Database - Feature #1668

add support for additional (non-dynamic) database attributes

10/22/2012 12:58 AM - Eric Faulhaber

Status:	Closed	**Start date:**	06/03/2013
Priority:	Normal	**Due date:**	06/28/2013
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Runtime Support for Server Features		
billable:	No	**vendor_id:**	GCD

Description

Subtasks:

Feature # 1974: add conversion support for (non-dynamic) database attributes	**Closed**
Feature # 1975: add runtime support for (non-dynamic) database attributes	**Closed**

Related issues:

Related to Database - Feature #1655: add conversion and runtime support for c...	**Closed**	**06/04/2013**	**06/11/2013**

History

#1 - 10/22/2012 12:59 AM - Eric Faulhaber

Need customer report results to make a list of needed attributes.

#2 - 10/31/2012 04:00 PM - Greg Shah

- Target version set to Milestone 7

#3 - 01/26/2013 01:53 AM - Eric Faulhaber

- Estimated time changed from 40.00 to 60.00

The following is the full list of attributes (including database and non-database) which need to be supported for our current project, in order of most frequently to least frequently used:

- handle [KW_HANDLE]
- buffer-value [KW_BUF_VAL]
- adm-data [KW_ADM_DATA]
- filename [KW_FIL_NAME]
- default-buffer-handle [KW_DEF_BUFH]
- super-procedures [KW_SUP_PROC]
- error [KW_ERROR]
- format [KW_FORMAT]
- internal-entries [KW_INT_ENT]
- name [KW_NAME]
- next-sibling [KW_NEXT_SIB]
- private-data [KW_PRIV_DAT]
- first-procedure [KW_FIRST_PR]
- is-open [KW_IS_OPEN]
- batch-mode [KW_BATCH_MO]
- persistent [KW_PERSIST]
- node-value [KW_NODE_VAL]
- available [KW_AVAIL]
- data-type [KW_DATATYPE]
- num-messages [KW_NUM_MSG]
- query-off-end [KW_QRY_OFF]
- extent [KW_EXTENT]
- num-results [KW_NUM_RES]
- num-buffers [KW_NUM_BUFF]
- undo [KW_UNDO]
- server-operating-mode [KW_SRV_OP_M]
- num-fields [KW_NUM_FLD]
- label [KW_LABEL]

- width-chars [KW_WIDTH_C]
- num-children [KW_NUM_CHLN]
- screen-value [KW_SCRN_VAL]
- table-handle [KW_TAB_HAND]
- position [KW_POS]
- remote [KW_REMOTE]
- table [KW_TABLE]
- full-pathname [KW_FULLPATH]
- rowid [KW_ROWID]
- modified [KW_MODIFIED]
- type [KW_TYPE]
- column [KW_COL]
- row [KW_ROW]
- owner-document [KW_OWN_DOC]
- title [KW_TITLE]
- file-type [KW_FIL_TYPE]
- height-chars [KW_HEIGHT_C]
- subtype [KW_SUBTYPE]
- attribute-names [KW_ATTR_NAM]
- column-label [KW_COL_LAB]
- auto-resize [KW_AUTO_RES]
- formatted [KW_FORMATTE]
- sensitive [KW_SENSITIV]
- first-child [KW_FIRST_CH]
- client-type [KW_CLNT_TYP]
- frame [KW_FRAME]
- locked [KW_LOCKED]
- parameter [KW_PARM]
- string-value [KW_STR_VAL]
- parse-status [KW_PARSE_ST]
- display-type [KW_DISP_TYP]
- frame-col [KW_FR_COL]
- has-records [KW_HAS_REC]
- multiple [KW_MULTIPLE]
- num-selected-rows [KW_NUM_SR]
- strict [KW_STRICT]
- temp-directory [KW_TEMP_DIR]
- dbname [KW_DBNAME]
- parent [KW_PARENT]
- visible [KW_VISIBLE]
- box [KW_BOX]
- date-format [KW_DATE_FMT]
- handler [KW_HANDLER]
- initial [KW_INIT]
- prepared [KW_PREPARED]
- ambiguous [KW_AMBIG]
- help [KW_HELP]
- locator-column-number [KW_LOC_C_N]
- locator-line-number [KW_LOC_L_N]
- read-only [KW_READ_ONL]
- buffer-name [KW_BUF_NAME]
- encoding [KW_ENCODING]
- hidden [KW_HIDDEN]
- index-information [KW_IDX_INFO]
- prev-sibling [KW_PREV_SIB]
- write-status [KW_WR_STAT]
- list-item-pairs [KW_LST_PAIR]
- literal-question [KW_LIT_QSTN]
- prepare-string [KW_PREP_STR]
- validate-message [KW_VAL_MSG]
- column-resizable [KW_COL_RES]
- current-result-row [KW_CUR_RES]
- dcolor [KW_DCOLOR]
- down [KW_DOWN]
- error-object-detail [KW_ERR_OBJD]
- file-mod-date [KW_FIL_M_D]
- file-size [KW_FIL_SIZE]
- frame-row [KW_FR_ROW]
- list-items [KW_LIST_ITM]
- mandatory [KW_MAND]
- num-items [KW_NUM_ITMS]
- query [KW_QUERY]
- validate-expression [KW_VAL_EXPR]
- window-system [KW_WIN_SYS]
- allow-column-searching [KW_ALLW_C_S]

- bgcolor [KW_BGCOLOR]
- buffer-field [KW_BUF_FLD]
- column-movable [KW_COL_MOV]
- column-read-only [KW_COL_R_O]
- current-column [KW_CUR_COL]
- current-iteration [KW_CUR_ITER]
- current-window [KW_CUR_WIN]
- dynamic [KW_DYNAMIC]
- expandable [KW_EXPANDBL]
- fgcolor [KW_FGCOLOR]
- first-buffer [KW_FIRST_BU]
- first-tab-item [KW_FIRST_TI]
- forward-only [KW_FWD_ONLY]
- get-bytes-available [KW_GET_B_A]
- hwnd [KW_HWND]
- keep-frame-z-order [KW_KEEP_ZOR]
- last-child [KW_LAST_CH]
- max-height [KW_MAX_HT]
- max-width [KW_MAX_WID]
- message-area [KW_MSG_AREA]
- next-tab-item [KW_NEXT_TAB]
- numeric-separator [KW_NUM_SEP]
- recid [KW_RECID]
- resize [KW_RESIZE]
- row-markers [KW_ROW_MARK]
- scroll-bars [KW_SCROLLBA]
- separators [KW_SEPS]
- soap-fault-string [KW_SOAP_F_S]
- status-area [KW_STATUS_A]
- three-d [KW_3D]
- time-source [KW_TIME_SRC]
- unique-id [KW_UNIQ_ID]
- virtual-height-chars [KW_VIRT_HC]
- virtual-width-chars [KW_VIRT_WC]
- x [KW_X]
- y [KW_Y]

I have not had the time to distinguish between database and non-database features; hence, the first task of this issue is to separate this list into database and non-database sublists. We need the list of non-database attributes for separate issues, so both sublists need to be created, not just the one for database features.

The next task is to separate the database-related sublist into attributes that represent "dynamic" database features (i.e., those that are associated with the CREATE [BUFFER | QUERY | TEMP-TABLE] language statements), and those which operate with statically defined buffers, queries, and temp-tables. Finally, AFAIR we currently only fully support one of the above database-related attributes (KW_AVAIL), though we may partially support others (i.e., the runtime support may be in place, but the related attribute does not yet convert). So, the existing runtime API should be reviewed to avoid adding redundant interfaces.

#4 - 01/28/2013 12:07 PM - Ovidiu Maxiniuc

- File isue1668.ods added

I took the list of attributes and I used the OpenEdge Reference to compile some details for them. You can group the attributes however you like by sorting the sheet accordingly.
Maybe there should be another column with the assignee. I think this is needed because otherwise we may work on the same attribute at the same time.
At this moment I cannot distinct between static and dynamic database features. In my test samples for CREATE QUERY/BUFFER/TEMP-TABLE, I needed/added conversion support for BUFFER-FIELD, CREATE-LIKE, ADD-NEW-FIELD, ADD-NEW-INDEX, ADD-LIKE-FIELD, ADD-LIKE-INDEX, TEMP-TABLE-PREPARE, DEFAULT-BUFFER-HANDLE, ADD-BUFFER, SET-BUFFERS, QUERY_PREPARE, QUERY-OPEN and QUERY_CLOSE attributes/methods. However most of them are not in the above list.

#5 - 01/28/2013 01:11 PM - Eric Faulhaber

Ovidiu, thank you for organizing this list further. The reason most of the attributes you mentioned are not in the above list is simply that they are not in use in the current customer project. But I am happy for you to implement them in order to enable your testing.

I think there may already have been overlapping effort between you and Vadim in some of the database methods, because I did not anticipate you needing these features for your CREATE QUERY/BUFFER/TEMP-TABLE testing. When I noticed this, I added you as a watcher to his issue #1947, so you could coordinate with him on those features.

I have not yet had a look at your work for the CREATE statements. Vadim is nearing the end of #1947 and we will have to rationalize your combined efforts into a single version. Please review the history for that issue and compare with your notes, then coordinate with Vadim on this.

Do you anticipate needing/implementing any further attribute support for the CREATE work? If so, please list any additional attributes you intend to implement so that Vadim can skip those, and go ahead with that work.

Vadim: please have a look at Ovidiu's spreadsheet and claim all the remaining, database-related attributes for your work with #1974. I am not so concerned with separating dynamic vs. static attributes because as Ovidiu notes, the distinction between them may be blurred. This distinction on my part was just intended to divide the work because we did not have support for dynamic database features, but Ovidiu is close with that now, so the main goal at this point is to just get all the above attributes supported in the most expedient way possible.

