Database - Feature #3815

more temp-table options

11/25/2018 01:57 PM - Eric Faulhaber

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

Description

Related issues:

Related to Database - Feature #3809: ProDataSet support	**Closed**	
Related to Database - Bug #4520: Implement proper codepage inheritance from p...	**New**	**01/24/2020**

History

#1 - 11/25/2018 01:59 PM - Eric Faulhaber

The following temp-table options need to be supported:

- LIKE-SEQUENTIAL
- BEFORE-TABLE (ProDataSet-related)

#2 - 11/25/2018 01:59 PM - Eric Faulhaber

- Related to Feature #3809: ProDataSet support added

#3 - 11/25/2018 02:13 PM - Eric Faulhaber

Temp-table field option:

- COLUMN-CODEPAGE

#4 - 04/22/2019 07:48 AM - Greg Shah

BEFORE-TABLE

Isn't this going to be handled as part of the work for #3809?

#5 - 04/22/2019 07:48 AM - Greg Shah

- Assignee set to Stanislav Lomany

#6 - 05/08/2019 02:42 PM - Stanislav Lomany

Do I need to work on BEFORE-TABLE as well?

#7 - 05/08/2019 03:28 PM - Eric Faulhaber

Ovidiu, please see previous posts. Is conversion of BEFORE-TABLE already handled as part of #3809?

#8 - 05/08/2019 03:48 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Ovidiu, please see previous posts. Is conversion of BEFORE-TABLE already handled as part of #3809?

I was not aware of this option. However, it seems to be already working.
OTOH, in 3809a (already in trunk), I added only the attribute. For the moment, it is only a default method in Temporary interface.

#9 - 05/08/2019 04:06 PM - Greg Shah

Stanislav: Don't do anything on BEFORE-TABLE. It will be handled in #3809.

> However, it seems to be already working.

Ovidiu: Do you mean that it is already working in 3809c?

#10 - 05/09/2019 03:29 AM - Ovidiu Maxiniuc

Yes, the BEFORE-TABLE option is working for me. I created a small testcase:

```
DEFINE TEMP-TABLE tt2 BEFORE-TABLE tt5
   FIELD f21 AS CHAR.

FIND FIRST tt5
    WHERE f21 EQ ?
     NO-ERROR.
```

And FWD created the DMO interfaces Tt5_2_1 and Tt2_1_1 and their implementations. The generated code is like this:

```
         silent(() -> new FindQuery(tt5_2_1, "upper(tt5_2_1.f21) is null", null, "tt5_2_1.id asc").first());
```

I think this is fine.
The branch 3809_ only adds the method declarations (default methods) for BEFORE-TABLE and AFTER-TABLE attributes to temp-table object handle.

#11 - 05/09/2019 09:09 AM - Stanislav Lomany

As far as I tested, LIKE and LIKE-SEQUENTIAL behave in the same way (like LIKE-SEQUENTIAL). E.g. we have table definitions where POSITION and ORDER have an opposite order.

```
ADD TABLE "some-table"
  AREA "Schema Area"
  DUMP-NAME "some-table"

ADD FIELD "field1" OF "some-table" AS character
  FORMAT "x(8)"
  INITIAL ""
  POSITION 2
  MAX-WIDTH 16
  ORDER 20

ADD FIELD "filed2" OF "some-table" AS character
  FORMAT "x(8)"
  INITIAL ""
  POSITION 3
  MAX-WIDTH 16
  ORDER 10

.
PSC
cpstream=ISO8859-1
.
0000000327
```

Testcase:

```
def temp-table tt like some-table.
def temp-table tt-seq like-sequential some-table.

def var bufh as handle.

bufh = buffer tt:handle.
message bufh:buffer-field(1):name bufh:buffer-field(2):name.

bufh = buffer tt-seq:handle.
message bufh:buffer-field(1):name bufh:buffer-field(2):name.
```

Output:

```
field2 field1
field2 field1
```

So the fields are always ordered by ORDER. What am I missing?

#12 - 05/09/2019 10:42 AM - Greg Shah

This seems to be a "feature" of the BUFFER-FIELD() method.

Please see this article:

https://knowledgebase.progress.com/articles/Article/000027330

> When the BUFFER-FIELD method is used with a numeric parameter (index), the fields are retrieved by the ORDER (_order) value.

#13 - 05/09/2019 05:09 PM - Stanislav Lomany

OK, we have fieldId, order and position fields, and the last two are optional.

```
@LegacyField(fieldId = 1, name = "field2", format = "x(8)", order = 0, position = 3)
private final character field2;
```

E.g. RAW-TRANSFER picks order or position-based order to walk through the fields. But how does it know which way to choose? Should we add table-level boolean field to LegacyTable annotation which specifies if LIKE-SEQUENTIAL was used?

#14 - 05/09/2019 07:32 PM - Greg Shah

If we need this information for runtime purposes, then we do need to add this to annotations.

#15 - 05/10/2019 10:33 AM - Stanislav Lomany

Do you know other function than RAW-TRANSFER and BUFFER-FIELD that depend on fields order?

#16 - 05/10/2019 10:43 AM - Greg Shah

I don't know the answer. Ovidiu and/or Eric may have some ideas there.

I think searching the 4GL docs and knowledge base for the metadata field names may help you find some dependencies.

#17 - 05/10/2019 11:04 AM - Ovidiu Maxiniuc

Stanislav Lomany wrote:

> Do you know other function than RAW-TRANSFER and BUFFER-FIELD that depend on fields order?

Not at this moment. However, the READ/WRITE-XML/JSON methods might depend on the order of the fields.

#18 - 05/10/2019 11:07 AM - Greg Shah

Hmm. I suspect the order of processing inside BUFFER-COPY and BUFFER-COMPARE may also depend on ORDER or POSITION. For example, if two failures were to occur, it might affect which one would be reported.

#19 - 05/10/2019 11:38 AM - Eric Faulhaber

Greg Shah wrote:

> Hmm. I suspect the order of processing inside BUFFER-COPY and BUFFER-COMPARE may also depend on ORDER or POSITION. For example, if two failures were to occur, it might affect which one would be reported.

Possibly. If so, I'm not sure we support this properly today.

Also, what about displaying or updating the entire table (i.e., not specific fields)?

