged since it caused a regression in the ChUI application.

#83 - 03/13/2020 05:54 PM - Igor Skornyakov

It seems that the CONNECT statement which caused the gso_241 fail is incorrect.
I've added logging and the arguments of the ConnectionManager.connect call are:

```
"<customer_logical_db_name>","from-db","127.0.0.1","<customer_db_service_name>"
```

This is an incorrect call as only one argument (db name) can be provided w/o being preceeded with -<name> key.

#84 - 03/14/2020 06:38 AM - Igor Skornyakov

Igor Skornyakov wrote:

> It seems that the CONNECT statement which caused the gso_241 fail is incorrect.
> I've added logging and the arguments of the ConnectionManager.connect call are:
> [...]
> This is an incorrect call as only one argument (db name) can be provided w/o being preceeded with -<name> key.

This is a conversion error as the 4GL code contains preceding keys:\

```
connect <customer db name> -ld value(ipLDB) -H value(t-ip) -S value(t-socket + "<customer suffix>")
```

#85 - 03/14/2020 01:42 PM - Igor Skornyakov

I've found the fix for the CONNECT conversion (see #3813-84).
To which branch should I commit it?
Thank you.

#86 - 03/15/2020 06:07 PM - Greg Shah

If it does not depend on 4335a, you can put it in 4231b. If 4335a is needed first, then wait for the 4231b rebase from trunk 11345 and then check it in.

#87 - 03/16/2020 04:04 AM - Igor Skornyakov

Greg Shah wrote:

> If it does not depend on 4335a, you can put it in 4231b. If 4335a is needed first, then wait for the 4231b rebase from trunk 11345 and then check it in.

I see. Thank you.

#88 - 03/16/2020 01:14 PM - Igor Skornyakov

A fixed CONNECT statement rework was committed to the branch 4231b revision 11368.

#89 - 03/31/2020 10:14 AM - Igor Skornyakov

Working on BUFFER-FIELD:DEFAULT-STRING and BUFFER-DECIMALS attributes support.

#90 - 04/02/2020 08:22 AM - Igor Skornyakov

Contrary to what is said in the Progress documentation the DEFAULT-STRING attribute is read-only. This is logical as the related INITIAL attribute is such.

#91 - 04/03/2020 06:33 PM - Greg Shah

By the end of the weekend, please report the list of open items remaining in this task. We can push the #3813-81 items into another task unless you see a reason they are needed now.

#92 - 04/06/2020 09:42 AM - Igor Skornyakov

We have no support for the BUFFER-FIELD:DEFAULT-VALUE (even conversion is not supported). This can be a little bit tricky at least for the DATE/DATETIME/DATETIME-TZ fields.
For example for the field FIELD s-datetime-tz AS DATETIME-TZ INITIAL NOW the value of the DEFAULT-STRING attribute is the string "now" while for INITIAL and the DEFAULT-VALUE it is a value of the NOW function at the moment of attribute retrieval.

#93 - 04/06/2020 09:50 AM - Greg Shah

Doesn't the DEFAULT-VALUE of the DMO map more closely to BUFFER-FIELD:DEFAULT-VALUE instead of using the DEFAULT-STRING in the DMO?

#94 - 04/06/2020 09:57 AM - Igor Skornyakov

Greg Shah wrote:

> Doesn't the DEFAULT-VALUE of the DMO map more closely to BUFFER-FIELD:DEFAULT-VALUE instead of using the DEFAULT-STRING in the DMO?

Sorry, I do not understand the question. In my test BUFFER-FIELD:DEFAULT-VALUE may have different values at different times. I forgot the details about the DMO map but I understand the value of BUFFER-FIELD:DEFAULT-VALUE and BUFFER-FIELD:INITIAL may have sense only at runtime and for a specific moment.

#95 - 04/06/2020 11:40 AM - Igor Skornyakov

At the moment FWD runtime support if the INITIAL attribute for fields like FIELD s-datetime AS DATETIME INITIAL NOW is not correct. A runtime exception is thrown by instantiateFromStringWorker.
We need to correctly distinguish between datetime literal and valid expression as NOW. The first question is what can be such expressions.

#96 - 04/06/2020 12:26 PM - Ovidiu Maxiniuc

I am working on 4011a branch, where the field's attributes (including the default value) are defined as DMO interface annotation. At runtime, the following code from instantiateFromStringWorker() will porcess the initial attribute of Property, just after unknown checks, but I do not recall if I added in this branch or it should already be in trunk:

```
      if ("now".equalsIgnoreCase(spec))
      {
         assign(datetimetz.now());
         return;
      }
```

#97 - 04/06/2020 12:29 PM - Igor Skornyakov

The following field definition

```
   FIELD s-datetime AS DATETIME EXTENT 2 INITIAL ["04-01-2020 13:25:30.000", NOW]
```

is converted to the code which does not compile:

```
SourceData_1_1Impl.java:20: error: SourceData_1_1Impl is not abstract and does not override abstract method se
tSDatetime(date) in SourceData_1
```

#98 - 04/06/2020 12:40 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> I am working on 4011a branch, where the field's attributes (including the default value) are defined as DMO interface annotation. At runtime, the following code from instantiateFromStringWorker() will porcess the initial attribute of Property, just after unknown checks, but I do not recall if I added in this branch or it should already be in trunk:
> [...]

Thank you, Ovidiu. I'm working with 4321b. As of rev. 11417 it doesn't contain this change.
BTW: do I understand correctly that more complicated expressions cannot be used in the INITIAL clause?
Thank you.

#99 - 04/06/2020 01:29 PM - Ovidiu Maxiniuc

> BTW: do I understand correctly that more complicated expressions cannot be used in the INITIAL clause?