#6 - 01/29/2013 02:39 PM - Vadim Nebogatov

Several questions:

```
1) Conversion result of

hBuffer:AVAILABLE.
hBuffer:AVAILABLE().

is the same. Was it supposed? I understand that second line is not compilable because 4GL does not support usi
ng attribute as function.

2) Query handle based INDEX-INFORMATION Attribute. Why it is called as attribute, not method?

Applies To: Query Object Handle
SYNTAX
INDEX-INFORMATION ( n )
```

```
For example, there is (buffer handle based) method with the same name:

Applies To: Buffer Object Handle
SYNTAX
INDEX-INFORMATION(i)

3) What if we have both method (with no parameters) and attribute with equal names and we want to convert them
 differently? Or 4GL has no such methods/attributes?
```

#7 - 01/30/2013 04:21 PM - Eric Faulhaber

Vadim Nebogatov wrote:

> 1) Conversion result of
>
> hBuffer:AVAILABLE.
> hBuffer:AVAILABLE.
>
> is the same. Was it supposed? I understand that second line is not compilable because 4GL does not support using attribute as function.

If the 4GL rejects certain syntax, but we allow it, don't worry about it. We won't come up against real code written this way, so the fact that we are more lenient than Progress in this regard is a non-issue.

#8 - 01/30/2013 04:34 PM - Eric Faulhaber

Vadim Nebogatov wrote:

> 2) Query handle based INDEX-INFORMATION Attribute. Why it is called as attribute, not method?
>
> Applies To: Query Object Handle
> SYNTAX
> INDEX-INFORMATION (n)
>
> For example, there is (buffer handle based) method with the same name:
>
> Applies To: Buffer Object Handle
> SYNTAX
> INDEX-INFORMATION

It behaves like a method, even though the Progress documentation classifies it as an attribute. In progress.g, search for:

```
sym.addAttributeOrMethod( KW_IDX_INFO, ATTR_CHAR    );
```

and change it to:

```
sym.addAttributeOrMethod( KW_IDX_INFO, METH_CHAR    );
```

#9 - 01/30/2013 06:09 PM - Eric Faulhaber

Vadim Nebogatov wrote:

> 3) What if we have both method (with no parameters) and attribute with equal names and we want to convert them differently? Or 4GL has no such methods/attributes?

Actually, it seems you have found such a case with INDEX-INFORMATION. As part of the conversion, we have to emit code to unwrap the backing resource object from within a com.goldencode.p2j.util.handle instance. This is done by using one of the handle.unwrapXXX methods, each of which returns an interface representing a specific backing resource, like Buffer for buffer resources, or P2JQuery for query resources.

There are some cases where an attribute or method can be applicable to multiple backing resources, like PRIVATE-DATA and NAME. For these cases, the interface returned by the handle.unwrapXXX method must be implemented by all backing resources which support that method/attribute. In the case of INDEX-INFORMATION, this interface would be a super-interface which would be extended by both Buffer and P2JQuery; something like:

```
package com.goldencode.p2j.persist;

/**
 * Defines the API for methods and attributes which can operate on either a
 * buffer handle or a query handle. The work performed by each method and the
 * data returned, if any, may differ depending on the backing resource type.
 */
public interface CommonBufferQuery
{
   /**
    * Support for the INDEX-INFORMATION method. See the resource-specific
    * interface or implementation class for details specific to the backing
    * resource.
    *
    * @return  A comma-separated list of information about an index. The meaning
    *          of the information is specific to the type of the resource which
    *          provides it.
    */
   public character indexInformation(integer n);
}
```

It seems sort of silly to have an interface just for this specific intersection, but the alternative is to sprinkle the 4GL project with hints files which would tell us which backing resource type would go with each invocation of INDEX-INFORMATION within that project. I don't know if any more methods/attributes exist that can be invoked on both queries and buffers, but if so, they would be added to this interface (and to the classes which implement the Buffer and P2JQuery interfaces).

#10 - 01/31/2013 05:21 AM - Vadim Nebogatov

Support for EXTENT attribute should be implemented in FieldReference class. Is it correct?

```
EXTENT attribute
The number of elements in an array field.
Data type: INTEGER
Access: Readable
Applies to: Buffer-field object handle
```

#11 - 01/31/2013 09:43 AM - Ovidiu Maxiniuc

I got down to "remote" attribute, but last for the 5 of them are not finished.
For some of then I needed some help from Constantin, (see today's patch from issue #1920).

#12 - 01/31/2013 11:35 AM - Eric Faulhaber

Vadim Nebogatov wrote:

> Support for EXTENT attribute should be implemented in FieldReference class. Is it correct?

Yes. While this class was not written originally with the intention of making it a "buffer-field object" in Progress terms, it is the most appropriate class to become the backing resource for buffer-field services.

#13 - 02/01/2013 02:20 AM - Vadim Nebogatov

- File isue1668.ods added

Attached table with information about supported attributes

#14 - 02/01/2013 12:39 PM - Ovidiu Maxiniuc

- File om_upd20130201b.zip added

- File om_upd20130201a.zip added

I have merged all my changes form CREATE BUFFER/QUERY/TEMP-TABLE along with other methods/attributes and functions I worked on into a single update.
Here are the packs for both projects. Here is the list of things added (no particular order):
kw_scrn_val, kw_buf_rel, kw_buf_fld, kw_buf_val, kw_creat_lk, kw_add_newf, kw_add_newi, kw_add_likf, kw_add_liki, kw_tt_prep, kw_def_bufh, kw_def_bufh, kw_add_buf, kw_set_buf, kw_buf_rel, kw_qry_prep, kw_qry_open, kw_qry_clos, kw_adm_data, kw_def_bufh, kw_is_open, kw_node_val, kw_extent, kw_num_res, kw_undo, kw_srv_op_m, kw_num_fld, kw_num_chln, kw_tab_hand, kw_pos, kw_remote, kw_buf_crea, kw_buf_copy, kw_error and kw_handle.

#15 - 02/12/2013 08:50 AM - Vadim Nebogatov

Implemented, tested and commented attributes:
BUFFER-NAME (KW_BUF_NAME),
INDEX-INFORMATION (KW_IDX_INFO)
LITERAL-QUESTION (KW_LIT_QSTN)
VALIDATE-MESSAGE (KW_VAL_MSG).
MANDATORY (KW_MAND)
VALIDATE-EXPRESSION (KW_VAL_EXPR)
CURRENT-ITERATION (KW_CUR_ITER)
FORWARD-ONLY (KW_FWD_ONLY)
RECID (KW_RECID)
PREPARE_STRING (KW_PREP_STR)

#16 - 02/12/2013 10:59 AM - Vadim Nebogatov

I have created BrowseWidgetInterface and unwrap method, but converted code uses BrowseWidget directly:

```
fFrameFrame.widgetBrws().columnResizable();
```

Will we create the interface for BrowseWidget for using with hwrap?

#17 - 02/12/2013 11:04 AM - Greg Shah

The browse attributes and methods are UI-related, not database. So I think you don't need to be working on those.

Please post the list of the exact list of remaining attributes and methods that are database related. If there are some non-database stuff in there, we will remove them.

#18 - 02/12/2013 11:10 AM - Vadim Nebogatov

- File issue1668supportedby.ods added

I have attached spreadsheet with one more column showing already supported attributes

#19 - 02/12/2013 11:14 AM - Vadim Nebogatov

Practically all database related attributes are implemented excepting ones used in combined cases

#20 - 02/12/2013 11:38 AM - Greg Shah

Go ahead and post the update that you have. Then work with Eric to either finish any remaining attributes and methods OR to start work on something else, while Eric reviews the update.

#21 - 02/12/2013 11:01 PM - Eric Faulhaber

Based on the latest drop of customer source, the following attributes from the list in note 3 above are **no longer in use and do not need to be**

supported for Milestones 4 & 7:

- allow-column-searching [KW_ALLW_C_S]
- auto-resize [KW_AUTO_RES]
- bgcolor [KW_BGCOLOR]
- box [KW_BOX]
- buffer-field [KW_BUF_FLD]
- column-movable [KW_COL_MOV]
- column-read-only [KW_COL_R_O]
- column-resizable [KW_COL_RES]
- current-column [KW_CUR_COL]
- current-window [KW_CUR_WIN]
- date-format [KW_DATE_FMT]
- dcolor [KW_DCOLOR]
- down [KW_DOWN]
- expandable [KW_EXPANDBL]
- fgcolor [KW_FGCOLOR]
- frame-col [KW_FR_COL]
- frame-row [KW_FR_ROW]
- get-bytes-available [KW_GET_B_A]
- keep-frame-z-order [KW_KEEP_ZOR]
- last-child [KW_LAST_CH]
- max-height [KW_MAX_HT]
- max-width [KW_MAX_WID]
- message-area [KW_MSG_AREA]
- modified [KW_MODIFIED]
- prev-sibling [KW_PREV_SIB]
- query [KW_QUERY]
- read-only [KW_READ_ONL]
- resize [KW_RESIZE]
- row-markers [KW_ROW_MARK]
- screen-value [KW_SCRN_VAL]
- scroll-bars [KW_SCROLLBA]
- separators [KW_SEPS]
- status-area [KW_STATUS_A]
- three-d [KW_3D]
- virtual-height-chars [KW_VIRT_HC]
- virtual-width-chars [KW_VIRT_WC]
- x [KW_X]
- y [KW_Y]

Although most of these are not database-related, I thought this information would be useful to capture. We can copy it to other Redmine issues as needed.

Vadim: please provide a summary of all database-related attributes we still need to implement. I realize it's in the spreadsheet, but the list should be quite short at this point, and it would be a lot easier to see it directly in this history.