#20 - 05/13/2019 05:31 PM - Stanislav Lomany

- Status changed from New to WIP

WRITE-XML/JSON as well as buffer DISPLAY/UPDATE use order by ORDER.
I couldn't make BUFFER-COPY and BUFFER-COMPARE produce an error to check the order.

So, at this point, only RAW-TRANSFER is affected by the difference between LIKE and LIKE-SEQUENTIAL. I suppose ProDataSet functions can be affected too.

As far as I tested, an arbitrary POSITION can be specified in .df file and properly imported into a database. But if the set of POSITIONS is not continuous, i.e. 2, 3, 4... (position 1 is reserved by recid), then RAW-TRANSFER behaves like if the table is defined with LIKE-SEQUENTIAL. E.g. 4, 2, 3 is an appropriate set of positions, while 20, 30, 40 isn't.

#21 - 05/14/2019 10:42 AM - Constantin Asofiei

What is the state of BEFORE-TABLE option at conversion time? I need this for my current work...

#22 - 05/14/2019 04:10 PM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> What is the state of BEFORE-TABLE option at conversion time? I need this for my current work...

As I understand, the problem with static BEFORE-TABLE is that, if the buffer is not accessed in current procedure (probably the class, too), the buffer is not defined at all in Java source. This is not an optimization as I thought at first , but a consequence that the buffers are defined/registered (with BufferScopeWorker.registerBuffer()) when they are encountered the first time. The DMO class is yet constructed as the parser takes care to define the DMO class/interface and later process and generated as usual. Probably a annotation flag about before/after table will be necessary at runtime. I do not know at this moment how can we force the buffer to be defined: to do this we need all annotations, which are not present in the source's ast (only in .schema/.p2o).

#23 - 05/16/2019 09:12 PM - Stanislav Lomany

Into what branch should I commit?
Am I right that there is no runtime part for COLUMN-CODEPAGE?

#24 - 05/18/2019 01:29 AM - Eric Faulhaber

Stanislav Lomany wrote:

> Into what branch should I commit?

Probably 3751a. Constantin, is that ok with you?

> Am I right that there is no runtime part for COLUMN-CODEPAGE?

Are you implementing this as a DMO implementation class annotation? There should be corresponding runtime updates in TempTableMapper. We don't read the DMO annotations at each data access, but rather use TempTableMapper.

Beyond the immediate scope of your changes, but before long we will have to honor this setting wherever we handle CLOB encoding in the runtime. E.g., the LobCopy implementation, possibly in the RecordBuffer invocation handler, maybe other places...

#25 - 05/18/2019 10:00 AM - Greg Shah

> Beyond the immediate scope of your changes

I'm not sure about this. I don't know when we would do this otherwise.

#26 - 05/20/2019 02:20 PM - Stanislav Lomany

Guys, do we need to support arbitrary expressions as a COLUMN-CODEPAGE? As far as I get, it can be combination of:

1. constants,
2. initial values of variables,
3. values of input parameters.

In this case we'll have to pass the expression to some function at the beginning of the external procedure. E.g.

```
Tt1_1_1.Buf tt1 = TemporaryBuffer.define(Tt1_1_1.Buf.class, "tt1", "tt1", false, false);

public void execute(final character _p)
{
   character p = UndoableFactory.initInput(_p);
   character str = UndoableFactory.character("zzz");

   externalProcedure(Codepage.this, new Block((Body) () ->
   {
      setCodepage(tt1, concat(str, p, "const"));
      RecordBuffer.openScope(tt1);
      ...
```

Or we are fine with string constants only?

#27 - 05/20/2019 02:21 PM - Constantin Asofiei

Stanislav Lomany wrote:

> In this case we'll have to pass the expression to some function at the beginning of the external procedure. E.g.

Are you sure the expression is computed only once? And if so, when is resolved - at the same as e.g. DOWN or browse-title? I ask because we may need to emit this as a Resolvable or lambda expr.

#28 - 05/20/2019 02:25 PM - Constantin Asofiei

Eric Faulhaber wrote:

> Stanislav Lomany wrote:
>
> > Into what branch should I commit?
>
> Probably 3751a. Constantin, is that ok with you?

Not really - 3751a is getting pretty massive, and I want to test and merge it soon.

#29 - 05/20/2019 02:29 PM - Eric Faulhaber

Constantin Asofiei wrote:

> Eric Faulhaber wrote:
>
> > Stanislav Lomany wrote:
> >
> > > Into what branch should I commit?
> >
> > Probably 3751a. Constantin, is that ok with you?
>
> Not really - 3751a is getting pretty massive, and I want to test and merge it soon.

Ok. When I wrote this I thought these changes were needed for the same projects as 3751a.

Stanislav, please manage these changes in a separate branch or hold them until we have a follow-up branch 3751b.

#30 - 05/20/2019 02:32 PM - Greg Shah

Stanislav Lomany wrote:

> Guys, do we need to support arbitrary expressions as a COLUMN-CODEPAGE? As far as I get, it can be combination of:
>
> 1. constants,
> 2. initial values of variables,
> 3. values of input parameters.
>
> In this case we'll have to pass the expression to some function at the beginning of the external procedure. E.g.
> [...]
>
> Or we are fine with string constants only?

No, we cannot limit this to constants. I don't want to revisit this later, it will be very tricky to remember and then we will be chasing a conversion deadline when we are surprised by this.

Please check to confirm that the expression is only evaluated once (try to include a built-in function that has a counter). If you cannot add anything other than items that are known on entry to the procedure, then it is effectively evaluated once for the procedure.

If so, then your suggested implementation is good. If not, we will need to implement the lambda.

#31 - 05/20/2019 02:33 PM - Greg Shah

> Ok. When I wrote this I thought these changes were needed for the same projects as 3751a.

It is but for different deadlines. Does it make sense to include it in 3809c?

#32 - 05/20/2019 06:03 PM - Stanislav Lomany

> Please check to confirm that the expression is only evaluated once

It is evaluated once.

#33 - 05/21/2019 02:26 PM - Stanislav Lomany

It is interesting that when COLUMN-CODEPAGE is calculated *initial* value of the parameter is used instead of the passed one. I.e.

```
run "codepage.p" ("PASSED").
....
def input parameter p as char init "INIT".
```

"INIT" is used in this case. If there is no INIT specification the value is considered to be an empty string.
I don't think we should support this at this point.

#34 - 05/22/2019 10:17 AM - Greg Shah

> I don't think we should support this at this point.