TBH, before seeing your message, I was not aware that the INITIAL clauses can contain multiple values for extent values. I had the impression that you can set only one value for initialization of all items in the extent field. I tried your example and noticed that it's actually worse, the declaration for your s-datetime (from note 97) converted (with 4011a) as:

```
   @Property(id = 4, name = "SDatetime", column = "s_datetime", legacy = "s-datetime", initial = "04-01-2020"
, order = 30, extent = 2)
   public datetime getSDatetime(int index);
```

I will schedule working on this issue. 4011a is not yet stable enough, but we'll see whether we will ported to your branch. Do you need this fixed soon? What is the generated code for this field with your branch?

#100 - 04/06/2020 01:35 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> I will schedule working on this issue. 4011a is not yet stable enough, but we'll see whether we will ported to your branch. Do you need this fixed soon? What is the generated code for this field with your branch?

I do not think that it is very urgent. In my branch, the interface is generated as if there is no EXTENT clause (getSDatetime()/setSDatetime(date SDatetime)) while the implementation contains methods with index parameter (getSDatetime(int index)/setSDatetime(int index, date element)).

#101 - 04/06/2020 05:38 PM - Igor Skornyakov

I cannot find where the value of the DECIMAL option of the field definition is located in the conversion artifacts.
Can anybody help with this?
Thank you.

#102 - 04/06/2020 05:43 PM - Igor Skornyakov

Igor Skornyakov wrote:

> I cannot find where the value of the DECIMAL option of the field definition is located in the conversion artifacts.
> Can anybody help with this?
> Thank you.

Well, It seems to be a scale attribute in the .hbm.xml file. How It can be accessed programmatically?
Thank you.

#103 - 04/06/2020 07:29 PM - Eric Faulhaber

Where foo represents a DMO and bar is a decimal property, the following will return the value of DECIMAL from the field definition:

```
foo.getBar().getPrecision();
```

#104 - 04/07/2020 03:35 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Where foo represents a DMO and bar is a decimal property, the following will return the value of DECIMAL from the field definition:
>
> [...]

Got it. Thanks a lot!

#105 - 04/07/2020 04:40 AM - Igor Skornyakov

The issue described in the #3813-97 was a result of the following. I've compiled two tests. The TEMP-TABLE source-data was defined in both but in one test the field s-dataetime was defined as scalar but in another one as EXTENT 2

#106 - 04/07/2020 05:39 AM - Igor Skornyakov

Eric Faulhaber wrote:

> Where foo represents a DMO and bar is a decimal property, the following will return the value of DECIMAL from the field definition:
>
> [...]

Eric,
I understand that the getPrecision() you're talking about is a method of decimal, so it can be used only if the field has value. It is strange that the LegacyField annotation has no attribute which corresponds to the DECIMAL option of the field definition while other options do have such a counterpart. Was it done on purpose? Please clarify.
Thank you.

#107 - 04/07/2020 05:45 AM - Igor Skornyakov

Do we have methods for rendering date/numeric data using SESSION:DATE-FORMAT and SESSION:NUMERIC-FORMAT attributes' value?
Thank you.

#108 - 04/07/2020 07:36 AM - Greg Shah

Igor Skornyakov wrote:

> Do we have methods for rendering date/numeric data using SESSION:DATE-FORMAT and SESSION:NUMERIC-FORMAT attributes' value?
> Thank you.

From methods_attributes.rules:

```
                  <rule>ftype == prog.kw_date_fmt
                     <action>methodText = "SessionUtils.getDateFormat"</action>
                     <rule>isAssign
                        <action>methodText = "SessionUtils.setDateFormat"</action>
                     </rule>
                  </rule>
...
                  <rule>ftype == prog.kw_num_fmt
                     <action>methodText = "NumberType.getNumericFormat"</action>
                     <rule>isAssign
                        <action>
                           methodText = "NumberType.setNumericFormat"
                        </action>
                     </rule>
                  </rule>
```

From gaps/expressions.rules:

```
<rule>attrs.put(prog.kw_date_fmt, rw.cvt_lvl_full       | rw.rt_lvl_full)</rule>
...
<rule>attrs.put(prog.kw_num_fmt , rw.cvt_lvl_full       | rw.rt_lvl_full)</rule>
```

#109 - 04/07/2020 07:39 AM - Greg Shah

We need to correctly distinguish between datetime literal and valid expression as NOW. The first question is what can be such expressions.

In my experience, TODAY and NOW are considered "literals" even though they are really a kind a function. Thus with this classification they can be included in initializers that normally must only ever have a constant.

Please do confirm this and post the result here so that we are sure.

#110 - 04/07/2020 07:46 AM - Igor Skornyakov

Greg Shah wrote:

> We need to correctly distinguish between datetime literal and valid expression as NOW. The first question is what can be such expressions.

In my experience, TODAY and NOW are considered "literals" even though they are really a kind a function. Thus with this classification they can be included in initializers that normally must only ever have a constant.

Please do confirm this and post the result here so that we are sure.

As far as I understand this is correct. In 4GL if we specify INITIAL NOW for e.g. datetime field the value of the DEFAULT-STRING will be "now" (see #3813-92)

#111 - 04/07/2020 07:51 AM - Igor Skornyakov

Greg Shah wrote:

> Igor Skornyakov wrote:
>
> > Do we have methods for rendering date/numeric data using SESSION:DATE-FORMAT and SESSION:NUMERIC-FORMAT attributes' value?
> > Thank you.
>
> From methods_attributes.rules:
>
> [...]
>
> From gaps/expressions.rules:
>
> [...]

Thank you, Greg, but my question was about the standard method (function) of applying the values' of these attributes to formatting numeric/date values. Do we have any or I have to implement it myself? This is required for DEFAULT-STRING support.
Thank you,

#112 - 04/07/2020 08:14 AM - Greg Shah

The date, datetime, datetime-tz, integer, int64 and decimal classes all are designed to format their state honoring those configuration values. Just use those classes directly. Normally, formatted strings are rendered using toString() for the default format string and toString(String fmt) for a custom format string. Is this what you are asking?