#22 - 02/13/2013 03:23 AM - Ovidiu Maxiniuc

Just a question.
I saw that the conversion of BUFFER-VALUE attribute (KW_BUF_VAL) declared in FieldInterface is now:

```
public Object value();
public Object value(int index);
```

I think a better return type would be BaseDataType and also, the extend form to be overloaded:

```
public BaseDataType value();
public BaseDataType value(long index);
public BaseDataType value(NumberType index);
```

Also the pairing changeValue() should take the corresponding parameter types.

#23 - 02/13/2013 10:40 AM - Vadim Nebogatov

List of database attributes still to be supported (mostly in combined cases):

name (KW_NAME)
private-data [KW_PRIV_DAT]
data-type [KW_DATATYPE]
has-records [KW_HAS_REC]
prepared [KW_PREPARED]
num-buffers [KW_NUM_BUFF] (seems buggy)
label [KW_LABEL]
width-chars [KW_WIDTH_C]
table [KW_TABLE]
dbname [KW_DBNAME]
read-only [KW_READ_ONL]
dynamic [KW_DYNAMIC]
unique-id [KW_UNIQ_ID]

#24 - 02/13/2013 11:00 AM - Eric Faulhaber

Ovidiu already implemented the following (his update is in the queue for regression testing -- see om_upd20130213a.zip in #1652):

- has-records [KW_HAS_REC]
- prepared [KW_PREPARED]
- dynamic [KW_DYNAMIC]
- unique-id [KW_UNIQ_ID]

Note that the following has dropped out, no need to support for Milestone 4 (see note 21 above):

- read-only [KW_READ_ONL]

So, only the following are left:

- name [KW_NAME] (note: there is an implementation of this in methods_attributes.rules, but I'm not sure it's right)
- private-data [KW_PRIV_DAT]
- data-type [KW_DATATYPE]
- num-buffers [KW_NUM_BUFF] (seems buggy) (I think this should map to P2JQuery.getTableCount, which already exists)
- label [KW_LABEL]
- width-chars [KW_WIDTH_C]
- table [KW_TABLE]
- dbname [KW_DBNAME]

#25 - 02/13/2013 11:09 AM - Vadim Nebogatov

- File vmn_upd20130213a.zip added

- File vmn_upd20130213a_test_cases.zip added

I have attached update containing support for

BUFFER-NAME (KW_BUF_NAME),
INDEX-INFORMATION (KW_IDX_INFO)
LITERAL-QUESTION (KW_LIT_QSTN)
VALIDATE-MESSAGE (KW_VAL_MSG).
MANDATORY (KW_MAND)
VALIDATE-EXPRESSION (KW_VAL_EXPR)
CURRENT-ITERATION (KW_CUR_ITER)
FORWARD-ONLY (KW_FWD_ONLY)
RECID (KW_RECID)
PREPARE_STRING (KW_PREP_STR)

and CommonErrorStatus.setError() fix

Update sources are based on latest code from bzr + cs_upd20130211c.zip + vmn_upd20130211a.zip

1) vmn_upd20130213a.zip - support database attributes
2) vmn_upd20130213a_test_cases.zip - associated test cases

#26 - 02/13/2013 12:19 PM - Greg Shah

Constantin: in regard to the NAME and PRIVATE-DATA attributes, in which interface are they best located? Same question for LABEL and WIDTH-CHARS when used with database instead of a widget?

#27 - 02/13/2013 12:22 PM - Greg Shah

Constantin: looking at note 24, I guess all of these remaining attributes (not just the ones in my last note) are not exclusively database related. Please help us identify the proper interfaces for these "shared" attrs.

#28 - 02/13/2013 12:55 PM - Constantin Asofiei

- name [KW_NAME] and private-data [KW_PRIV_DAT] - these two are defined by the CommonHandleChain interface. Looking at the buffer/table handles, looks like they don't support the NEXT-SIBLING and PREV-SIBLING attributes; thus, we need to move NAME and PRIVATE-DATA one level up the hierarchy, in a distinct interface, which will be implemented by CommonHandleChain and which will implement CommonHandle.
- data-type [KW_DATATYPE], table [KW_TABLE], label [KW_LABEL], dbname [KW_DBNAME]:
  - DATA-TYPE - widget, buffer-field
  - LABEL - widget, buffer-field, last-event
  - TABLE - widget, buffer-object, buffer-field
  - DBNAME - widget, buffer-object, buffer-field

  If we need to support these attributes for all these handle types, then we will need to have these interfaces:
  - Interface1: DBNAME, TABLE
  - Interface2: LABEL
  - Interface3: DATA-TYPE
- width-chars [KW_WIDTH_C], width-pixels and height-chars/height-pixels can be used with the same resource set (widget and session), so they can be put all in the same interface.

Unfortunately, this is the downside of not using dynamic proxies, we have to use interfaces to define each resource.

#29 - 02/13/2013 01:07 PM - Constantin Asofiei

> and CommonErrorStatus.setError() fix

Please share the reason/testcase you used to change setError from accepting a boolean to accepting a logical. When changing APIs in these interfaces, you need to be careful that some resources are statically implemented, thus a change in an interface API will not produce compile-time errors, but runtime errors. For this case, changing CommonErrorStat.setError not to accept boolean anymore means that ErrorManager.setError(boolean) must be changed to ErrorManager.setError(logical). But, I think the solution for your case was not to change this API, but to add another one which accepts a logical.

When adding/changing APIs from an interface, you should do a reference check throughout the project and see who is statically implementing this interface. Currently, only ErrorManager, FileSystemOps, KeyReader and SessionUtils statically implement interfaces (see the asHandle API in each of these classes for the list of interfaces).

#30 - 02/13/2013 01:16 PM - Greg Shah

I'll let Eric name Interface1.

My ideas:

Interface2 - Labelable
Interface3 - Typable

#31 - 02/13/2013 01:27 PM - Eric Faulhaber

Greg Shah wrote:

> I'll let Eric name Interface1.

Let's go with DatabaseInfo, since both DBNAME and TABLE return database-related information (name of database and name of table, respectively). Using "Info" instead of "Name" I think leaves us some flexibility if we find additional, similar methods later on that are not specifically about names.

#32 - 02/13/2013 07:47 PM - Vadim Nebogatov

Constantin Asofiei wrote

```
Comment

    and CommonErrorStatus.setError() fix

Please share the reason/testcase you used to change setError from accepting a boolean to accepting a logical.
When changing APIs in these interfaces, you need to be careful that some resources are statically implemented,
 thus a change in an interface API will not produce compile-time errors, but runtime errors. For this case, ch
anging CommonErrorStat.setError not to accept boolean anymore means that ErrorManager.setError(boolean) must b
e changed to ErrorManager.setError(logical). But, I think the solution for your case was not to change this AP
I, but to add another one which accepts a logical.

When adding/changing APIs from an interface, you should do a reference check throughout the project and see wh
o is statically implementing this interface. Currently, only ErrorManager, FileSystemOps, KeyReader and Sessio
nUtils statically implement interfaces (see the asHandle API in each of these classes for the list of interfac
es).
```

Example is in test_cases I attached along with update. Namely, the line

hBuffer:ERROR = log.

is converted to not-compilable line

handle.unwrapErrorStatus(hBuffer).setError(log);

Besides, it was strange for me that getter returns logical but setter accepts boolean parameter. So it was the reason, I decided that it was a bug.
Do you mean I only needed to add setter accepting logical parameter? What about getter returning boolean?

#33 - 02/14/2013 12:07 AM - Constantin Asofiei

Vadim: please go ahead and add setError(logical) setter (and keep setError(boolean)) in both CommonErrorStatus and ErrorManager. You don't need to worry about getter, its fine if it returns only logical.

#34 - 02/14/2013 08:29 AM - Greg Shah

There is a reason why the getters and setters for attributes/builtin functions/methods... are not symmetrical.

4GL example:

some-attribute = my-logical-var.
some-attribute = yEs.

In Java, we convert yEs (and Yes, yeS, true, tRue...) to the boolean literal true. But of course, we convert my-logical-var to a myLogicalVar instance of com.goldencode.p2j.util.logical. So we need multiple setters to handle the overloading of the possible inputs.

Of course, the getter is always returned as a logical since we often don't know when the 4GL code could safely take a boolean return. Instead, we keep it simple and always return the "wrapper type" (logical in this case).

#35 - 02/14/2013 09:51 AM - Eric Faulhaber

Vadim, Ovidiu:

Both of you included this same change to the CommonErrorStatus interface in your most recent updates, but neither of you provided classes that implement this interface in those updates. Why did you need to make this change? I understand that the ERROR attribute can be used with BUFFER and TEMP-TABLE handles, and currently, CommonErrorStatus includes {get|set}Error methods, but neither of you updated methods_attributes.rules to handle conversion of the ERROR attribute for buffers and temp-tables. Please help me understand what's going on with this change.

#36 - 02/14/2013 10:08 AM - Constantin Asofiei

Another note about ERROR attribute: as this is common to more than ERROR-HANDLE resource, it needs to be placed in an interface which groups attributes for the supported subset of these possible resources: Asynchronous request, Buffer, COMPILER, ERROR-STATUS, ProDataSet, Temp-table. All interfaces which group attributes/methods should document all the resources with which these attrs/methods can be used, in the interface's javadoc, to make it easier to find the location of other attributes/methods.