The problem here is that if we don't support this now, we will have to debug this later when we hit it somewhere. It is a stupid implementation, but we should just handle this. On a positive note, any references to parameters can be replaced with a string literal during annotations. Then the expression will convert normally with no other special processing.

#35 - 05/23/2019 11:58 AM - Stanislav Lomany

Guys, I need some technical help. I was going to emit a function that sets the codepage in convert/buffer_definitions.rules. Here is the tree at the beginning of buffer_definitions.rules:

```
BUFFER-DEFINITIONS: block [BLOCK]:1971389988865 @0:0
      (builtin-cls=false, dotnet-cls=false, basename=scope2.p, support_level=16400, package-base=, pkgname=com
.goldencode.testcases, classname=Scope2, relative-name=scope2.p, numundolocal=0, numundoglobal=0, numparms=0,
new_shared_scope=false, recordscoping=true, loop=false, loopnext=false, toplevel=true, translevel=137, reasons
=0, javaname=Scope2, peerid=1988569858062)
   statement [STATEMENT]:1971389988866 @0:0
      def [DEFINE_TEMP_TABLE]:1971389988867 @1:1
            (schemaname=tt, bufname=tt, dbname=, support_level=16400, bufdefkey=tt,tt, bufrefkey=tt,tt,-1, jav
aname=tt, classname=Tt_1_1, bufclassname=Tt_1_1.Buf, pkgname=com.goldencode.testcases.dmo._temp, bufreftype=18
, uniquename=tt_tt)
         temp-table [KW_TEMP_TAB]:1971389988869 @1:5
            tt [SYMBOL]:1971389988871 @1:16

            no-undo [KW_NO_UNDO]:1971389988873 @1:19
                  (support_level=16400)
            reference-only [KW_REF_ONLY]:1971389988875 @1:27
                  (support_level=8208)
            define field [DEFINE_FIELD]:1971389988879 @0:0
               f1 [SYMBOL]:1971389988880 @1:48
               as [KW_AS]:1971389988882 @1:51
                  clob [KW_CLOB]:1971389988884 @1:54
               COLUMN-CODEPAGE [KW_COL_CP]:1971389988886 @2:45
                     (support_level=257)
                  expression [EXPRESSION]:1971389988888 @0:0
                        (support_level=16400)
                     "test" [STRING]:1971389988889 @2:61
                           (is-literal=true, support_level=16400)
            define field [DEFINE_FIELD]:1971389988893 @0:0
               f2 [SYMBOL]:1971389988894 @3:48
               as [KW_AS]:1971389988896 @3:51
                  clob [KW_CLOB]:1971389988898 @3:54

   tt_tt [BUFFER_SCOPE]:1971389988906 @0:0
         (bufname=tt, schemaname=tt, dbname=, scopetype=23, implicit=false, readonly=false, static=false, java
name=tt, recordtype=14, first=true, classname=Tt_1_1, bufclassname=Tt_1_1.Buf, pkgname=com.goldencode.testcase
s.dmo._temp)
   statement [STATEMENT]:1971389988901 @0:0
      create [KW_CREATE]:1971389988902 @5:1
            (support_level=16400)
         tt [TEMP_TABLE]:1971389988904 @5:8
               (schemaname=tt, bufname=tt, dbname=, recordtype=14, bufrefkey=tt,tt,-1, bufreftype=21, uniquena
me=tt_tt, refid=1971389988906)
```

And here is the order in which this tree is iterated in walk-rules:

```
BLOCK 1971389988865
STATEMENT 1971389988866
DEFINE_TEMP_TABLE 1971389988867
KW_TEMP_TAB 1971389988869
SYMBOL 1971389988871

BUFFER_SCOPE 1971389988906
STATEMENT 1971389988901
KW_CREATE 1971389988902
TEMP_TABLE 1971389988904
```

Do you have an idea why the middle part is not iterated? Nodes are not hidden.

#36 - 05/23/2019 12:53 PM - Greg Shah

> Do you have an idea why the middle part is not iterated? Nodes are not hidden.

The only good reason is that they are hidden. And I think they usually are hidden. See annotations/cleanup.rules:

```
      <!-- structural definition within a define temp table -->
      <rule>
         (parent.parent.type == prog.define_temp_table or
          parent.parent.type == prog.define_work_table)  and
         type != prog.symbol
         <action>hidenode = copy</action>
      </rule>
```

#37 - 05/23/2019 05:35 PM - Stanislav Lomany

> The only good reason is that they are hidden.

Sorry for bothering you, I wrote a code in DumpTree which prints node visibility and there was a typo in it.

#38 - 05/27/2019 06:01 PM - Stanislav Lomany

I've found that the CODEPAGE can be inherited from another permanent or temp-table through either table- or field-level LIKE option.

For permanent databases I don't see a way to create a dynamic expression as a codepage, so we can store the codepage in LegacyField annotation of the permanent table. (I'm not sure if we support CLOB-CODEPAGE in .df files)

However we don't keep the information about the parent table/field specified by LIKE statement. So I suggest to add like field to the LegacyField/LegacyTable annotation where the parent field/table will be kept "as is", i.e. "table-name" or "field-name" or "table-name.field-name" etc.

The only thing is that we have to keep track of table hierarchy at runtime, so CODEPAGE assigned to a parent temp-table will take affect on the child tables.

Are you OK with this changes?

#39 - 05/28/2019 10:15 AM - Eric Faulhaber

Stanislav Lomany wrote:

> I've found that the CODEPAGE can be inherited from another permanent or temp-table through either table- or field-level LIKE option.
>
> For permanent databases I don't see a way to create a dynamic expression as a codepage, so we can store the codepage in LegacyField annotation of the permanent table. (I'm not sure if we support CLOB-CODEPAGE in .df files)

We don't yet. I had not noticed this keyword previously. We should support this in the same way (in the LegacyField annotation).

> However we don't keep the information about the parent table/field specified by LIKE statement. So I suggest to add like field to the LegacyField/LegacyTable annotation where the parent field/table will be kept "as is", i.e. "table-name" or "field-name" or "table-name.field-name" etc.
>
> The only thing is that we have to keep track of table hierarchy at runtime, so CODEPAGE assigned to a parent temp-table will take affect on the child tables.
>
> Are you OK with this changes?

Yes.