#113 - 04/07/2020 08:23 AM - Igor Skornyakov

Greg Shah wrote:

> The date, datetime, datetime-tz, integer, int64 and decimal classes all are designed to format their state honoring those configuration values. Just use those classes directly. Normally, formatted strings are rendered using toString() for the default format string and toString(String fmt) for a custom format string. Is this what you are asking?

Greg. According to the Progress documentation

```
DEFAULT-STRING renders its dates and numeric values in LOCAL format, subject to the DATE-FORMAT and NUMERIC-FO
RMAT attributes of the SESSION system handle.
```

Do you mean that e.g. for datetime value I can use toString(SessionUtils.getDateFormat())?
Thank you.

#114 - 04/07/2020 10:45 AM - Greg Shah

> DEFAULT-STRING renders its dates and numeric values in LOCAL format, subject to the DATE-FORMAT and NUMERIC-FORMAT attributes of the SESSION system handle.

I don't know what they mean by LOCAL. And it is not clear if there is some problem here or not.

> Do you mean that e.g. for datetime value I can use toString(SessionUtils.getDateFormat())?

No. A format string is not the same thing as the DATE-FORMAT. A format string for date might by 99/99/9999 while the date-format might be dmy. As I explained above, the BDT classes internally already honor the date-format and numeric-format. You just need to call the right version of toString() to render it. Please look at the toString() implementations to see why I mean.

#115 - 04/07/2020 11:04 AM - Igor Skornyakov

Greg Shah wrote:

> No. A format string is not the same thing as the DATE-FORMAT. A format string for date might by 99/99/9999 while the date-format might be dmy. As I explained above, the BDT classes internally already honor the date-format and numeric-format. You just need to call the right version of toString() to render it. Please look at the toString() implementations to see why I mean.

As far as I can see the best match is toStringExport() for DATE/DATETIME/DATETIME-TZ. However, I do not a corresponding method for numeric (based only on group/decimal separators). It is easy to implement though.

#116 - 04/07/2020 01:33 PM - Igor Skornyakov

DEFAULT-STRING uses only decimal separator when rendering DECIMAL values. The value of the group separator is not used (groups are not separated).

#117 - 04/07/2020 02:56 PM - Eric Faulhaber

Igor Skornyakov wrote:

> Eric Faulhaber wrote:
>
> > Where foo represents a DMO and bar is a decimal property, the following will return the value of DECIMAL from the field definition:
> >
> > [...]
>
> Eric,
> I understand that the getPrecision() you're talking about is a method of decimal, so it can be used only if the field has value.

The DMO field will always return a BDT value, never null, even if it is unknown value. The precision does not change, because the getter method returns a copy of the BDT stored in the DMO. So, getPrecision() should always return the correct value.

> It is strange that the LegacyField annotation has no attribute which corresponds to the DECIMAL option of the field definition while other options do have such a counterpart. Was it done on purpose? Please clarify.

I don't think that was intentional. Please add it if you need it. My advice about getPrecision presumes you have a DMO instance to work with. If this is not the case, you'll need to use the TableMapper API.

#118 - 04/07/2020 03:09 PM - Igor Skornyakov

On the conversion of the field definition

```
FIELD s-datetime AS DATETIME INITIAL "04-01-2020 13:25:30.000"
```

the time part is lost.

#119 - 04/07/2020 03:21 PM - Igor Skornyakov

In addition to the #3813-95, I've found that current INITIAL attribute support generates exceptions for other data types (such as raw or even character) when trying to instantiate the value reflectively.
I understand that fixing this is not in the scope of this task. Is this correct?
Thank you.

#120 - 04/07/2020 03:23 PM - Igor Skornyakov

Eric Faulhaber wrote:

> I don't think that was intentional. Please add it if you need it. My advice about getPrecision presumes you have a DMO instance to work with. If this is not the case, you'll need to use the TableMapper API.

Thank you Eric. I will try to follow your advice.

#121 - 04/07/2020 03:27 PM - Greg Shah

Igor Skornyakov wrote:

> On the conversion of the field definition
> [...]
> the time part is lost.

Please show details about this.

We already expected that the TZ offset was lost **at the database** for DATETIME-TZ types (see #2156), but we did not expect the time to be lost.

#122 - 04/07/2020 03:30 PM - Greg Shah

> I've found that current INITIAL attribute support generates exceptions for other data types (such as raw or even character) when trying to instantiate the value reflectively.

This is surprising. What types are supported correctly? I'm especially surprised that character is not supported. That is the most trivial of the types for an initializer.

In regard to types like HANDLE, RAW or CLASS, it was my understanding that there is no valid way to define an initial value for those.

> I understand that fixing this is not in the scope of this task. Is this correct?

It depends on the scope of the problem.