#37 - 02/14/2013 10:15 AM - Eric Faulhaber

Also note that the ERROR attribute is only writable for ProDataSet, buffer, and temp-table handles. So, it seems like we need two new interfaces: Error and WritableError. Proposed (partial) hierarchy:

```
Error (getError)
  |
  WritableError (adds setError variants)
    |
    CommonErrorStatus (adds all other ERROR-STATUS methods)
```

#38 - 02/14/2013 10:28 AM - Constantin Asofiei

Eric, that will do the trick. At some point, maybe we should put these interfaces under the same package? IMO it will be easier to search for available interfaces, but OTOH there is always the WrappedResource hierarchy.

#39 - 02/14/2013 11:42 AM - Vadim Nebogatov

Eric Faulhaber wrote:

```
Vadim, Ovidiu:

Both of you included this same change to the CommonErrorStatus interface in your most recent updates, but neit
her of you provided classes that implement this interface in those updates. Why did you need to make this chan
ge? I understand that the ERROR attribute can be used with BUFFER and TEMP-TABLE handles, and currently, Commo
nErrorStatus includes {get|set}Error methods, but neither of you updated methods_attributes.rules to handle co
nversion of the ERROR attribute for buffers and temp-tables. Please help me understand what's going on with th
is change.
```

Class name CommonErrorStatus confused me: I thought it was common interface specially created for supporting all ERROR attributes. Now I understood that it is only for ERROR-STATUS system handle

#40 - 02/14/2013 11:48 AM - Ovidiu Maxiniuc

> Class name CommonErrorStatus confused me: I thought it was common interface specially created for supporting all ERROR attributes. Now I understood that it is only for ERROR-STATUS system handle

AFAIR, I got Vadim's changes from an update pack and I only updated the parameter type to logical.

#41 - 02/14/2013 11:58 AM - Vadim Nebogatov

Additionally, hwrap = "ErrorStatus" unwrapped just for CommonErrorStatus confused in methods_attributes.rules

#42 - 02/14/2013 12:13 PM - Eric Faulhaber

OK, given that CommonErrorStatus.java appears to not be necessary for either of your updates, I am going to drop it from both and put those into regression testing. I will repost those updates to their respective Redmine issues shortly.

Vadim: please create the Error and WritableError interfaces and update CommonErrorStatus to extend WritableError as proposed above. The ERROR attribute-related methods in CommonErrorStatus should be moved to the applicable superinterfaces. Update methods_attributes such that

the rule for prog.kw_error uses an hwrap of Error if isAssign is false, else WritableError if it is true. Then, implement WritableError in BufferImpl and TempTableBuilder (you will have to merge with my reposted version of om_upd20130213a.zip, which will be available shortly in #1652 and which will contain TempTableBuilder). Please post any questions here.

Ovidiu: if there is any special implementation requirement you know of for the ERROR attribute for temp-table handles, which may be different than buffer handles, please mention it here.

#43 - 02/14/2013 12:23 PM - Ovidiu Maxiniuc

> Ovidiu: if there is any special implementation requirement you know of for the ERROR attribute for temp-table handles, which may be different than buffer handles, please mention it here.

Nothing special at this time, as I read from manual, this attribute is similar for temp-tables, buffer objects and ProDataSets.

#44 - 02/14/2013 12:31 PM - Eric Faulhaber

CORRECTION:

The partial interface hierarchy should look like this, since CommonErrorStatus should not extend WritableError:

```
Error
  |
  +- WritableError
  |
  +- CommonErrorStatus
```

Sorry for the confusion.

#45 - 02/15/2013 12:33 PM - Eric Faulhaber

- File vmn_upd20130214a.zip added

Attached is an updated version of your last code drop, merged with newer levels of Ovidiu's code, with which this update is being regression tested. As noted above, I have removed the CommonErrorStatus interface.

#46 - 02/17/2013 01:22 PM - Vadim Nebogatov

- File vmn_upd20130217a.zip added

- File vmn_upd20130217a_test_cases.zip added

I have attached update

1) vmn_upd20130217a.zip - support rest of database attributes
2) vmn_upd20130217a_test_cases.zip - associated test cases

Update contains support for attributes

ERROR (KW_ERROR)
NAME (KW_NAME)
PRIVATE-DATA (KW_PRIV_DATA)
DATA-TYPE (KW_DATATYPE)
LABEL (coKW_LABEL)
DBNAME (KW_DBNAME)
TABLE (KW_TABLE)

All attributes not always writeable, so I have created the following interfaces:

Errorable/WritableError
Nameable/WriteableName (second one was existed)
Typable/WriteableType
Labelable/WriteableLabel
PrivateData/WriteablePrivateData

I also created Sizeable/WriteableSizeable interfaces for attributes

WIDTH-CHARS (KW_WIDTH_C)
HEIGHT-CHARS (KW_HEIGHT_C)
WIDTH-PIXELS (KW_WIDTH_P)
HEIGHT-PIXELS (KW_HEIGHT_P)

and tested. Questions: 1) it seems they are not database attributes at all; 2) they are unwrapped nevertheless to Session; 3) should I continue with them? Possible setters should be added with numeric.

Fixed also bug for

CURRENT-ITERATION (not BufferField, but Buffer object)

and as discussed

FIND FIRST whatever WHERE NO-LOCK.

(issue #1947, comment #53-#58)

#47 - 02/18/2013 12:24 PM - Vadim Nebogatov

- File vmn_upd20130218a.zip added

- File vmn_upd20130218a_test_cases.zip added

I have attached merge of previous update (vmn_upd20130218a) with current bzr

1) vmn_upd20130218a.zip - support rest of database attributes
2) vmn_upd20130218a_test_cases.zip - associated test cases (actually tests were not changed)

#48 - 02/18/2013 05:49 PM - Greg Shah

There are some DB attributes still missing from the checked in code:

DBNAME
IS-OPEN
POSITION
TABLE-HANDLE
SESSION:FIRST-BUFFER (see #1612)

#49 - 02/18/2013 11:47 PM - Eric Faulhaber

- File vmn_upd20130218b.zip added

Vadim Nebogatov wrote:

> I also created Sizeable/WriteableSizeable interfaces for attributes
>
> WIDTH-CHARS (KW_WIDTH_C)
> HEIGHT-CHARS (KW_HEIGHT_C)
> WIDTH-PIXELS (KW_WIDTH_P)
> HEIGHT-PIXELS (KW_HEIGHT_P)
>
> and tested. Questions: 1) it seems they are not database attributes at all; 2) they are unwrapped nevertheless to Session; 3) should I continue with them?

According to the 4GL docs, WIDTH-CHARS can be invoked on buffer-field handles (read-only), as well as on many other UI handle types. There should be a separate interface for this (WidthCharable), which should be extended by Sizeable. WidthCharable should be implemented in FieldReference.java. The methods should be named {get|set}WidthChars (instead of {get|set}Width), to differentiate them from {get|set}WidthPixels. Note also that the getters should return integer, not int, and the setters should have 2 variants, one that accepts int and another that accepts NumberType. I have made these changes and attached the modified update. Please also note I have changed WritableError to WriteableError, to be consistent with the spelling of Writeable you used in all the other interface names.

Other than the issues noted above and some minor coding standard format issues, this update looks good. However, I would ask that you please avoid making a lot of unnecessary format changes, as these make comparing old and new versions difficult. For instance, dozens of whitespace changes were made to methods_attributes.rules which were unrelated to the functional changes.

Update vmn_20130218b.zip contains these changes and is attached. It replaces vmn_20130218a.zip.

#50 - 02/19/2013 12:36 PM - Vadim Nebogatov

- File vmn_upd20130219a_test_cases.zip added

- File vmn_upd20130219a.zip added

IS-OPEN
POSITION
TABLE-HANDLE
SESSION:FIRST-BUFFER
(DBNAME was already supported)

Update vmn_20130219a.zip contains support these attributes and is attached. It replaces vmn_20130218b.zip. Merged with current bzr and conflicts resolved. Updated tests are also attached.

Eric,

hField:WIDTH-CHARS.

is still unwrapped to Session

handle.unwrapSession(hField).getWidthChars()

#51 - 02/19/2013 12:54 PM - Eric Faulhaber

Vadim Nebogatov wrote:

> IS-OPEN
> POSITION
> TABLE-HANDLE
> SESSION:FIRST-BUFFER
> (DBNAME was already supported)
>
> Update vmn_20130219a.zip contains support these attributes and is attached. It replaces vmn_20130218b.zip. Merged with current bzr and conflicts resolved. Updated tests are also attached.

Please note that vmn_20130218b.zip is already in regression testing. Please resubmit this update, including only those files that have changed since vmn_20130218b.zip (rather than as a full replacement). We'll get this into a future round of testing.

> Eric,
>
> hField:WIDTH-CHARS.
>
> is still unwrapped to Session
>
> handle.unwrapSession(hField).getWidthChars()

I see this in methods_attributes.rules:

```
                  <rule>ftype == prog.kw_width_c
                     <action>
                        methodText = "LogicalTerminal.getWidthChars"
```

```
                    </action>

                    <rule>isAssign
                       <action>
                          methodText = "LogicalTerminal.setWidthChars"
                       </action>
                    </rule>
                    <action>uiimport = true</action>
                 </rule>
```

I wonder how we end up without the LogicalTerminal qualifier in your example?

Greg, Constantin: thoughts on how to properly support this? I added the WidthCharable (I know, lame name) interface to deal with the fact that BUFFER-FIELD handles only support the WIDTH-CHARS attribute (read-only), and none of the other WIDTH/HEIGHT-{CHARS|PIXELS} attributes. Is the SESSION form of conversion appropriate in any circumstance, or should we remove the above rule from the SESSION section?