#40 - 05/30/2019 06:52 PM - Stanislav Lomany

Created task branch 3815a from P2J trunk revision 11311.

#41 - 06/06/2019 11:36 AM - Stanislav Lomany

Side note: when a longchar/clob value is assigned to a clob field, it is implicitly converted into the codepage of the target field. I suppose we'll have to handle it in RecordBuffer.invoke in the future.

#42 - 06/06/2019 07:21 PM - Stanislav Lomany

> I've found that the CODEPAGE can be inherited from another permanent or temp-table through either table- or field-level LIKE option.

I think I misunderstood the documentation, temp-table doesn't inherit CODEPAGE from a permanent table.

#43 - 06/09/2019 02:43 PM - Stanislav Lomany

Guys, I want to know your opinion on how to better implement code page inheritance. We have the following things to consider:

1. Where to store a temp-table code page? I suggest to use StaticTempTable/TempTableBuilder.
2. If the field is a LIKE-field, then the parent temp-table is searched by its legacy definition. That means that the parent StaticTempTable should already be initialized.
3. We need to determine the point where the code page is applied to a table. I'm inclined to do it at the openScope **IF** the buffers are guaranteed to be opened in the definition order (it seems to be true).

Code page is set in the separate function TemporaryBuffer.setCodePage(buffer, fieldName, codepage). The question is where to put it considering the items above? If the items above are OK it should be placed before the openScope. Can it be placed it block initialization section? Also, should the fieldName be legacy name or DMO property name?
This function will put code pages into some temporary storage from which it'll be applied in openScope to parent and then child tables.

#44 - 06/10/2019 05:26 PM - Eric Faulhaber

Stanislav Lomany wrote:

> Guys, I want to know your opinion on how to better implement code page inheritance. We have the following things to consider:
> Where to store a temp-table code page? I suggest to use StaticTempTable/TempTableBuilder.

Which one are you suggesting? I did not design these classes, nor TableMapper, but it seems we split legacy information across these classes. What is the reasoning behind which type of legacy information goes into each of these classes?

> If the field is a LIKE-field, then the parent temp-table is searched by its legacy definition. That means that the parent StaticTempTable should already be initialized.

This presumes the previous suggestion, but I want to understand the answer to my additional question first...

> We need to determine the point where the code page is applied to a table. I'm inclined to do it at the openScope **IF** the buffers are guaranteed to be opened in the definition order (it seems to be true).

One thing to confirm: at some point, I think the order was not guaranteed, due to the use of a HashMap under the covers to store BUFFER_SCOPE AST branches, but that may have been resolved in the meantime, with the use of LinkedHashMap.

> Code page is set in the separate function TemporaryBuffer.setCodePage(buffer, fieldName, codepage). The question is where to put it considering the items above? If the items above are OK it should be placed before the openScope. Can it be placed it block initialization section?

Can the code page for a temp-table, once set, be overridden dynamically? If not, I would think this would be appropriate. Greg, any objection?

> Also, should the fieldName be legacy name or DMO property name?

Is there any use case which would require one rather than the other? We have the mapping between legacy name and property name available at runtime. Which is the more efficient to use (i.e., minimal string processing and lookups at runtime)?

> This function will put code pages into some temporary storage from which it'll be applied in openScope to parent and then child tables.

#45 - 06/10/2019 05:34 PM - Greg Shah

> Code page is set in the separate function TemporaryBuffer.setCodePage(buffer, fieldName, codepage). The question is where to put it considering the items above? If the items above are OK it should be placed before the openScope. Can it be placed it block initialization section?

> Can the code page for a temp-table, once set, be overridden dynamically? If not, I would think this would be appropriate. Greg, any objection?

No objection.

#46 - 06/11/2019 05:03 AM - Stanislav Lomany

> Which one are you suggesting? I did not design these classes, nor TableMapper, but it seems we split legacy information across these classes. What is the reasoning behind which type of legacy information goes into each of these classes?

StaticTempTable is supposed to keep code pages (property name -> code page map) for a static table. TempTableBuilder is a parent table object for dynamic tables therefore it'll keep code pages for a dynamic table.

> Can the code page for a temp-table, once set, be overridden dynamically?

No.

> Is there any use case which would require one rather than the other? We have the mapping between legacy name and property name available at runtime. Which is the more efficient to use (i.e., minimal string processing and lookups at runtime)?

From that point property name is better.

#47 - 06/11/2019 01:47 PM - Stanislav Lomany

An interesting observation. Consider we declared a shared temp-table:

```
def new shared temp-table tt1 no-undo field f1 as clob COLUMN-CODEPAGE func1().
```

And then used it in an another procedure:

```
def shared temp-table tt1 no-undo field f1 as clob COLUMN-CODEPAGE func1().
```

Note that the functions func1() may be different. The question: which function will define the code page? The answer: if the shared table is involved in a LIKE- table of field definition, e.g.:

```
def temp-table tt11 like tt1.
```

then the second function (shared ... func1()) is used. Otherwise the first function (new shared ... func1()) is used.

#48 - 06/12/2019 12:09 PM - Stanislav Lomany

Looks like the code page expression should be resolvable/lambda. Yes, it is evaluated once, but it can depend on the state of the buffers previously opened in the same openScope(buf1, buf2, buf3 ...) statement. So, if we don't want to split openScope statement, we have to do this. Should I go for resolvable or lambda?

#49 - 06/12/2019 12:15 PM - Greg Shah

Use a lambda. This is easily done. See convert/expressions.rules and the usage of tw.graft("lambda_expr", copy, closestPeerId).

#50 - 06/12/2019 12:16 PM - Eric Faulhaber

Agreed. Unless there is a technical advantage to using a Resolvable (e.g., re-usability), we generally prefer to use lambdas to inner classes for inlined converted code.

#51 - 06/13/2019 04:57 PM - Stanislav Lomany

Guys, can you give me an advice how to emit something at Init section of the external procedure or in the main block before the openScope(buffers) call? Normally I wouldn't bother you, but it is the last conversion issue and I have limited amount of time.

#52 - 06/13/2019 06:08 PM - Greg Shah

The init block can be found using the controlid annotation in the inner block node.

