Database - Feature #2000

add conversion support for table and table handle parameters

02/15/2013 06:07 PM - Eric Faulhaber

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

Description

History

#1 - 02/19/2013 12:07 PM - Stanislav Lomany

- Status changed from New to WIP

Am I correct that the scope of this task is

DEFINE PARAMETER TABLE
DEFINE PARAMETER TABLE-HANDLE

TABLE function parameters

?

#2 - 02/19/2013 12:20 PM - Eric Faulhaber

Yes, that's correct.

#3 - 02/20/2013 03:55 PM - Stanislav Lomany

About TABLE parameters.

We must pass to the called function information that identifies the table and information whether APPEND option was specified in the calling procedure. As the table-identifying information I suggest to pass the target Buffer class. And to wrap this class together with APPEND flag using a wrapper, say, TableParameter:

Called procedure:

```
TempRecord1.Buf tt = TemporaryBuffer.define(TempRecord1.Buf.class, "tt", false);

public void execute(TableParameter ttParameter)
{
   externalProcedure(new Block()
   {
      public void body()
      {
         RecordBuffer.openScope(tt);
         useInputTableParameter(ttParameter, tt);
      }
   });
}
```

Calling function:

```
proc.execute(new TableParameter(TempRecord2.Buf.class, true));
```

Eric, what do you think?

#4 - 02/20/2013 05:02 PM - Eric Faulhaber

How do we convert this today? I vaguely remember dealing with this issue in the past, but I don't recall whether we dealt with it fully at the time.

#5 - 02/20/2013 05:30 PM - Greg Shah

We already support it today, as far as I understand. Remember that you implemented all the deep copy of temp-tables and so forth.

Majic uses things like: define input parameter table for tt-course.

I suspect we also handle FUNCTION my-func RETURNS INT (INPUT my-temp-table)...

I don't know if we handle APPEND, BIND or BY-VALUE.

I am sure we don't support TABLE-HANDLE today in either procs or functions.

#6 - 02/20/2013 05:48 PM - Stanislav Lomany

Heh, I didn't expected it to be converted...

It is converted to:

```
TempRecord1.Buf tt = TemporaryBuffer.define(TempRecord1.Buf.class, "tt", false);
```

```
   public void execute(final Temporary ttBuf2)
   {
      externalProcedure(new Block()
      {
         public void body()
         {
            RecordBuffer.openScope(tt);
            TemporaryBuffer.associate(ttBuf2, tt, true, true);
```

Procedure call:

```
ControlFlowOps.invokeWithMode("proc.p", "U", tt);
```

Notes: 1. Temporary should be at least Buffer 2. mode specified for invokeWithMode doesn't reflect APPEND append option, but anyway APPEND should be passed to TemporaryBuffer.associate, not the procedure invocation code.

#7 - 02/20/2013 05:56 PM - Stanislav Lomany

Conversion for TABLE-HANDLE parameters should probably look in the similar way:

```
public class Proc
{
   handle th;

   public void execute(final handle th)
   {
      externalProcedure(new Block()
      {
         public void body()
         {
            TemporaryBuffer.createDynamicTable(th, true, true);
            Proc.this.th = th;
```

(This example is not taking into consideration APPEND parameter)

#8 - 02/20/2013 06:37 PM - Eric Faulhaber

Stanislav Lomany wrote:

> Notes: 1. Temporary should be at least Buffer

OK, in that case I think we can have Temporary extend Buffer.

> 2. mode specified for invokeWithMode doesn't reflect APPEND append option, but anyway APPEND should be passed to TemporaryBuffer.associate, not the procedure invocation code.

I'm pretty sure we already support the APPEND option, even though it's not used in Majic.

#9 - 02/20/2013 06:41 PM - Eric Faulhaber

Stanislav Lomany wrote:

> Conversion for TABLE-HANDLE parameters should probably look in the similar way:
> [...]

This looks OK to me. I note that you are emitting TemporaryBuffer.createDynamicTable in Block.body instead of Block.init. I don't remember why we emit TemporaryBuffer.associate that way, but it strikes me as odd now.