#52 - 02/19/2013 01:10 PM - Vadim Nebogatov

- File vmn_upd20130219b_test_cases.zip added

- File vmn_upd20130219b.zip added

Update above is resubmitted

#53 - 02/19/2013 01:25 PM - Constantin Asofiei

> Greg, Constantin: thoughts on how to properly support this? I added the WidthCharable (I know, lame name) interface to deal with the fact that BUFFER-FIELD handles only support the WIDTH-CHARS attribute (read-only), and none of the other WIDTH/HEIGHT-{CHARS|PIXELS} attributes. Is the SESSION form of conversion appropriate in any circumstance, or should we remove the above rule from the SESSION section?

At the end of the SESSION section, you will see that if the attribute is used with the SESSION handle, then this is converted directly using a static method call. Else, if is used with a var handle, then it converts to a method call and the unwrapper is emitted. Idea is, is enough to specify the correct hwrap value at the attribute in this section, and if a var is used, it will unwrapp correctly. These width/height-related attributes which can be used also in SESSION should be handled in the session-specific section (or at least don't allow the SESSION-specific section to emit code if the SESSION handle is not used - see the test on line 617 which allows this section to execute for vars too). The fact that the width/height-related attributes appear too in the generic section is incorrect, they should be removed from there.

Also, note that the ERROR-STATUS specific section is broken (it doesn't default to unwrapErrorStatus for the other attributes, when variable is used).

Also #2 - I see that PRIVATE-DATA has an unwrap case of WriteablePrivateData - are there cases where PRIVATE-DATA is read-only? Because AFAIK this is a read/write attribute in all cases.

#54 - 02/19/2013 04:18 PM - Eric Faulhaber

Constantin,

We had a regression in staging for a browse column widget handle which used to do handle.unwrapWidget, but now does handle.unwrapWriteableLabel. There are a number of attributes which can be used by database-related handles (e.g., buffer field) and also by UI widgets (e.g., WIDTH-CHARS, LABEL, etc.) which as you note above, we have not dealt with correctly. The regression seems to be that WriteableLabel needs to be the superinterface of an interface on the UI side, but after looking at it for a bit, I'm not sure which one.

Since you have given this interface hierarchy a lot of thought, and are the most knowledgeable in this area, Greg and I decided the most efficient way to get over the finish line of the conversion side of this task (i.e., #1974) would be for you to complete it. As your baseline, please start with vmn_20130218a.zip, which is what we tested, overlay vmn_20130219a.zip, and take it from there. Feel free to rename WidthCharable, if a better name comes to mind.

Vadim: please move forward with #1999.

#55 - 02/19/2013 04:20 PM - Eric Faulhaber

CORRECTION: vmn_upd20130218b.zip is what we tested; it should be overlaid with vmn_upd20130219b.zip.

#56 - 02/20/2013 09:08 AM - Vadim Nebogatov

Constantin Asofiei wrote:

> Also #2 - I see that PRIVATE-DATA has an unwrap case of WriteablePrivateData - are there cases where PRIVATE-DATA is read-only? Because AFAIK this is a read/write attribute in all cases.

Yes, you are correct.

#57 - 02/20/2013 09:09 AM - Constantin Asofiei

> Yes, you are correct.

Please provide an example of read-only private-data attribute.

#58 - 02/20/2013 09:29 AM - Vadim Nebogatov

Constantin Asofiei wrote:

Yes, you are correct.

Please provide an example of read-only private-data attribute.

I wrote "you are correct": there are no such examples. At least I did not find.

#59 - 02/21/2013 11:49 AM - Constantin Asofiei

The update is added for review at #2002.

#60 - 02/26/2013 04:05 AM - Eric Faulhaber

Added conversion support for EMPTY-TEMP-TABLE and BUFFER-DELETE (see ecf_upd20130226a.zip at #1985). This update passed conversion regression testing and is committed to bzr revision 10208.

#61 - 02/27/2013 11:53 AM - Vadim Nebogatov

Discussion moved from emails.

Eric,

As far as I understood, Ovidiu is already busy with BUFFER-VALUE runtime? And I see from last updates that methods BUFFER-DELETE and EMPTY-TEMP-TABLE are also supported? What attributes from #1975 are still not in work?

Regards,
Vadim

Vadim,

Ovidiu is busy fixing conversion errors for the customer's server project conversion. I don't think he's working on any runtime attribute support. His recent conversion work with the double-colon syntax did touch this area, though.

I had to add conversion support for BUFFER-DELETE and EMPTY-TEMP-TABLE for the customer's server project conversion as well, but the runtime support for these had long been present, because they both have language statement forms that we already had to make work for our first project.

No one (beside you) currently is assigned the task of implementing runtime support for any database attributes. You will need to look at the runtime code to see is support is already there for any given attribute (i.e., like BUFFER-DELETE and EMPTY-TEMP-TABLE, could be we had to support the builtin function or language statement related to that attribute). If you find an unimplemented stub, have at it.

Thanks,
Eric

Hi Eric,
Some questions.
1. I have started from from BUFFER-VALUE, but understood that at first I should implement BUFFER-FIELD.
2. For implementation of BUFFER-FIELD we need enumerated list of getters in RecordBuffer. But now PropertyHelper contains only methods returning maps. So we need to find some way how to pass fields order to RecordBuffer constructor. Is it correct?
3. Will we use HandleField class for field handle? I see this class is not used so far.
Regards,
Vadim

Hi Vadim,
1. The order in which these are implemented doesn't matter.

2. I'm not sure how we should implement this yet. This has larger implications for schema metadata, which we need to support for this project. I have to consider the design of our metadata support first, but I won't be doing that until after we deliver Milestone 4. I suspect many of the buffer and buffer-field informational attributes will be backed by schema metadata tables (think system tables). Perhaps you can start with some attributes that are more straightforward to implement until we can address this design? If there are not enough such attributes, let me know, and I will find another issue for you to work on.
3. No, there will not be a dedicated handle type for buffer fields. Handles in Progress are not typed. The same handle can be used to hold different object types at runtime. We will just be storing a FieldReference inside an instance of handle.
Thanks,
Eric

Eric,

OK, good.

> The order in which these are implemented doesn't matter.

But testing BUFFER-VALUE seems requires BUFFER-FIELD support. I did not find how to test BUFFER-VALUE without BUFFER-FIELD.

Thanks,
Vadim

Vadim,

OK, I understand what you are saying. Still, this implementation is not the best place to start; we have to discuss how to implement the infrastructure needed for this, and I can't do that right now.

Please work first on some of the other TODOs in BufferImpl, such as bufferCreate, bufferRelease, bufferCompare, bufferCopy, findByRowID, ... Avoid the find* methods which accept a where clause predicate for now; these require additional design discussions, too.

As you work through these, bear in mind that BufferImpl is intended primarily as a thin layer which delegates work to other classes (e.g., RecordBuffer, FindQuery, etc.). To the extent practical, we want to limit the logic here to delegating the work, catching/handling errors, and managing the mismatches in return types that attributes expect and those that the worker classes provide.

Please move this discussion into Redmine.

Thanks,
Eric

#62 - 03/01/2013 01:01 AM - Constantin Asofiei

FieldReference.value is missing the value(NumberType) version (subscript can be more than int literals).

PS: and FieldInterface too

#63 - 03/04/2013 10:54 AM - Ovidiu Maxiniuc

- File om_upd20130304c.zip added

Added FieldInterface/FieldReference.value(NumberType). This is related to #2068, note 18.1.
This update must be applied AFTER om_upd20130303a and om_upd20130304b.

#64 - 03/05/2013 09:58 PM - Eric Faulhaber

Code review 20130304c:

Code changes look fine, but please correct the file header numbering for the last 2 entries in FieldReference.java. I am running this through conversion regression testing with your 20130304b update (primarily due to the valid/unknown API change in that update).

#65 - 03/05/2013 11:39 PM - Eric Faulhaber

Update 20130304c has passed conversion regression testing. Please correct the FieldReference header as noted above, then commit and distribute.

#66 - 03/08/2013 03:24 PM - Constantin Asofiei

The update doesn't handle this case:

```
def var h as handle.
message h:GET-NEXT(NO-LOCK).
```

which gets converted to:

```
message(handle.unwrapQuery(h).getNext(null));
```

and doesn't cast the null to an explicit type, thus causing the compiler to produce this error:

```
reference to getNext is ambiguous, both method getNext(LockType) in P2JQuery and method getNext(NumberType) in
 P2JQuery match
```

#67 - 03/08/2013 03:43 PM - Eric Faulhaber

Interesting...outside of the message statement, the NO_LOCK_LITERAL is the child of an intermediate EXPRESSION node, whose parent is the METH_LOGICAL (GET-NEXT) node. But, in this case, NO_LOCK_LITERAL is the direct child of the METH_LOGICAL node.

The fix will require a change to database_general.rules to handle both cases, or to change the parser to be consistent across both scenarios.

Greg: any opinion on which way to go?

#68 - 03/08/2013 03:56 PM - Constantin Asofiei

Eric, we've hit this on another occasion (? literal). For now, just handle both cases. The parser changes related to EXPRESSION nodes are not that simple (it triggers lots of TRPL code review/refactor).

#69 - 03/08/2013 04:40 PM - Eric Faulhaber

- File ecf_upd20130308a.zip added

Fix candidate attached. I'll push this through conversion regression testing.

Vadim: please take note of this update, which will conflict with your current changes for the FIND-XXXX API changes I asked you to make.

#70 - 03/10/2013 03:29 PM - Eric Faulhaber

Update ecf_upd20130308a.zip has passed conversion regression testing and is committed as bzr rev. 10274.

#71 - 03/19/2013 07:30 AM - Constantin Asofiei

Eric,

- num-buffers [KW_NUM_BUFF] (seems buggy) (I think this should map to P2JQuery.getTableCount, which already exists)

For this, I think we should add a P2JQuery.tableCount API which returns integer and will be used to mapp the num-buffers attribute. This is easier than solving all cases which expect an integer value instead of Java native int, as in:

```
def var i as int.
do i = 1 to h:num-buffers transaction:
end.
```

#72 - 03/19/2013 01:11 PM - Eric Faulhaber

Constantin Asofiei wrote:

> For this, I think we should add a P2JQuery.tableCount API which returns integer and will be used to mapp the num-buffers attribute.