See save_control_point_ids in common-progress.rules for how we store it. If you look how that is called in control_flow.rules, you can see that in lambda mode, we are processing using the tree from the block_lambda_def template in java_templates.tpl. We start at the peer node which is the BLOCK node in the Body section. We look up instmeth = ref.getAncestor(3) to the CONSTRUCTOR node and to find the ID of the Init section, we use controlid = instmeth.getChildAt(1).getChildAt(0).getChildAt(1).id. We then store the controlid annotation in the prog.inner_block node.

The key here is you just need to lookup the controlid annotation from the prog.inner_block. Call getAst(controlid) to get the BLOCK node for the Init section. Or you can use the ID directly in one of the create or move or graft AST calls.

#53 - 06/14/2019 08:32 AM - Stanislav Lomany

> I've found that the CODEPAGE can be inherited from another permanent or temp-table through either table- or field-level LIKE option.

I think I misunderstood the documentation, temp-table doesn't inherit CODEPAGE from a permanent table.

Heh, it's an open question if temp-table inherits CODEPAGE from a permanent table. It is just buggy. A field was not inheriting. Then I noticed that the other field seems to be inheriting code page from a permanent table. I've recreated the first field and, voila, it became inheriting.
I'm not sure how it should be by design.

#54 - 06/14/2019 08:41 AM - Stanislav Lomany

FYI, I'm not going to implement #3815-47 quirk because it requires to create hierarchy tracking mechanism for an unreal scenario.

#55 - 06/14/2019 09:16 AM - Greg Shah

Create a task that describes the quirk, make Eric and myself watchers and make it related to this task.

#56 - 06/14/2019 01:40 PM - Stanislav Lomany

I put some code page conversion rules into annotations/sorting.rules, but because it's not sorting-related rules, should I put it into a new separate file?

#57 - 06/14/2019 01:48 PM - Eric Faulhaber

Stanislav Lomany wrote:

> I put some code page conversion rules into annotations/sorting.rules, but because it's not sorting-related rules, should I put it into a new separate file?

Since the changes are not primarily about sorting the nodes of certain structures into a known order (the main point of this rule set), then yes, it should be in a separate file. How extensive are the changes? Are they likely to be expanded later?

#58 - 06/14/2019 02:40 PM - Stanislav Lomany

> How extensive are the changes?

60 core lines of separate rules.

> Are they likely to be expanded later?

I doubt about that.

#59 - 06/14/2019 03:03 PM - Stanislav Lomany

Rebased task branch 3815a from P2J trunk revision 11317.

#60 - 06/14/2019 09:34 PM - Stanislav Lomany

Please review 3815a rev 11327.

#61 - 06/17/2019 04:44 AM - Stanislav Lomany

> Please review 3815a rev 11327.

FYI it passed regression testing.

#62 - 06/20/2019 02:52 PM - Eric Faulhaber

Code review 3815a/11327:

Nice update. A few things to consider...

Not your issue, but rather the correction of an omission I previously made: please add the package name for the BLOB data type to the packages map in java_dmo.xml.

Copyright notice year update needed in DumpTree.

Should BLOB be added to the ParmType enumeration? I wasn't considering parameters when I added LOB support.

Same (add BLOB) for DynamicTablesHelper.

P2JField: it looks like the refactoring of the equals method has caused a regression. Some comparisons from the old implementation which were meant to test for inequality to exit early with a false return are now being used as subexpressions which should evaluate to true. That is, the extent and caseSensitive comparisons should be testing for equality, not inequality.

I have no idea if this would be a bottleneck, but is there a more direct way to access the code page from RecordBuffer.getCodePage(BufferField)? It seems like creating a FieldReference just to do this is expensive. Don't spend much time on this, since we don't actually know if this represents a performance problem yet, but if you have a cheaper way to get this value, let's use it.

TempTableBuilder: what is special about ParmType.CLOB that we use toLowerCase on this type and not on any of the others? Is this a latent bug for the other types? Also, should we add BLOB here?

In TempTableBuilder.addNewField, there is an existing comment:

```
// Note: case-sensitivity cannot be specified using P4GL ADD-NEW-FIELD method
```

Does the same apply to code page, since you are passing null for code page to the P2JField c'tor? If so, please add that info to the comment.

Please add TYPE_BLOB to DmoAsmTypes and DmoAsmWorker.

Greg: please take a quick look at the changes in p2j.uast, particularly the parser changes. They are pretty simple and seem OK to me.

There is a focus-related change in BrowseGuiImpl and some other browse-related doc changes that seem out of place for this branch, but I assume it is ok, since you are the expert there. Just pointing this out in case you didn't mean to include this in this branch.

I'm not crazy about the addition of Text.assign(clob), since this is an ancestor class that is now aware of its descendant in the class hierarchy. What is driving this change?

I don't think another run of ChUI regression testing is needed after my proposed changes (unless the Text.assign(clob) thing becomes more complicated). That test environment doesn't hit any of the features I've asked you to update.

#63 - 06/20/2019 03:58 PM - Greg Shah

Code Review Task Branch 3815a Revision 11327

The src/com/goldencode/p2j/uast/* changes are fine. I didn't look at anything else.

#64 - 06/24/2019 06:13 AM - Stanislav Lomany

- File codepage-dyn.p added

> I'm not crazy about the addition of Text.assign(clob), since this is an ancestor class that is now aware of its descendant in the class hierarchy.

Me neither. I suppose it can be solved by changes in runtime conversion part.

> What is driving this change?

```
Caused by: java.lang.NoSuchMethodError: com.goldencode.p2j.util.clob.assign(Lcom/goldencode/p2j/util/clob;)V
        at com.goldencode.p2j.persist.dynamic._temp.impl.DynamicRecord1Impl.setField1(Unknown Source)
        at com.goldencode.p2j.persist.dynamic._temp.impl.DynamicRecord1Impl.assign(Unknown Source)
        at com.goldencode.p2j.persist.dynamic._temp.impl.DynamicRecord1Impl.deepCopy(Unknown Source)
        at com.goldencode.p2j.persist.RecordBuffer.create(RecordBuffer.java:9058)
        at com.goldencode.p2j.persist.BufferImpl.bufferCreate(BufferImpl.java:2153)
        at com.goldencode.testcases.CodepageDyn.lambda$execute$2(CodepageDyn.java:34)
```

Testcase attached.

#65 - 06/25/2019 03:40 PM - Stanislav Lomany

I found an issue: add-fields-from and add-like-field cannot find the source LIKE table if it is a shared table. The same issue when we try to evaluate a CODEPAGE declared for a shared table.