#123 - 04/07/2020 03:31 PM - Igor Skornyakov

Another issue:
The converted statement SessionUtils.setDateFormat(new character("ydm")) causes runtime exception:

```
Caused by: java.lang.RuntimeException: Unresolvable remote export public abstract void com.goldencode.p2j.util
.SessionExports.setDateFormat(com.goldencode.p2j.util.character).
        at com.goldencode.p2j.net.RemoteObject$RemoteAccess.obtainRoutingKey(RemoteObject.java:1580)
        at com.goldencode.p2j.net.RemoteObject$RemoteAccess.invokeCore(RemoteObject.java:1464)
        at com.goldencode.p2j.net.InvocationStub.invoke(InvocationStub.java:145)
        at com.sun.proxy.$Proxy11.setDateFormat(Unknown Source)
        at com.goldencode.p2j.util.SessionUtils.setDateFormat(SessionUtils.java:557)
        at com.goldencode.testcases.db.DefaultStringTest.lambda$execute$0(DefaultStringTest.java:40)
        at com.goldencode.p2j.util.Block.body(Block.java:604)
        at com.goldencode.p2j.util.BlockManager.processBody(BlockManager.java:8527)
        at com.goldencode.p2j.util.BlockManager.topLevelBlock(BlockManager.java:8200)
        at com.goldencode.p2j.util.BlockManager.externalProcedure(BlockManager.java:480)
        at com.goldencode.p2j.util.BlockManager.externalProcedure(BlockManager.java:451)
        at com.goldencode.testcases.db.DefaultStringTest.execute(DefaultStringTest.java:29)
        at sun.reflect.GeneratedMethodAccessor10.invoke(Unknown Source)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at com.goldencode.p2j.util.Utils.invoke(Utils.java:1496)
        at com.goldencode.p2j.main.StandardServer$MainInvoker.execute(StandardServer.java:2152)
        at com.goldencode.p2j.main.StandardServer.invoke(StandardServer.java:1588)
        at com.goldencode.p2j.main.StandardServer.standardEntry(StandardServer.java:549)
        at com.goldencode.p2j.main.StandardServerMethodAccess.invoke(Unknown Source)
        at com.goldencode.p2j.util.MethodInvoker.invoke(MethodInvoker.java:156)
        at com.goldencode.p2j.net.Dispatcher.processInbound(Dispatcher.java:755)
        at com.goldencode.p2j.net.Conversation.block(Conversation.java:412)
        at com.goldencode.p2j.net.Conversation.run(Conversation.java:232)
        at java.lang.Thread.run(Thread.java:748)
Caused by: java.lang.RuntimeException: No export or no access to com.goldencode.p2j.util.SessionExports:public
 abstract void com.goldencode.p2j.util.SessionExports.setDateFormat(com.goldencode.p2j.util.character)
```

Formally this issue is not related to #3813 and does not prevent tesing.
SessionUstls.setNumericFormat and gettesr work OK.

#124 - 04/07/2020 03:37 PM - Igor Skornyakov

Greg Shah wrote:

> Igor Skornyakov wrote:
>
> > On the conversion of the field definition
> > [...]
> > the time part is lost.
>
> Please show details about this.
>
> We already expected that the TZ offset was lost **at the database** for DATETIME-TZ types (see #2156), but we did not expect the time to be lost.

I've defined a field like this:

```
FIELD s-datetime AS DATETIME INITIAL "04-01-2020 13:25:30.000"
```

But in the

```
      String property = fieldRef.getProperty();
      String init = TableMapper.getLegacyFieldInitial(getDMOProxy(), property);
```

The value of the init doesn't contain time part.

#125 - 04/07/2020 03:48 PM - Greg Shah

> Formally this issue is not related to #3813 and does not prevent tesing.

We still should probably fix this now. The core issue is not obvious to me. I don't know why we are using SessionUtils on the client for this purpose. It is a bad practice because the majority of the code in this class is only ever valid for use on the server. We really should have a separate SessionDaemon for the client which implements SessionExports and only does that job.

#126 - 04/07/2020 03:59 PM - Igor Skornyakov

Greg Shah wrote:

> > I've found that current INITIAL attribute support generates exceptions for other data types (such as raw or even character) when trying to instantiate the value reflectively.
>
> This is surprising. What types are supported correctly? I'm especially surprised that character is not supported. That is the most trivial of the types for an initializer.
>
> In regard to types like HANDLE, RAW or CLASS, it was my understanding that there is no valid way to define an initial value for those.

Well, regarding character is was probably a result of my mistake - it works. For the BinaryData, handle, rowid, and recid either constructor do not accept empty character (which is returned by TableMapper.getLegacyFieldInitial(getDMOProxy(), property) as for such types the INITIAL clause in the field definition is not allowed) or such a constructor does not exist.

> > I understand that fixing this is not in the scope of this task. Is this correct?
>
> It depends on the scope of the problem.

I've implemented preventive pre-check in the getDefaultString and do not try to vet the values for the problematic datatypes.

#127 - 04/07/2020 04:01 PM - Igor Skornyakov

Greg Shah wrote:

> Formally this issue is not related to #3813 and does not prevent tesing.

> We still should probably fix this now. The core issue is not obvious to me. I don't know why we are using SessionUtils on the client for this purpose. It is a bad practice because the majority of the code in this class is only ever valid for use on the server. We really should have a separate SessionDaemon for the client which implements SessionExports and only does that job.

OK. I will work on this.
Thank you.

#128 - 04/07/2020 04:24 PM - Greg Shah

> I've implemented preventive pre-check in the getDefaultString and do not try to vet the values for the problematic datatypes.

Most likely, the initial value for these problem cases should be the unknown value. Please check the 4GL to confirm.

#129 - 04/07/2020 04:27 PM - Igor Skornyakov

Greg Shah wrote:

> Most likely, the initial value for these problem cases should be the unknown value. Please check the 4GL to confirm.

This is correct, and FWD now returns UNKNOWN as DEFAULT-STRING without trying to get the value of the INITIAL attribute.

#130 - 04/07/2020 04:46 PM - Igor Skornyakov

The runtime support for the DEFAULT-STRING (modulo the issue with EXTENT fields (see #3813-99) is finished.
Committed to 4231b revision 11429.

Working on DECIMALS attribute support and issue #3813-123

#131 - 04/09/2020 02:45 PM - Igor Skornyakov

A SESSION:TIMEZONE = 3. (SessionUtils.setTimezone) causes the same exception as in #3813-123 (No export or no access to com.goldencode.p2j.util.SessionExports).
It is strange as the SessionUtils.setTimezone is executed at the client-side.

#132 - 04/09/2020 02:47 PM - Greg Shah

Igor Skornyakov wrote:

> A SESSION:TIMEZONE = 3. (SessionUtils.setTimezone) causes the same exception as in #3813-123 (No export or no access to com.goldencode.p2j.util.SessionExports).
> It is strange as the SessionUtils.setTimezone is executed at the client-side.