Let's name it P2JQuery.numBuffers to make it more recognizable to customers. The implementation can call getTableCount under the covers.

Vadim: as your top priority, please add this new method to P2JQuery and to the necessary implementation classes, and submit it as a separate update here.

#73 - 03/20/2013 12:37 PM - Costin Savin

For:

```
[javac] .../server/common/Agifcfn0.java:903: error: no suitable method found for addFieldToIndex(character,cha
racter)
```

and

```
[javac] .../server/common/Apictrl0.java:2444: error: no suitable method found for addFieldToIndex(String,chara
cter,character)
```

there are only these methods defined in the TempTable interface:
public logical addFieldToIndex(String indexName, String field, boolean ascending)
public logical addFieldToIndex(String indexName, String field)

In Progress 4gl the ADD-INDEX-FIELD function is defined as having 2 mandatory character parameters and a 3rd one which is also character, not logical and can be one of 2 "asc" and "desc"

What would you prefer as solution?:

- The 3rd parameter put as character/string, or work in the rules to emit it as boolean literal depending on it's value (this might not be great if we have an error to display in case we have something different than "asc" / "desc" as the last parameter)

- Overload methods to cover all the String / character combinations, or only have character parameter functions and work with rules to wrap them in a new character()?

Also the addFieldToIndex methods are not added to implementation in TempTableBuilder abstract class which implements the TempTable interface, should I add a stub for them there?

LE: GES removed customer package paths from this entry.

#74 - 03/20/2013 01:07 PM - Eric Faulhaber

Costin Savin wrote:

> there are only these methods defined in the TempTable interface:
> public logical addFieldToIndex(String indexName, String field, boolean ascending)
> public logical addFieldToIndex(String indexName, String field)
>
> In Progress 4gl the ADD-INDEX-FIELD function is defined as having 2 mandatory character parameters and a 3rd one which is also character, not logical and can be one of 2 "asc" and "desc"
>
> What would you prefer as solution?:
>
> - The 3rd parameter put as character/string, or work in the rules to emit it as boolean literal depending on it's value (this might not be great if we have an error to display in case we have something different than "asc" / "desc" as the last parameter)

Write a test case and see what happens if you pass in something other than "asc" or "desc" as the last parameter. Do you get an error? Is it a compile error or a runtime error? If it is a compile error, keep the last parameter as a boolean and change the conversion rules to emit a true/false literal. If it is a runtime error, change the API to accept a String or character; we'll have to do error checking at runtime in this case.

> - Overload methods to cover all the String / character combinations, or only have character parameter functions and work with rules to wrap them in a new character()?

Since there are only a small number of parameters and we already know there are mixed uses of parameter types in the current project, please add API variants to support all combinations. Note that if you determine with your test case that the last parameter can remain a boolean, we don't need a logical variant of this, since we will only ever be emitting true/false literals for this parameter during conversion.

> Also the addFieldToIndex methods are not added to implementation in TempTableBuilder abstract class which implements the TempTable interface, should I add a stub for them there?

That's strange, all the TempTable methods should be implemented in TempTableBuilder, AFAIK. I don't know why they are missing.

Ovidiu: why is TempTableBuilder an abstract class?

#75 - 03/20/2013 02:05 PM - Costin Savin

> Write a test case and see what happens if you pass in something other than "asc" or "desc" as the last parameter. Do you get an error? Is it a compile error or a runtime error? If it is a compile error, keep the last parameter as a boolean and change the conversion rules to emit a true/false literal. If it is a runtime error, change the API to accept a String or character;

Tested this case, The result is a runtime error message: "Third argument to ADD-INDEX-FIELD must be DESC or ASC in quotes. (9062)"

#76 - 03/20/2013 02:07 PM - Eric Faulhaber

Costin Savin wrote:

> Tested this case, The result is a runtime error message: "Third argument to ADD-INDEX-FIELD must be DESC or ASC in quotes. (9062)"

Please post your test case.

#77 - 03/20/2013 05:38 PM - Vadim Nebogatov

- File vmn_upd20130320a.zip added

- File vmn_upd20130320b.zip added

Update vmn_upd20130320a.zip (vmn_upd20130320b.zip - conversion test) contains correction of NUM-BUFFERS(KW_NUM_BUFF) support according #1668, notes 71-72
Merged with revision 10302

#78 - 03/20/2013 06:05 PM - Eric Faulhaber

Code review 20130320a:

All looks good except QueryWrapper: the new numBuffers method implementation should delegate work to the internal delegate object, as in return getDelegate().numBuffers();. Also, the new method should be grouped with all the other ones, above the DefaultDelegate inner class, rather than being alone at the bottom of the class.

Ovidiu has an update (om_upd20130320a.zip) which affects all of these same files, which I have asked him to check in. Once you receive a notification from him that his update is in bzr, please merge your changes (and those requested above) with that revision and resubmit for conversion regression testing. I don't expect any problems, since your changes were straightforward.

#79 - 03/21/2013 06:55 AM - Costin Savin

Please post your test case.

I used the pers-addr table from p2j_test and created a temp table like pers-addr and added the index active.

```
DEFINE VARIABLE tth AS HANDLE.
DEFINE VARIABLE pers-addrH AS HANDLE.
pers-addrH = BUFFER pers-addr:HANDLE.
CREATE TEMP-TABLE tth.
tth:CREATE-LIKE(pers-addrH).
tth:ADD-INDEX-FIELD("active", "active", "bla").
```

#80 - 03/21/2013 09:17 AM - Ovidiu Maxiniuc

That's strange, all the TempTable methods should be implemented in TempTableBuilder, AFAIK. I don't know why they are missing.

Ovidiu: why is TempTableBuilder an abstract class?

I intended that TempTableBuilder to act as a builder which constructs some other class instances based on atomic primitives like ADD-NEW-FIELD, ADD-NEW-INDEX. At that time I wanted it to be a node in class hierarchy with some other sub-classes implementing the actual behavior. But now I think I over-planned the hierarchy and the class to stay "normal" (not abstract any more) with all needed functionality implemented here.

#81 - 03/21/2013 10:59 AM - Vadim Nebogatov

- File vmn_upd20130321a.zip added

- File vmn_upd20130321b.zip added

Update vmn_upd20130321a.zip (vmn_upd20130321b.zip - conversion test) is replacement of vmn_upd20130320a.zip (vmn_upd20130320b.zip) with fixes according note 78
Merged with revision 10306

#82 - 03/21/2013 11:36 AM - Eric Faulhaber

Code review 20130321a:

Looks good, I'm putting it through conversion regression test now.

#83 - 03/21/2013 11:50 AM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> That's strange, all the TempTable methods should be implemented in TempTableBuilder, AFAIK. I don't know why they are missing.
>
> Ovidiu: why is TempTableBuilder an abstract class?

I intended that TempTableBuilder to act as a builder which constructs some other class instances based on atomic primitives like ADD-NEW-FIELD, ADD-NEW-INDEX. At that time I wanted it to be a node in class hierarchy with some other sub-classes implementing the actual behavior. But now I think I over-planned the hierarchy and the class to stay "normal" (not abstract any more) with all needed functionality implemented here.

Costin: please remove the abstract modifier from the TempTableBuilder class definition and add implementation stubs for any missing methods from the TempTable interface. Also, go ahead with the approach discussed in note 74 above to change the last parameter of addFieldToIndex from boolean to String/character. Please rename the parameter from ascending to direction.

#84 - 03/21/2013 12:55 PM - Eric Faulhaber

Vadim, your 20130321a update has passed conversion regression testing. Please check it in (and the test case) and distribute.

#85 - 03/21/2013 01:25 PM - Costin Savin

> That's strange, all the TempTable methods should be implemented in TempTableBuilder, AFAIK. I don't know why they are missing.

In conclusion do I make TempTableBuilder a public non-abstract class and add the 127 unimplemnted methods?(for addFieldToIndex I added the stubs) or do I leave it abstract for now and put a TODO to add stubs?

#86 - 03/21/2013 01:30 PM - Eric Faulhaber

Costin Savin wrote:

> That's strange, all the TempTable methods should be implemented in TempTableBuilder, AFAIK. I don't know why they are missing.

> In conclusion do I make TempTableBuilder a public non-abstract class and add the 127 unimplemnted methods?(for addFieldToIndex I added the stubs) or do I leave it abstract for now and put a TODO to add stubs?

Sorry, where does the 127 number come from?

#87 - 03/21/2013 01:31 PM - Constantin Asofiei

Eric/Ovidiu/Costin: the interfaces extended by TempTable don't look OK. In the docs, the temp-table resource can access only these methods:

```
ADD-FIELDS-FROM( ) method
ADD-INDEX-FIELD( ) method
ADD-LIKE-FIELD( )
method ADD-LIKE-INDEX( ) method
ADD-NEW-FIELD( ) method
ADD-NEW-INDEX( ) method
CLEAR( ) method
COPY-TEMP-TABLE( ) method
CREATE-LIKE( )
method READ-XML( ) method
READ-XMLSCHEMA( ) method
TEMP-TABLE-PREPARE( ) method
WRITE-XML( ) method
WRITE-XMLSCHEMA( ) method
```

Why is TempTable interface extending Buffer, ADMData and UniqueID interfaces?