#66 - 06/25/2019 03:42 PM - Eric Faulhaber

Stanislav Lomany wrote:

> I found an issue: add-fields-from and add-like-field cannot find the source LIKE table if it is a shared table. The same issue when we try to evaluate a CODEPAGE declared for a shared table.

Do you have a fix in mind? If so, what is your estimate of effort/time to implement?

#67 - 06/25/2019 04:17 PM - Stanislav Lomany

I need help with an idea for the fix. getStaticTempTableDepth(String name) searches for the table using (table-name, procedure) key. The *current* procedure is used in the key. But the table is stored in openStaticTempTables using the key that contains the procedure in which the shared table was *defined*.

#68 - 06/26/2019 04:09 AM - Stanislav Lomany

> TempTableBuilder: what is special about ParmType.CLOB that we use toLowerCase on this type and not on any of the others? Is this a latent bug for the other types? Also, should we add BLOB here?

For whatever reason "CLOB" had data type name in capital letters. Now I've changed it to "clob", removed to toLowerCase, added "blob" and checked that code page tetcases work.

#69 - 06/26/2019 10:00 AM - Eric Faulhaber

Stanislav Lomany wrote:

> > I'm not crazy about the addition of Text.assign(clob), since this is an ancestor class that is now aware of its descendant in the class hierarchy.
>
> Me neither. I suppose it can be solved by changes in runtime conversion part.

What are you proposing in this regard?

#70 - 06/26/2019 10:12 AM - Eric Faulhaber

Stanislav Lomany wrote:

> I need help with an idea for the fix. getStaticTempTableDepth(String name) searches for the table using (table-name, procedure) key. The *current* procedure is used in the key. But the table is stored in openStaticTempTables using the key that contains the procedure in which the shared table was *defined*.

I don't know how we would go about determining the procedure in which a NEW SHARED TEMP-TABLE was defined from the current procedure. Constantin, Ovidiu, any ideas here?

Ovidiu, how do we track this for static conversion of shared temp-tables?

Stanislav, a temp-table needs to be defined in the current procedure as SHARED, even if the "original" (NEW SHARED) definition is in another procedure. What information do you need which is missing from the SHARED definition in the current procedure, which is only available in the NEW SHARED definition in the "original" procedure?

#71 - 06/26/2019 11:41 AM - Ovidiu Maxiniuc

The NEW (SHARED) TEMP-TABLES are tracked by SharedVariableManager. See addTempTable() and lookupTempTable() methods. The definition of the NEW table is in former one. I am not sure you can find the instantiating procedure at that moment. INSTANTIATING-PROCEDURE support exists for static handles / HandleResource.

#72 - 06/26/2019 05:10 PM - Eric Faulhaber

Stanislav, what information do you need from the NEW SHARED TEMP-TABLE definition which is not in the SHARED TEMP-TABLE definition? Perhaps there is another way to store and retrieve this...

We have to get 3815a merged to trunk ASAP. I can live with the Text.assign(clob) issue temporarily if the runtime conversion changes you have in mind are time consuming. If these two issues currently are the only items blocking a merge to trunk, then let's get the "a" branch merged and open a "b" branch to address these. Are there any changes you've made after the code review that would require re-running regression testing?

#73 - 06/26/2019 05:19 PM - Stanislav Lomany

> I can live with the Text.assign(clob) issue temporarily if the runtime conversion changes you have in mind are time consuming.

I made this change to save time, I had no ideas on the first look. Do you want me specifically to look into it?

> If these two issues currently are the only items blocking a merge to trunk, then let's get the "a" branch merged and open a "b" branch to address these. Are there any changes you've made after the code review that would require re-running regression testing?

I don't think so. You can make a quick review.

#74 - 06/26/2019 05:36 PM - Eric Faulhaber

Stanislav Lomany wrote:

> I can live with the Text.assign(clob) issue temporarily if the runtime conversion changes you have in mind are time consuming.

I made this change to save time, I had no ideas on the first look. Do you want me specifically to look into it?

I thought you had a specific idea, based on your comment in #3815-64 about runtime conversion changes (I assumed you had some other signature in mind for DMO setters for CLOB fields). Let's not hold up merging 3815a to trunk for this. Please put a TODO in Text.assign(clob) for this and come back to it after the merge.

> If these two issues currently are the only items blocking a merge to trunk, then let's get the "a" branch merged and open a "b" branch to address these. Are there any changes you've made after the code review that would require re-running regression testing?

I don't think so. You can make a quick review.

The changes look good. I don't think re-running ChUI regression testing is necessary either.

If there is nothing else holding up this branch, please merge 3815a to trunk, then open a 3815b for these two remaining issues. We can work them in this task.

#75 - 06/26/2019 07:26 PM - Stanislav Lomany

Rebased task branch 3815a from P2J trunk revision 11319.

#76 - 06/26/2019 08:40 PM - Stanislav Lomany

3815a has been merged into the trunk as bzr revision 11320.

#77 - 06/26/2019 08:50 PM - Stanislav Lomany

- File codepage.p added

- File codepage-dyn.p added

- File codepage-hierarchy.p added

- File codepage-like.p added

- File codepage-order.p added

- File codepage-param.p added

- File codepage-param-run.p added

- File codepage-run.p added

- File codepage-shared.p added

- File codepage-shared-run.p added

I'll attach some testcases here. Note that at this point they do not work "out of the box" because of #3815-41. You have to manually replace longchar.getCodePage calls with RecordBuffer.getCodePage calls. E.g.

```
longchar.getCodePage((clob) new FieldReference(tt1, "fieldOne").getValue())
```

with

```
RecordBuffer.getCodePage(new FieldReference(tt1, "fieldOne"))
```

#78 - 06/27/2019 08:34 PM - Stanislav Lomany

> Stanislav, what information do you need from the NEW SHARED TEMP-TABLE definition which is not in the SHARED TEMP-TABLE definition? Perhaps there is another way to store and retrieve this...

I need to find a shared table by its legacy name. Using openStaticTempTables can only find a shared table only within the procedure it was defined. Using lookupTempTable() algorithm I can find a shared temp table previously registered by addTempTable, but the items in SharedVariableManager.tempTables are keyed by converted names. Do you have a better idea than adding another mapping by legacy names?