Yes. This is what I meant by my comment:

> I don't know why we are using SessionUtils on the client for this purpose. It is a bad practice because the majority of the code in this class is only ever valid for use on the server. We really should have a separate SessionDaemon for the client which implements SessionExports and only does that job.

#133 - 04/09/2020 02:55 PM - Igor Skornyakov

Greg Shah wrote:

> Yes. This is what I meant by my comment:
>
> > I don't know why we are using SessionUtils on the client for this purpose. It is a bad practice because the majority of the code in this class is only ever valid for use on the server. We really should have a separate SessionDaemon for the client which implements SessionExports and only does that job.

Maybe it will be faster to implement this rather than trying to understand what's exactly wrong with the current approach? There are only five simple methods in SessionExport. What do you think?
Thank you.

#134 - 04/09/2020 03:29 PM - Igor Skornyakov

BTW: maybe the problem is not with the SessionUtils *per se*, but in the way SessionExports is instantiated in the ThinClient - as a separate StaticNetworkServer?

#135 - 04/09/2020 03:29 PM - Greg Shah

Yes, it will certinly make it better, even if it is not the full solution.

You can pattern it using the EnvironmentDaemon (the client side of EnvironmentOps).

#136 - 04/09/2020 03:30 PM - Greg Shah

Igor Skornyakov wrote:

> BTW: maybe the problem is not with the SessionUtils *per se*, but in the way SessionExports is instantiated in the ThinClient - as a separate StaticNetworkServer?

Perhaps. Either way, implementing the proper split client daemon part will be better.

#137 - 04/09/2020 03:32 PM - Igor Skornyakov

Greg Shah wrote:

> Yes, it will certinly make it better, even if it is not the full solution.
>
> You can pattern it using the EnvironmentDaemon (the client side of EnvironmentOps).

OK, thank you.

#138 - 04/10/2020 01:11 PM - Igor Skornyakov

I understand that EnvironmentDaemon is not a suitable pattern for the situation with SESSION:DATE-FORMAT and the other methods from SessionExports as its state is not updated programmatically from the server-side.
I've found another solution - extracted setters/getters for the attributes in question to a separate class that is used both on server and client sides and delegated all the remote operations to the ThinClient/LogicalTerminal. This resolves the problem.
However, another issue was discovered after that: FWD uses SESSION:DATE-FORMAT when parsing a date literal in the INITIAL clause of the field definition and complains if they are not compatible. This is not compatible with 4GL behavior which accepts "MM-DD-YYYY" format regardless of the value of the SESSION:DATE-FORMAT.
Should I fix this new issue now?
Thank you.

#139 - 04/10/2020 01:46 PM - Greg Shah

> I understand that EnvironmentDaemon is not a suitable pattern for the situation with SESSION:DATE-FORMAT and the other methods from SessionExports as its state is not updated programmatically from the server-side.

The core structure is a good model. If you want to see storing state, you can look at FileSystemOps/FileSystemDaemon.

> I've found another solution - extracted setters/getters for the attributes in question to a separate class that is used both on server and client sides and delegated all the remote operations to the ThinClient/LogicalTerminal. This resolves the problem.

LogicalTerminal is meant to be dedicated to UI. Non-UI stuff should not be there.

ThinClient does (regrettably) have some non-UI stuff related to the client process startup/restart. But we still try to avoid non-UI stuff there as well.

Please use the separate daemon model for this case and move the stuff out of LogicalTerminal/ThinClient.

> FWD uses SESSION:DATE-FORMAT when parsing a date literal in the INITIAL clause of the field definition and complains if they are not compatible. This is not compatible with 4GL behavior which accepts "MM-DD-YYYY" format regardless of the value of the SESSION:DATE-FORMAT.

Yes, fix it.

#140 - 04/10/2020 01:57 PM - Igor Skornyakov

Greg Shah wrote:

> > I understand that EnvironmentDaemon is not a suitable pattern for the situation with SESSION:DATE-FORMAT and the other methods from SessionExports as its state is not updated programmatically from the server-side.
>
> The core structure is a good model. If you want to see storing state, you can look at FileSystemOps/FileSystemDaemon.
>
> > I've found another solution - extracted setters/getters for the attributes in question to a separate class that is used both on server and client

> sides and delegated all the remote operations to the ThinClient/LogicalTerminal. This resolves the problem.

> LogicalTerminal is meant to be dedicated to UI. Non-UI stuff should not be there.
>
> ThinClient does (regrettably) have some non-UI stuff related to the client process startup/restart. But we still try to avoid non-UI stuff there as well.
>
> Please use the separate daemon model for this case and move the stuff out of LogicalTerminal/ThinClient.

> > FWD uses SESSION:DATE-FORMAT when parsing a date literal in the INITIAL clause of the field definition and complains if they are not compatible. This is not compatible with 4GL behavior which accepts "MM-DD-YYYY" format regardless of the value of the SESSION:DATE-FORMAT.

> Yes, fix it.

OK, thank you. Please note however that, as far as I understand at this moment the only remote operations which are not LogicalTerminal/ThinClient interaction are these 5 methods from SessionExports (which do not work now). And SESSION:NUMBER-FORMAT setter propagated the change to the client-side via LogicalTerminal.

#141 - 04/10/2020 03:06 PM - Igor Skornyakov

I've removed my changes regarding LogicalTerminal. Apart from extracting SessionExports operations to a separate class, I've changed the way it is registered - instead of RemoteObject.registerStaticNetworkServer I'm using RemoteObject.registerServer like with EnvironmentDaemon. This resolves the issue with setters' propagation to the client. Working on the formatting issue from #3813-138.

#142 - 04/10/2020 03:58 PM - Igor Skornyakov

Fixed the issue with INITIAL value parsing for date fields.
Committed with SessionExport support to 4231b revision 11437.