#88 - 03/21/2013 01:38 PM - Constantin Asofiei

ADMData was my mistake (should have not added it, as I don't see the ADM-DATA attribute defined for the temp-table resource; I think I mistook TempTable with something neeed by temporary records). The UniqueId interface is OK. But the Buffer interface bugs me... at most, TempTable should extend the interfaces which define the needed attributes, and not the entire Buffer interface.

#89 - 03/21/2013 01:39 PM - Eric Faulhaber

Constantin Asofiei wrote:

> Why is TempTable interface extending Buffer, ADMData and UniqueID interfaces?

Good question. I don't think it needs to extend Buffer. You've already covered the other two.

#90 - 03/21/2013 01:42 PM - Eric Faulhaber

So, Costin: please remove Buffer and ADMData from the TempTable interface and post here what method stubs are still missing from TempTableBuilder, then add them.

#91 - 03/21/2013 01:49 PM - Costin Savin

> So, Costin: please remove Buffer and ADMData from the TempTable interface and post here what method stubs are still missing from TempTableBuilder, then add them.

After this change the following methods need to be added:

```
TempTable.canUndo()
TempTable.copyTempTable(handle, logical, logical, logical, character)
UniqueID.getUniqueID()
TempTable.addFieldsFrom(String)
TempTable.copyTempTable(handle, boolean, boolean, boolean, String)
TempTable.prepared()
TempTable.hasRecords()
TempTable.setCanUndo(logical)
HandleChain.resourceDelete()
TempTable.dynamic()
TempTable.createTableLike(String)
TempTable.changeErrorString(character)
WrappedResource.valid()
TempTable.uniqueId()
TempTable.errorString()
TempTable.addFieldsFrom(String, String...)
TempTable.clear()
TempTable.addFieldsFrom(handle, String...)
TempTable.addFieldsFrom(handle)
```

I'll go ahead and add them

#92 - 03/21/2013 01:54 PM - Constantin Asofiei

Eric/Costin: there is one more thing we should do - identify all attributes which are already defined by interfaces (this is easy, go to HandleCommon interface and search up the hierarchy for the attribute). The attributes to be searched are the one in the documentation for the temp-table resource. We need a table of the temp-table-related attributes and the interfaces which define them. If the interface has some other attributes beside those needed by TEMP-TABLE, mark that interface. If no interface defines it, mark the attribute.

Last, make sure that TempTableBuilder doesn't have stubs for Buffer-related APIs; if found, make a list and post it here.

#93 - 03/21/2013 03:54 PM - Costin Savin

From the TEMP-TABLE attributes the following are not implemented in any interface:

AFTER-TABLE, BEFORE-TABLE, DATA-SOURCE-MODIFIED , INSTANTIATING-PROCEDURE, MIN-SCHEMA-MARSHAL, NAMESPACE-PREFIX , NUM-REFERENCES , NO-SCHEMA-MARSHAL, ORIGIN-HANDLE, PRIMARY, REJECTED, SCHEMA-MARSHAL, SERIALIZE-NAME , TRACKING-CHANGES, XML-NODE-NAME

and the following exist and were not added to temp-table:
ADM-DATA (in ADMData interface), DEFAULT-BUFFER-HANDLE

#94 - 03/22/2013 04:35 AM - Constantin Asofiei

> ADMData was my mistake (should have not added it, as I don't see the ADM-DATA attribute defined for the temp-table resource; I think I mistook TempTable with something neeed by temporary records).

I was looking at old docs, which didn't have ADM-DATA in the list of attributes. In latest docs they are mentioned.

#95 - 03/22/2013 05:06 AM - Ovidiu Maxiniuc

This class evolved a little. The defaultBufferHandle() (implementation of DEFAULT-BUFFER-HANDLE) was at the beginning part of TempTable, but then it was moved (#1652, notes 26-29) to Buffer interface.
The other missing attributes were not added because they were not included in the list from note 3.

#96 - 03/22/2013 08:21 AM - Costin Savin

- File cs_upd20130322b.zip added

Added proposed update for missing addFieldToIndex methods. Moved the defaultBufferHandle() method form the Buffer interface to a new BufferHandle interface and added it to Buffer and TempTable

#97 - 03/22/2013 10:54 AM - Eric Faulhaber

cs_upd20130322b.zip looks good. I am putting it through conversion regression testing.

It does look like we still have a number of APIs that are not well-covered in terms of the possible permutations of parameter types (e.g., copyTempTable, so we will have to do another pass of adding methods after this is tested and checked in. I think the general rule of thumb should be that if we have 3 or fewer parameters, we provide all permutations and if there are more, we coerce wrapped literals.

#98 - 03/22/2013 01:07 PM - Eric Faulhaber

cs_upd20130322b.zip has passed conversion regression testing. Please check it in and distribute.

#99 - 03/22/2013 01:43 PM - Costin Savin

Commited as revision number 10317

#100 - 03/22/2013 02:12 PM - Constantin Asofiei

Costin, regarding cs_upd20130322b.zip - you forgot to add the new interface to HandleCommon. Please put it right after Connectable, in the list, to keep it sorted.

#101 - 03/22/2013 02:32 PM - Costin Savin

Well theoreticaly it is already there because both Buffer and TempTable extend it, but probably it's good to see it in the list of interfaces.

#102 - 03/22/2013 02:32 PM - Constantin Asofiei

Costin Savin wrote:

> Well theoreticaly it is already there because both Buffer and TempTable extend it, but probably it's good to see it in the list of interfaces.

Ah, OK then, no change needed.

#103 - 03/23/2013 10:23 AM - Constantin Asofiei

- File ca_upd20130323a.zip added

This update changes the API for BUFFER-DELETE to deleteRecord, because else will collide with the resource's delete() API. More, made the APIs for BUFFER-DELETE and EMPTY-TEMP-TABLE to return logical.

LE: for note 226 in #2068.

#104 - 03/23/2013 11:38 AM - Constantin Asofiei

Eric: the ca_upd20130323a.zip has regressions. I see that the Buffer.delete API is used by other cases, like DELETE statement and DELETE FROM embedded sql statement. So, considering that the Buffer.delete API collides with the Deletable.delete API (which is needed to delete the actual resource), how do you suggest to proceed? Should I make the other "delete" API cases to convert to "deleteRecord"?

#105 - 03/23/2013 12:19 PM - Eric Faulhaber

Let's use:

```
public logical deleteRecord()
public logical deleteRecord(BufferValidator)
```

I don't think the embedded SQL case of DELETE FROM is affected by these changes and can remain:

```
public void delete(String, Object...)
```

#106 - 03/23/2013 12:23 PM - Constantin Asofiei

For embedded SQL, I was referring to this rule in convert/database_access:

```
      <!-- emit a delete or deleteAll method call for an embedded SQL DELETE FROM statement -->
      <rule>type == prog.delete_from and parent.type == prog.embedded_sql
         <rule>this.getImmediateChild(prog.kw_where, null) == null
            <action>createPeerAst(java.method_call, "deleteAll", closestPeerId)</action>
            <action on="false">createPeerAst(java.method_call, "delete", closestPeerId)</action>
         </rule>
      </rule>
```

Which class implements the delete call emitted by this rule?

#107 - 03/23/2013 12:30 PM - Eric Faulhaber

This refers to:

```
Buffer[Impl].delete(String where, Object ...args)
```

There will always be at least a String argument to this call representing the WHERE clause, and optionally some substitution parameters, otherwise we convert to deleteAll. So, it should never conflict with Deletable.delete(), since the signatures are different. Am I missing something?

#108 - 03/23/2013 12:33 PM - Constantin Asofiei

You are not missing anything, this is the first time I touch embedded sqls, so that's why the questions. Now I have all the info to make the changes.

#109 - 03/23/2013 12:52 PM - Constantin Asofiei

- File ca_upd20130323d.zip added

Replaces ca_upd2013023a.zip, and has the changes from note 105.

#110 - 03/23/2013 12:58 PM - Eric Faulhaber

ca_upd20130323d.zip update looks good to me. If it passes conversion regression testing (expect some changes to Majic), please check it in and distribute.

#111 - 03/23/2013 06:54 PM - Constantin Asofiei

- File ca_upd20130323h.zip added

This update has passed conversion regression testing (CA 0323d used deleteRecord for the "quick_delete" loops, instead of "delete"). MAJIC changes are in #2102. Committed to bzr revision 10324

#112 - 04/17/2013 06:21 PM - Eric Faulhaber

- File ecf_upd20130417a.zip added

The attached update fixes conversion of the INITIAL buffer field attribute (was converting to operate on BUFFER handle instead of BUFFER-FIELD handle). It has passed conversion regression testing and is committed to bzr rev. 10340.

#113 - 04/25/2013 11:55 AM - Eric Faulhaber

The following are in scope for this task.

Buffer attributes:

- currentIteration
- is/setError
- get/setADMData
- getDbName
- getNamespaceURI
- getTable (requires metadata)
- getUniqueID
- numFields
- tableHandle

TempTable attributes:

- canUndo/setCanUndo
- errorString/changeErrorString
- defaultBufferHandle
- dynamic
- get/setADMData
- getUniqueID
- hasRecords
- is/setError
- prepared
- uniqueID (redundant with getUniqueID; should be removed)

#114 - 08/15/2013 02:14 PM - Eric Faulhaber

Please review those attributes in this task which require metadata support in order to implement. This includes any schema-level information stored in the _file, _field, _index, _index-field, _lock, or _user tables.

This information will be provided via the MetaSchema class, but in order to provide the appropriate methods in this class, I need to know the type of information needed and the context in which it is needed. This will help me shape the API and to make the most efficient implementation to retrieve this metadata.