#79 - 07/01/2019 11:43 AM - Stanislav Lomany

From Constantin: conversion fails because of a TEMP-TABLE with a COLUMN-CODEPAGE field defined in an interface.

```
     [java] ./abl/...iLoggingAdapter.cls
     [java] EXPRESSION EXECUTION ERROR:
     [java] --------------------------
```

```
     [java] tw.graft("lambda_expr", copy, lastid)
     [java]                               ^  { java.lang.NullPointerException }
     [java] ----------------------------
     [java] ERROR:
     [java] com.goldencode.p2j.pattern.TreeWalkException: ERROR!  Active Rule:
     [java] ------------------------
     [java]       RULE REPORT
     [java] ------------------------
     [java] Rule Type :   WALK
     [java] Source AST:  [ column-codepage ] BLOCK/KW_COL_CP/ @38:99 {13649406066853}
     [java] Copy AST  :  [ column-codepage ] BLOCK/KW_COL_CP/ @38:99 {13649406066853}
     [java] Condition :  tw.graft("lambda_expr", copy, lastid)
     [java] Loop      :  false
     [java] --- END RULE REPORT ---
     [java]
     [java]
     [java]
     [java]     at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:1069)
     [java]     at com.goldencode.p2j.convert.TransformDriver.processTrees(TransformDriver.java:537)
     [java]     at com.goldencode.p2j.convert.ConversionDriver.back(ConversionDriver.java:580)
     [java]     at com.goldencode.p2j.convert.TransformDriver.executeJob(TransformDriver.java:857)
     [java]     at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:983)
     [java] Caused by: com.goldencode.expr.ExpressionException: Expression execution error @1:31 [KW_COL_CP id
=13649406066853]
     [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:1632)
     [java]     at com.goldencode.p2j.pattern.PatternEngine.processAst(PatternEngine.java:1530)
     [java]     at com.goldencode.p2j.pattern.PatternEngine.processAst(PatternEngine.java:1478)
     [java]     at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:1033)
     [java]     ... 4 more
     [java] Caused by: com.goldencode.expr.ExpressionException: Expression execution error @1:31
     [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.RuleContainer.apply(RuleContainer.java:584)
     [java]     at com.goldencode.p2j.pattern.RuleSet.apply(RuleSet.java:98)
     [java]     at com.goldencode.p2j.pattern.RuleContainer.apply(RuleContainer.java:584)
     [java]     at com.goldencode.p2j.pattern.RuleSet.apply(RuleSet.java:98)
     [java]     at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:262)
     [java]     ... 9 more
     [java] Caused by: java.lang.NullPointerException
     [java]     at com.goldencode.expr.CE24549.execute(Unknown Source)
     [java]     at com.goldencode.expr.Expression.execute(Expression.java:391)
     [java]     ... 18 more
```

#80 - 07/01/2019 02:58 PM - Stanislav Lomany

Do you, guys, know, if a code page of a temp-table field defined in an interface have an actual effect? Temp-table definitions in an interface can be used as a parameter for interface methods. But the actual code page comes from an implementing class. Am I missing something?

#81 - 07/01/2019 02:59 PM - Constantin Asofiei

Stanislav Lomany wrote:

> Do you, guys, know, if a code page of a temp-table field defined in an interface have an actual effect? Temp-table definitions in an interface can be used as a parameter for interface methods. But the actual code page comes from an implementing class. Am I missing something?

This is confusing for me, too. For now, I would just drop any code related to a temp-table definition in an interface - as even a class implementing this interface wouldn't have access to the temp-table, it would have to define it, again.

#82 - 07/02/2019 04:44 PM - Stanislav Lomany

> For now, I would just drop any code related to a temp-table definition in an interface

I couldn't do it at this point: this requires additional changes because although eventually temp-tables definitions are not used in interface, they are used in the middle of the conversion process if the table is referenced in a table parameter.

However I fixed the issue by removing COLUMN-CODEPAGE node for tables defined in interfaces. I couldn't convert the full project in which the error occurred - some setup problems. But I converted the class which caused the error, so it should work.

Where I can put the fix?

#83 - 07/03/2019 08:27 AM - Constantin Asofiei

Stanislav Lomany wrote:

> Where I can put the fix?

Place it in 3855b.

#84 - 07/03/2019 02:27 PM - Stanislav Lomany

> Place it in 3855b.

Committed rev 11329.

#85 - 07/24/2019 04:52 PM - Greg Shah

Stanislav Lomany wrote:

> > Stanislav, what information do you need from the NEW SHARED TEMP-TABLE definition which is not in the SHARED TEMP-TABLE definition? Perhaps there is another way to store and retrieve this...
>
> I need to find a shared table by its legacy name. Using openStaticTempTables can only find a shared table only within the procedure it was defined. Using lookupTempTable() algorithm I can find a shared temp table previously registered by addTempTable, but the items in SharedVariableManager.tempTables are keyed by converted names. Do you have a better idea than adding another mapping by legacy names?

Mapping by legacy table name is the correct approach because it will duplicate what the 4GL is doing, if I understand correctly. This should be pretty simple to implement, but perhaps I'm missing something.

How much time do you expect is needed to resolve this open item?

#86 - 07/24/2019 04:53 PM - Greg Shah

> Please put a TODO in Text.assign(clob) for this and come back to it after the merge.

Unless there is a specific idea for this, let's just leave the TODO in the code and move on.

#87 - 07/24/2019 06:38 PM - Stanislav Lomany

> How much time do you expect is needed to resolve this open item?

I need to take a look again. If it's only about the adding a new map, it should be resolved quickly.

#88 - 07/26/2019 04:40 AM - Stanislav Lomany

> I need to take a look again. If it's only about the adding a new map, it should be resolved quickly.