#143 - 04/10/2020 04:38 PM - Igor Skornyakov

Eric Faulhaber wrote:

> The DMO field will always return a BDT value, never null, even if it is unknown value. The precision does not change, because the getter method returns a copy of the BDT stored in the DMO. So, getPrecision() should always return the correct value.

> It is strange that the LegacyField annotation has no attribute which corresponds to the DECIMAL option of the field definition while other options do have such a counterpart. Was it done on purpose? Please clarify.

I don't think that was intentional. Please add it if you need it. My advice about getPrecision presumes you have a DMO instance to work with. If this is not the case, you'll need to use the TableMapper API.

Eric,
I need the get the DECIMALS value from the BufferFieldImpl method I do not see how get the DMO instance. The TableMapper.getFieldValue() requires DMO interface name and it doesn't work for temporary tables.
As I wrote before, the only place in the generated artifacts where I see the value of the DECIMALS clause of the field definition is the scale attribute of the "<property column=" element in the "*Impl.hbm.xml@ files. How can I access it programmatically from the BufferFieldImpl method?
Please advise, this will save me a lot of time.
Thank you

#144 - 04/10/2020 06:00 PM - Eric Faulhaber

Igor Skornyakov wrote:

> I need the get the DECIMALS value from the BufferFieldImpl method I do not see how get the DMO instance. The TableMapper.getFieldValue() requires DMO interface name and it doesn't work for temporary tables.
> As I wrote before, the only place in the generated artifacts where I see the value of the DECIMALS clause of the field definition is the scale attribute of the "<property column=" element in the "*Impl.hbm.xml@ files. How can I access it programmatically from the BufferFieldImpl method?

Hibernate is deprecated and has been removed in branch 4011a. Do not add new dependencies on it. This will just create more work when we merge that branch to trunk.

It is possible to get a LegacyFieldInfo object for a temp-table. It is just not being done in that getFieldValue API. You need a TempTable instance, so that you can invoke the private TableMapper.locateTemp method. You get a TempTable instance from RecordBuffer.getParentTable (when invoked on a TemporaryBuffer instance). Do you have access to a TemporaryBuffer instance in your use case?

You will need to create a new TableMapper API to get what you want. See how the other APIs in that class use locateTemp with the TempTable instance that is passed to them. Model your solution on that. Once you access a LegacyFieldInfo for your temp-table, you should be able to extract the information you need from that.

#145 - 04/10/2020 06:06 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
>
> > I need the get the DECIMALS value from the BufferFieldImpl method I do not see how get the DMO instance. The TableMapper.getFieldValue() requires DMO interface name and it doesn't work for temporary tables.
> > As I wrote before, the only place in the generated artifacts where I see the value of the DECIMALS clause of the field definition is the scale attribute of the "<property column=" element in the "*Impl.hbm.xml@ files. How can I access it programmatically from the BufferFieldImpl method?
>
> Hibernate is deprecated and has been removed in branch 4011a. Do not add new dependencies on it. This will just create more work when we merge that branch to trunk.
>
> It is possible to get a LegacyFieldInfo object for a temp-table. It is just not being done in that getFieldValue API. You need a TempTable instance, so that you can invoke the private TableMapper.locateTemp method. You get a TempTable instance from RecordBuffer.getParentTable (when invoked on a TemporaryBuffer instance). Do you have access to a TemporaryBuffer instance in your use case?
>
> You will need to create a new TableMapper API to get what you want. See how the other APIs in that class use locateTemp with the TempTable instance that is passed to them. Model your solution on that. Once you access a LegacyFieldInfo for your temp-table, you should be able to extract the information you need from that.

Thank you, Eric.
I will try this approach at the weekend.

#146 - 04/11/2020 08:08 AM - Igor Skornyakov

Can anybody help me to understand how the LegacyField annotations are generated? We need at least two fixes for this:

1. Fix the INITIAL value to datetime/datetime-tz fields (at this moment only date part is generated)
2. Add the value of the DECIMALS attribute (without it the only place in the generated artifacts where the value of this attribute is present are .hbm.xml files, but we need support for the BUFFER-FIELD:DECIMALS attribute even in the absence of the underlying record and without using Hibernate).

Thank you.

#147 - 04/11/2020 02:50 PM - Ovidiu Maxiniuc

Igor, I believe both of the above attributes are members fields in the new Property DMO annotation of the new persistence framework developed in 4011.

- String initial() default ""; for the INITIAL. I noticed recently that there is a special syntax for extent field we are not properly support;
- int scale() default 10; for the DECIMALS.

Both attributes and can be directly accessed via the new DmoMetadataManager / DmoMeta classes. In this case I think working on these issues will be obsolete in a matter of weeks.
Am I right Eric?

#148 - 04/11/2020 03:03 PM - Igor Skornyakov

Ovidiu Maxiniuc wrote:

> Igor, I believe both of the above attributes are members fields in the new Property DMO annotation of the new persistence framework developed in 4011.
>
> - String initial() default ""; for the INITIAL. I noticed recently that there is a special syntax for extent field we are not properly support;
> - int scale() default 10; for the DECIMALS.
>
> Both attributes and can be directly accessed via the new DmoMetadataManager / DmoMeta classes. In this case I think working on these issues will be obsolete in a matter of weeks.
> Am I right Eric?

Ovidiu,
I'm supposed to work at least on DECIMALS support right now.

#149 - 04/11/2020 03:58 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> [...]
> Both attributes and can be directly accessed via the new DmoMetadataManager / DmoMeta classes. In this case I think working on these issues will be obsolete in a matter of weeks.
> Am I right Eric?

Yes.

#150 - 04/11/2020 04:01 PM - Eric Faulhaber

Igor Skornyakov wrote:

> I'm supposed to work at least on DECIMALS support right now.