#10 - 02/20/2013 07:04 PM - Stanislav Lomany

> I'm pretty sure we already support the APPEND option, even though it's not used in Majic.

APPEND option can be specified on the calling side as well as in the called side. At this point only conversion for called side is supported. So we should decide how to pass the parameter from the calling side.

#11 - 02/21/2013 02:16 PM - Stanislav Lomany

1.

> OK, in that case I think we can have Temporary extend Buffer.

Hmm, it seems that it makes *.Buf classes obsolete because they represent the mixture of Buffer and <? extends Temporary> interfaces.

2. We must make a decision on how APPEND parameter should be passed from the calling side.

3. BTW, why ControlFlowOps.invokeWithMode is used? As far as I tested, the calling mode should just match the one specified in DEFINE PARAMETER.

#12 - 02/21/2013 02:35 PM - Eric Faulhaber

Stanislav Lomany wrote:

1.

OK, in that case I think we can have Temporary extend Buffer.

Hmm, it seems that it makes *.Buf classes obsolete because they represent the mixture of Buffer and <? extends Temporary> interfaces.

Maybe I misunderstood your original comment "Temporary should be at least Buffer". I thought you meant the parameter, which is typed as Temporary today, needed to be Buffer due to some use inside the converted procedure. Please explain what you meant.

AFAIR, I just created Temporary as a marker interface for compile-time type safety, so maybe we can just set the type as Buffer instead and change the runtime signatures that accept Temporary accordingly.

2. We must make a decision on how APPEND parameter should be passed from the calling side.

I think we need Constantin or Greg's input here, since this is about RUN statement conversion for OUTPUT parameters.

3. BTW, why ControlFlowOps.invokeWithMode is used? As far as I tested, the calling mode should just match the one specified in DEFINE PARAMETER.

Again, I'll have to defer to Constantin on this one; this construct is new, AFAIK.

#13 - 02/21/2013 03:00 PM - Constantin Asofiei

2. We must make a decision on how APPEND parameter should be passed from the calling side.

I think we need Constantin or Greg's input here, since this is about RUN statement conversion for OUTPUT parameters.

From my testing, I don't remember finding any parameter validation based on APPEND clause for temp-tables. This is why this clause is not emitted in the parameter mode string. Are you saying that you can set the APPEND clause at the caller, but not at the proc/func definition (and viceversa)?

> 3. BTW, why ControlFlowOps.invokeWithMode is used? As far as I tested, the calling mode should just match the one specified in DEFINE PARAMETER.

> Again, I'll have to defer to Constantin on this one; this construct is new, AFAIK.

The reason is that all procedures are now invoked using ControlFlowOps APIs, as 4GL has no parameter validation at compile time for the RUN statement. After it decided which procedure/external program it needs to invoke, it will try to map the passed arguments with the defined parameters. Thus, the modes are emitted to be able to validate them with the procedure/function definition.

#14 - 02/21/2013 03:20 PM - Stanislav Lomany

> Maybe I misunderstood your original comment "Temporary should be at least Buffer". I thought you meant the parameter, which is typed as Temporary today, needed to be Buffer due to some use inside the converted procedure. Please explain what you meant.
>
> AFAIR, I just created Temporary as a marker interface for compile-time type safety, so maybe we can just set the type as Buffer instead and change the runtime signatures that accept Temporary accordingly.

Never mind, we can leave the things as is or, as you suggested, change it to Buffer for conformity with other types of parameters, like BUFFER. I do not expect any other use in the converted procedure.

#15 - 02/21/2013 03:36 PM - Stanislav Lomany

> From my testing, I don't remember finding any parameter validation based on APPEND clause for temp-tables. This is why this clause is not emitted in the parameter mode string. Are you saying that you can set the APPEND clause at the caller, but not at the proc/func definition (and

viceversa)?