For example, the implementation of the buffer attribute numFields requires the number of fields in a particular table, which is something we can get from the _file metadata table. So, we will need a MetaSchema API that provides this information. Perhaps something like:

```
public int numFields(String table)
```

These methods likely will throw an exception if the data cannot be found (let me know if you have a better idea to make this API more convenient -- we can do what we want here, it is just for internal use). But, the first step is to understand what information is needed.

#115 - 08/16/2013 09:56 AM - Vadim Nebogatov

Is currentIteration attribute planned for ProDataSet.FILL() support? In this case seems Buffer should refer to parent ProDataSet object which should contain mapping buffers to currentIteration values.

#116 - 08/16/2013 09:59 AM - Eric Faulhaber

ProDataSet support is out of scope for the current project.

#117 - 08/16/2013 11:33 AM - Vadim Nebogatov

But according to documentation, integer CURRENT-ITERATION attribute is used only for using buffers related with ProDataSet: "Indicates which iteration level corresponds to the buffer handle during a recursive FILL of a ProDataSet"

#118 - 08/16/2013 12:20 PM - Eric Faulhaber

I checked how this attribute is used in the project. There is only one use, and it is the widget version of the attribute, so it turns out CURRENT-ITERATION (for data objects) is out of scope for this issue after all.

#119 - 08/20/2013 05:06 PM - Vadim Nebogatov

I think only 2 methods are really needed from MetaSchema for attributes enumerated in note 113.

```
   /**
    * Get 4GL name for P2J table.
    *
    * Needed information should be provided in _file as separate column, i.e. _native_name.
    *
    * @param   database
    *          Database in which table resides.
    * @param   dmoClass
    *          DMO implementation class to which the table is mapped via ORM.
```

```
     *
     * @return  4GL name for P2J table.
     */
    public String getTable(Database database, Class<?> dmoClass);

    /**
     * Get number of fields.
     *
     * Needed information should be provided in _file as separate column, i.e. _num_fields.
     *
     * @param   database
     *          Database in which table resides.
     * @param   table
     *          Table name.
     *
     * @return  number of fields.
     */
    public int numFields(Database database, String table);
```

I suggest to create special MetaSchemaException which will be thrown with all methods of MetaSchema. Processing of such exception: actually, if such error is thrown, application should be stopped and corresponding exception message is logged.

#120 - 08/20/2013 05:34 PM - Vadim Nebogatov

Should be implemented only attributes from note 113, or all attributes from note 3?

#121 - 08/20/2013 05:39 PM - Eric Faulhaber

From note 113.

#122 - 08/21/2013 05:24 PM - Vadim Nebogatov

Buffer attributes status:

currentIteration – removed;

is/setError, get/setADMData, getNamespaceURI, getUniqueID – implemented using private member fields in RecordBuffer. uniqueID calculation will reuse Stat's code;

getDbName implemented with using RecordBuffer.getLogicalDatabase();

getTable, numFields, tableHandle require MetaSchema;

#123 - 08/29/2013 05:59 PM - Vadim Nebogatov

- File vmn_upd20130829a.zip added

- File vmn_upd20130829b.zip added

Update vmn_upd20130829a.zip/vmn_upd20130829b.zip is attached.

Test case corresponds to 4GL behavior for tablehandle (the same tablehandle is returned for all created buffers).

Merged with latest revision 10376.

#124 - 08/30/2013 11:58 AM - Eric Faulhaber

For all the buffer-related attributes in note 113, please determine with test cases that these all work the same way for temp-tables as they do for permanent tables in Progress. I expect they do, but we need to confirm this. The reason they might be different is that the Progress metaschema tables do not provide information for temp-tables, only for permanent tables.

So, for any buffer attributes (e.g., numFields and getTable) which require MetaSchema backing to provide information for temp-tables, we will have to develop a different MetaSchema implementation to provide that information for temp-tables than we will have for permanent tables (which will use the metadata available for permanent schemas).

#125 - 09/02/2013 05:15 PM - Vadim Nebogatov

From docs: "For ... Buffer object handle, the ERROR attribute indicates whether an error occurred during a FILL or SAVE-ROW-CHANGES operation."

Both operations are related with ProDataSet project not supported currently.

#126 - 09/02/2013 05:31 PM - Vadim Nebogatov

The following test case for note 124:

```
DEFINE VARIABLE pBuf as HANDLE.
DEFINE BUFFER pBuffer FOR book.
pBuf = BUFFER pBuffer:HANDLE.

DEFINE TEMP-TABLE temp like book.
DEFINE VARIABLE tBuf as HANDLE.
DEFINE BUFFER tBuffer FOR temp.
tBuf = BUFFER tBuffer:HANDLE.

message "permanent ERROR:" pBuf:ERROR.
message "temporary ERROR:" tBuf:ERROR.

pBuf:ADM-DATA = "test ADM-DATA".
tBuf:ADM-DATA = "test ADM-DATA".

message "permanent ADM-DATA:" pBuf:ADM-DATA.
message "temporary ADM-DATA:" tBuf:ADM-DATA.

message "permanent UNIQUE-ID:" pBuf:UNIQUE-ID.
message "temporary UNIQUE-ID:" tBuf:UNIQUE-ID.

message "permanent TABLE-HANDLE:" pBuf:TABLE-HANDLE.
message "temporary TABLE-HANDLE:" tBuf:TABLE-HANDLE.

message "permanent DBNAME:" pBuf:DBNAME.
message "temporary DBNAME:" tBuf:DBNAME.
```

```
message "permanent NAMESPACE-URI:" pBuf:NAMESPACE-URI.
message "temporary NAMESPACE-URI:" tBuf:NAMESPACE-URI.

message "permanent TABLE:" pBuf:TABLE.
message "temporary TABLE:" tBuf:TABLE.

message "permanent NUM-FIELDS:" pBuf:NUM-FIELDS.
message "temporary NUM-FIELDS:" tBuf:NUM-FIELDS.
```

has test results:

```
permanent ERROR: no
temporary ERROR: no

permanent ADM-DATA: test ADM-DATA
temporary ADM-DATA: test ADM-DATA

permanent UNIQUE-ID: 84
temporary UNIQUE-ID: 86

permanent TABLE-HANDLE: ?
temporary TABLE-HANDLE: 1105

permanent DBNAME: p2j_test
temporary DBNAME: PROGRESST

permanent NAMESPACE-URI: ?
temporary NAMESPACE-URI: ?

permanent TABLE: Book
temporary TABLE: temp

permanent NUM-FIELDS: 10
temporary NUM-FIELDS: 10
```

Behavior for permanent and temporary tables differs only for TABLE-HANDLE as expected.

#127 - 09/03/2013 12:27 PM - Eric Faulhaber

Interesting temp-table result for DBNAME, too. Suggests that internally, Progress uses a distinct database for temp-tables.

#128 - 01/07/2014 12:12 PM - Eric Faulhaber

- Status changed from New to Closed

See #1655.

#129 - 11/16/2016 11:42 AM - Greg Shah

- Target version changed from Milestone 7 to Runtime Support for Server Features

Files

isue1668.ods	24 KB	01/28/2013	Ovidiu Maxiniuc
isue1668.ods	23.4 KB	02/01/2013	Vadim Nebogatov
om_upd20130201b.zip	6.48 KB	02/01/2013	Ovidiu Maxiniuc
om_upd20130201a.zip	343 KB	02/01/2013	Ovidiu Maxiniuc
issue1668supportedby.ods	23.4 KB	02/12/2013	Vadim Nebogatov
vmn_upd20130213a.zip	90.7 KB	02/13/2013	Vadim Nebogatov
vmn_upd20130213a_test_cases.zip	2.26 KB	02/13/2013	Vadim Nebogatov
vmn_upd20130214a.zip	90.4 KB	02/15/2013	Eric Faulhaber
vmn_upd20130217a.zip	320 KB	02/17/2013	Vadim Nebogatov
vmn_upd20130217a_test_cases.zip	3.14 KB	02/17/2013	Vadim Nebogatov
vmn_upd20130218a.zip	322 KB	02/18/2013	Vadim Nebogatov
vmn_upd20130218a_test_cases.zip	3.14 KB	02/18/2013	Vadim Nebogatov
vmn_upd20130218b.zip	322 KB	02/19/2013	Eric Faulhaber
vmn_upd20130219a.zip	364 KB	02/19/2013	Vadim Nebogatov
vmn_upd20130219a_test_cases.zip	3.28 KB	02/19/2013	Vadim Nebogatov
vmn_upd20130219b.zip	86.5 KB	02/19/2013	Vadim Nebogatov
vmn_upd20130219b_test_cases.zip	3.28 KB	02/19/2013	Vadim Nebogatov
om_upd20130304c.zip	10.5 KB	03/04/2013	Ovidiu Maxiniuc
ecf_upd20130308a.zip	6.82 KB	03/08/2013	Eric Faulhaber
vmn_upd20130320a.zip	55.7 KB	03/20/2013	Vadim Nebogatov
vmn_upd20130320b.zip	735 Bytes	03/20/2013	Vadim Nebogatov
vmn_upd20130321a.zip	56.3 KB	03/21/2013	Vadim Nebogatov
vmn_upd20130321b.zip	735 Bytes	03/21/2013	Vadim Nebogatov
cs_upd20130322b.zip	15.4 KB	03/22/2013	Costin Savin
ca_upd20130323a.zip	31.4 KB	03/23/2013	Constantin Asofiei
ca_upd20130323d.zip	51.1 KB	03/23/2013	Constantin Asofiei
ca_upd20130323h.zip	51.2 KB	03/23/2013	Constantin Asofiei
ecf_upd20130417a.zip	40.8 KB	04/17/2013	Eric Faulhaber
vmn_upd20130829a.zip	121 KB	08/29/2013	Vadim Nebogatov
vmn_upd20130829b.zip	482 Bytes	08/29/2013	Vadim Nebogatov