If there is not a critical bug that is dependent upon this, then I think it makes sense to skip working on this since, as Ovidiu notes, it is already supported (albeit in a different way) in 4011a. We are trying to merge that branch to trunk ASAP, though it looks to be a number of weeks out yet.

Greg, thoughts?

#151 - 04/11/2020 04:08 PM - Eric Faulhaber

Sorry, Igor, would not have been giving you implementation advice had I realized Ovidiu already implemented this feature in 4011a.

#152 - 04/11/2020 04:12 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Sorry, Igor, would not have been giving you implementation advice had I realized Ovidiu already implemented this feature in 4011a.

I see. Thank you, Eric.

#153 - 04/12/2020 07:40 PM - Greg Shah

Eric Faulhaber wrote:

> Igor Skornyakov wrote:
>
> > I'm supposed to work at least on DECIMALS support right now.
>
> If there is not a critical bug that is dependent upon this, then I think it makes sense to skip working on this since, as Ovidiu notes, it is already supported (albeit in a different way) in 4011a. We are trying to merge that branch to trunk ASAP, though it looks to be a number of weeks out yet.
>
> Greg, thoughts?

Eric: I assume you mean that just the backing value is there in the DMO in 4011a. The rest of the support is not there yet?

Igor: Can you implement the parts of the support except for the lookup of the value? Then it can be finished when 4011a is merged to trunk.

#154 - 04/12/2020 09:20 PM - Eric Faulhaber

Greg Shah wrote:

> Eric: I assume you mean that just the backing value is there in the DMO in 4011a. The rest of the support is not there yet?
>
> Igor: Can you implement the parts of the support except for the lookup of the value? Then it can be finished when 4011a is merged to trunk.

OK, I see what you are saying. Correct, BUFFER-FIELD:DECIMALS is not exposed as an attribute at this point. In 4011a, the BufferFieldImpl implementation still looks like this:

```
   /**
    * Obtain the number of decimal places, after the decimal point, that are stored in this
    * buffer-field object that corresponds to a DECIMAL field.
    *
    * @return  the number of decimals after the decimal point for this buffer-field.
    */
   @Override
   public integer getDecimals()
   {
      UnimplementedFeature.missing("DECIMALS attribute not implemented");
      return new integer();
   }
```

#155 - 04/13/2020 03:24 AM - Igor Skornyakov

Greg Shah wrote:

> Igor: Can you implement the parts of the support except for the lookup of the value? Then it can be finished when 4011a is merged to trunk.

Greg, At this moment the only way to move forward I see is to fix LegacyField annotation generation (see #3813-143). I will resume working on it but any advice regarding generation of this annotation will help a lot.
Thank you.

#156 - 04/13/2020 05:03 AM - Constantin Asofiei

Igor, for LegacyField look in dmo_common.rules, around line 1780, for code like this:

```
         <!-- graft "fieldId" annotation -->
         <rule>
            tpl.graft('annotation_assign_num', null, fieldAst,
                      'key', 'fieldId',
                      'value', fieldId.toString())
         </rule>
```

#157 - 04/13/2020 05:05 AM - Eric Faulhaber

Igor Skornyakov wrote:

> Greg Shah wrote:
>
> > Igor: Can you implement the parts of the support except for the lookup of the value? Then it can be finished when 4011a is merged to trunk.
>
> Greg, At this moment the only way to move forward I see is to fix LegacyField annotation generation (see #3813-143). I will resume working on it but any advice regarding generation of this annotation will help a lot.
> Thank you.

No, please don't work on the conversion part. The LegacyField annotation is precisely the part of the puzzle which is made obsolete in 4011a. Please focus on the runtime portion of the solution instead.

That is, add an instance variable to TableMapper$LegacyFieldInfo for the DECIMALS attribute and add a corresponding parameter for it to that inner class' constructor.

In the call to the constructor, you can hard-code that parameter (e.g., to 10) for now, but leave a TODO in the code to lookup the actual value once branch 4011a has been merged to trunk.

Add a TableMapper API to expose this information to the persistence runtime layer, then use that API from BufferFieldImpl.getDecimals to implement the attribute, so it can be accessed by converted business logic.

#158 - 04/13/2020 05:26 AM - Igor Skornyakov

Eric Faulhaber wrote:

> No, please don't work on the conversion part. The LegacyField annotation is precisely the part of the puzzle which is made obsolete in 4011a. Please focus on the runtime portion of the solution instead.
>
> That is, add an instance variable to TableMapper$LegacyFieldInfo for the DECIMALS attribute and add a corresponding parameter for it to that inner class' constructor.
>
> In the call to the constructor, you can hard-code that parameter (e.g., to 10) for now, but leave a TODO in the code to lookup the actual value once branch 4011a has been merged to trunk.
>
> Add a TableMapper API to expose this information to the persistence runtime layer, then use that API from BufferFieldImpl.getDecimals to implement the attribute, so it can be accessed by converted business logic.

I see. Thank you, Eric.

#159 - 04/13/2020 09:11 AM - Igor Skornyakov

Implemented changes suggested by Eric (see #3813-157) and fixed dependency of the DMO instantiation on SESSION:DATE-FORMAT value (#3813-138).
Committed to 4231b rev. 11442.

#160 - 04/14/2020 03:30 PM - Igor Skornyakov

Implemented runtime support for the BUFFER:BUFFER-VALIDATE.
4231b revision 11447.

#161 - 04/14/2020 03:33 PM - Igor Skornyakov

The only remaining thing in this task is Built-In Function Usage in WHERE Clauses.
Where can I take a look at the sample support for such functions?
Thank you.

#162 - 04/14/2020 06:54 PM - Eric Faulhaber

Igor Skornyakov wrote:

> The only remaining thing in this task is Built-In Function Usage in WHERE Clauses.
> Where can I take a look at the sample support for such functions?
> Thank you.

Igor, I will open a new task for this. We will need to put together a process for identifying which built-in functions need to be supported in WHERE clauses. I'll link from here when it's ready, but it may be a day or two...