Actually there are two problems:
1. Error occurs if the source table in target LIKE source has a COLUMN-CODEPAGE specified. The source table cannot be found by its name specified in like legacy annotation. The proposed solution is to modify helper functions that search for a temp-table by its name, like TempTableBuilder.getExistingFields in order to make them look not only in static temp-tables using BufferManager.getStaticTempTable, but also using SharedVariableManager. The problem is that the entries in the SVM.tempTables registry are keyed by converted name, not legacy name. I suppose that we don't want to have another registry keyed by legacy name, so I suggest to add converted name -> legacy name mapping into SVM. Because there can be multiple shared tables with the same name (scoped), I suggest to add "reference counter" into values of this map for proper cleanup. Are you OK with this?
2. Incorrect code page for a LIKE table inherited from a shared table. It should be in this way:

```
def new shared temp-table tt-one no-undo field f1 as clob COLUMN-CODEPAGE "UTF-8".  <--- UTF-8
...
def shared temp-table tt-one no-undo field f1 as clob COLUMN-CODEPAGE "UTF-32".     <--- UTF-8
def temp-table tt-like like tt-one.                                                 <--- UTF-32
```

#89 - 07/26/2019 07:20 AM - Greg Shah

> The problem is that the entries in the SVM.tempTables registry are keyed by converted name, not legacy name. I suppose that we don't want to have another registry keyed by legacy name, so I suggest to add converted name -> legacy name mapping into SVM. Because there can be multiple shared tables with the same name (scoped), I suggest to add "reference counter" into values of this map for proper cleanup. Are you OK with this?

Yes.

> Incorrect code page for a LIKE table inherited from a shared table. It should be in this way:

Is def shared temp-table tt-one in the same source file as def temp-table tt-like like tt-one.? It seems you are describing a static (compile/conversion time) issue here while the 1st issue as a runtime problem.

If this is a compile-time issue, then the codepage should just be copied as part of the LIKE processing. It should be an easy fix since we copy all the other kinds of field options already.

#90 - 07/26/2019 07:59 AM - Constantin Asofiei

Ovidiu/Stanislav: we need to determine how the TemporaryBuffer.tempTableRef is behaves. For master != null, I found that this is not set - so I changed tableHandle() to look at the master, if is set.

But I don't know if tempTableRef really needs to be null for non-master buffers - if this is true, and it can be null, then all usage in this class needs to be fixed so that the tempTableRef is resolved properly.

#91 - 07/26/2019 09:38 AM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> Ovidiu/Stanislav: we need to determine how the TemporaryBuffer.tempTableRef is behaves. For master != null, I found that this is not set - so I changed tableHandle() to look at the master, if is set.
>
> But I don't know if tempTableRef really needs to be null for non-master buffers - if this is true, and it can be null, then all usage in this class needs to be fixed so that the tempTableRef is resolved properly.

I encountered yesterday one issue with this field. Or better said that it was not initialized at the moment the scope of the buffer was not yet opened. As result I was unable to get the metadata from TableMapper. I disable the validation code, for the moment.

#92 - 07/26/2019 01:21 PM - Stanislav Lomany

> But I don't know if tempTableRef really needs to be null for non-master buffers - if this is true, and it can be null, then all usage in this class needs to be fixed so that the tempTableRef is resolved properly.

As Ovidiu noticed, it is null until the scope is opened. We need to determine in what cases we need the table information before that point and how to handle it.

#93 - 07/26/2019 02:08 PM - Stanislav Lomany

> Is def shared temp-table tt-one in the same source file as def temp-table tt-like like tt-one.?

No, they are in different files, sorry for vague notation. It's a runtime issue, I need to adjust codepage processing.

#94 - 07/31/2019 08:56 PM - Stanislav Lomany

Fixes for remaining CODEPAGE issues are in 3816a rev 11327.

#95 - 10/10/2019 04:04 PM - Stanislav Lomany

- Status changed from WIP to Review

3816a has been merged into the trunk as bzr revision 11337.

#96 - 10/15/2019 12:46 PM - Greg Shah

Everything is complete in this task?

Is the gap marking up to date as well?

#97 - 10/20/2019 07:31 AM - Stanislav Lomany

> Everything is complete in this task?

I think yes. Code page is properly resolved and set, but data it is not actually converted: when a longchar/clob value is assigned to a clob field, it should be implicitly converted into the code page of the target field. I suppose we'll have to handle it in RecordBuffer.invoke in the future.
Also I left a comment in the code which says that it's not clear if a temp-table inherits code page from a permanent table. I got controversial results while testing. We can return to it when we'll actually need it.

> Is the gap marking up to date as well?

What branch can I use for that?

#98 - 11/14/2019 12:47 PM - Greg Shah

- Status changed from Review to WIP

- % Done changed from 0 to 100

> Code page is properly resolved and set, but data it is not actually converted: when a longchar/clob value is assigned to a clob field, it should be implicitly converted into the code page of the target field. I suppose we'll have to handle it in RecordBuffer.invoke in the future.

I've created #4378 for this.

> Also I left a comment in the code which says that it's not clear if a temp-table inherits code page from a permanent table. I got controversial results while testing. We can return to it when we'll actually need it.

Please create a new task in the Database project to document this problem.

> Is the gap marking up to date as well?

> What branch can I use for that?

3809e

When these two items are done, I'll close this.

#99 - 01/23/2020 12:28 PM - Greg Shah

> Also I left a comment in the code which says that it's not clear if a temp-table inherits code page from a permanent table. I got controversial results while testing. We can return to it when we'll actually need it.

Please create a new task in the Database project to document this problem.

Stanislav: Please create this task. I'd like to close this.

#100 - 01/24/2020 01:42 PM - Stanislav Lomany

- Related to Bug #4520: Implement proper codepage inheritance from permanent to temp tables added

#101 - 01/24/2020 02:04 PM - Stanislav Lomany

Task #4520 was created for the inheritance from permanent to temp tables issue. This task can be closed.

#102 - 01/24/2020 05:53 PM - Greg Shah

- Status changed from WIP to Closed

Files

codepage-dyn.p	635 Bytes	06/24/2019	Stanislav Lomany
codepage.p	1.95 KB	06/27/2019	Stanislav Lomany
codepage-dyn.p	635 Bytes	06/27/2019	Stanislav Lomany
codepage-hierarchy.p	2.13 KB	06/27/2019	Stanislav Lomany
codepage-like.p	1.1 KB	06/27/2019	Stanislav Lomany
codepage-order.p	341 Bytes	06/27/2019	Stanislav Lomany
codepage-param.p	670 Bytes	06/27/2019	Stanislav Lomany
codepage-param-run.p	154 Bytes	06/27/2019	Stanislav Lomany
codepage-run.p	584 Bytes	06/27/2019	Stanislav Lomany
codepage-shared.p	1.15 KB	06/27/2019	Stanislav Lomany
codepage-shared-run.p	753 Bytes	06/27/2019	Stanislav Lomany