#163 - 04/14/2020 07:03 PM - Igor Skornyakov

Eric Faulhaber wrote:

> Igor, I will open a new task for this. We will need to put together a process for identifying which built-in functions need to be supported in WHERE clauses. I'll link from here when it's ready, but it may be a day or two...

I see. Thank you, Eric.

#164 - 04/15/2020 06:29 PM - Constantin Asofiei

Igor, there is an issue in BufferFieldImpl - this import "import org.aspectj.org.eclipse.jdt.core.dom.*;" is not OK, the project won't compile in Eclipse. Please don't use compile-classpath.jar, and add in Eclipse project's classpath all jars from build/lib/ only. Is your Eclipse project a gradle project?

Also, you've added lots of imports to this file, which don't make sense...

#165 - 04/15/2020 07:15 PM - Igor Skornyakov

Constantin Asofiei wrote:

> Igor, there is an issue in BufferFieldImpl - this import "import org.aspectj.org.eclipse.jdt.core.dom.*;" is not OK, the project won't compile in Eclipse. Please don't use compile-classpath.jar, and add in Eclipse project's classpath all jars from build/lib/ only. Is your Eclipse project a gradle project?
>
> Also, you've added lots of imports to this file, which don't make sense...

Sorry. I understand these imports were added automatically by Eclipse code assistant. I will check the imports more carefully in the future.
In my environment, the project compiles both with Eclipse and Gradle but the Eclipse project doesn't have a Gradle nature. It uses jars from build/lib/ as you recommended some time ago.

#166 - 04/16/2020 11:24 AM - Greg Shah

Igor: Can you please commit your import fixes so that the build is working again? I need to push this to our customers.

#167 - 04/16/2020 12:05 PM - Igor Skornyakov

Greg Shah wrote:

> Igor: Can you please commit your import fixes so that the build is working again? I need to push this to our customers.

Done in 4231b revision 11461.

#168 - 04/24/2020 11:46 AM - Greg Shah

- Related to Feature #4375: add support for -pf <profile_filename> option for CONNECT statement added

#169 - 04/24/2020 11:46 AM - Greg Shah

- Related to deleted (Feature #4392: add -pf support for appserver CONNECT() method)

#170 - 06/08/2020 07:37 AM - Greg Shah

- Related to Feature #4613: add missing support for database user defined functions (UDF) if they are supported in runtime added

#171 - 06/08/2020 07:38 AM - Greg Shah

Considering that the WHERE clause work has been moved to #4613, is there any reason to keep this task open?

#172 - 06/20/2020 11:57 AM - Greg Shah

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

> Considering that the WHERE clause work has been moved to #4613, is there any reason to keep this task open?

This question is still open.

#173 - 06/28/2020 11:40 PM - Greg Shah

Branch 3821c revision 11374 has the implementation of BUFFER-FIELD:DEFAULT-VALUE as well as the fixes to date/datetime to properly parse TODAY (only works for date) and NOW (only works for datetime and datetime-tz). These are the items noted in #3813-92 (and following notes).

#174 - 08/13/2020 08:12 AM - Greg Shah

What is left open in this task?

#175 - 08/13/2020 08:34 AM - Igor Skornyakov

Greg Shah wrote:

> What is left open in this task?

According to #3813-(162,163), the only remaining issue is Built-In Function Usage in WHERE Clauses but there is a separate task for this. I also remember that I had problems with BUFFER-FIELD:DECIMALS but now it is easy to implement with new properties of the LegacyField annotation. At least for permanent databases.

#176 - 08/13/2020 10:59 AM - Greg Shah

> According to #3813-162, #3813-163, the only remaining issue is Built-In Function Usage in WHERE Clauses but there is a separate task for this.

We will work these in #4613.

> I also remember that I had problems with BUFFER-FIELD:DECIMALS but now it is easy to implement with new properties of the LegacyField annotation. At least for permanent databases.

We have other such property changes in 3821c, right? If you finish BUFFER-FIELD:DECIMALS in 3821c, it can just be brought over to the new ORM approach during rebase. Or will this be a bigger issue to handle now rather than waiting?

#177 - 08/13/2020 11:56 AM - Igor Skornyakov

Greg Shah wrote:

> We have other such property changes in 3821c, right? If you finish BUFFER-FIELD:DECIMALS in 3821c, it can just be brought over to the new ORM approach during rebase. Or will this be a bigger issue to handle now rather than waiting?

Just rebasing will not work because the annotation is different with the new ORM. However, the required manual change is very simple.

#178 - 08/13/2020 12:04 PM - Greg Shah

I understand there will be some edits. Am I correct in my understanding that there are already other such edits that will be needed?

#179 - 08/13/2020 12:09 PM - Igor Skornyakov

Greg Shah wrote:

> I understand there will be some edits. Am I correct in my understanding that there are already other such edits that will be needed?

Ovidiu made these edits when 4011 was rebased to a trunk which contained my changes regarding metadata.

#180 - 08/13/2020 12:55 PM - Greg Shah

So there are no pending properties changes in 3821c?

#181 - 08/13/2020 01:01 PM - Igor Skornyakov

Greg Shah wrote:

> So there are no pending properties changes in 3821c?

It looks like that. Please note that it was 4 months ago and can forget some minor things.

#182 - 08/13/2020 01:05 PM - Greg Shah

OK, then lets wait until 4011b is in trunk (and 3821c rebased) before we finish this last item.

#183 - 04/12/2021 05:40 AM - Igor Skornyakov

The BUFFER-FIELD:DECIMALS support looks correct with 3821c/12264.

#184 - 04/12/2021 06:40 AM - Greg Shah

- % Done changed from 0 to 100

- Status changed from WIP to Closed

#185 - 08/11/2023 10:49 AM - Greg Shah

- Related to Feature #4392: add -pf support for appserver CONNECT() method added