Progress documentation tells that "You can only use the APPEND option for an INPUT parameter in the calling procedure or an OUTPUT parameter in the called procedure". Actually it is vice versa, but anyway, yes, you can specify or not specify APPEND in the calling procedure (Progress 9.1E):

```
RUN proc.p (OUTPUT TABLE tt APPEND).
RUN proc.p (OUTPUT TABLE tt).
```

#16 - 02/22/2013 12:34 PM - Stanislav Lomany

So, Eric, do you prefer to leave Temporary or switch to Buffer?

Constantin, Greg, how do you prefer to pass APPEND option?

#17 - 02/22/2013 12:36 PM - Eric Faulhaber

Leave it as Temporary if you don't have a need to change it. Less runtime change. If we find we need it to be Buffer later, we can change it then.

#18 - 02/22/2013 12:44 PM - Greg Shah

> Progress documentation tells that "You can only use the APPEND option for an INPUT parameter in the calling procedure or an OUTPUT parameter in the called procedure". Actually it is vice versa, but anyway, yes, you can specify or not specify APPEND in the calling procedure (Progress 9.1E):

Before we decide on this, let's determine if specifying APPEND in the caller actually makes a difference in behavior. Progress is full of cases where there is optional syntax. For example, if you specify it in the caller but the called procedure does not specify it, does append mode actually get used? Please fill out this matrix:

```
-------------------------------------------------------------------------------------------------------------
--------
|                                  | Called Program Specifies APPEND        | Called Program DOESN'T Specify A
PPEND  |
|----------------------------------|----------------------------------------|---------------------------------
-------|
| Caller Specifies APPEND          |                                        |
       |
|----------------------------------|----------------------------------------|---------------------------------
-------|
| Caller DOESN'T Specify APPEND    |                                        |
       |
-------------------------------------------------------------------------------------------------------------
--------
```

In which cases does APPEND actually get honored?

#19 - 02/22/2013 01:55 PM - Stanislav Lomany

Please fill out this matrix:

When you specify APPEND in the calling procedure it logically applies to the table in the calling procedure (tt in this case):

```
RUN proc.p (OUTPUT TABLE tt APPEND).
```

In the calling procedure APPEND can be applied to OUTPUT table only.

When you specify APPEND in the called procedure it logically applies to the table in the called procedure (tt2 in this case):

```
DEFINE INPUT PARAMETER TABLE FOR tt2 APPEND.
```

In the called procedure APPEND can be applied to INPUT table only.

For INPUT-OUTPUT tables you can specify APPEND on both sides.

In any case APPEND in the calling procedure and APPEND in the called procedure are *different* options targeted to different tables.

#20 - 02/22/2013 10:47 PM - Eric Faulhaber

Stas: please provide an update for all the other functionality discussed above ASAP, even if you don't yet have an answer on the APPEND in the RUN statement for OUTPUT parameters. We need to integrate this functionality and begin conversion of the customer project immediately.

Constantin: what do you think is the best way to convert the APPEND option in the RUN statement, as discussed above for OUTPUT parameters?

#21 - 02/23/2013 03:00 AM - Constantin Asofiei

Stanislav, OK now I understand what you mean. APPEND clause used with OUTPUT/INPUT-OUTPUT table at the caller affects the source table used by the caller, and APPEND clause used with INPUT/INPUT-OUTPUT table at the parameter definition affects the target table. The problem here is that we don't know which table uses the target procedure/function, so we can't do anything before the actual procedure is called. Thus, I think we need to use an approach similar to how the output/input-output extent parameters are converted, and send the source table's append clause to the called procedure; the good news is that the mode must be the same, so the conversion rules can use the table parameter mode set at the definition. We will not emit Temporary parameters for the function/procedures, instead we will convert them as a TableParameter. When a procedure/function is called, the table parameter will be emitted as a new TableParameter(<table>, <append>) . I don't think this will be as complex as the extent clause (where we needed to emit inner classes to get/set the source array variable), but you will need to emit some processing code in the Block.init clause,

depending on the target table parameter definition (input/input-output and append clause). The runtime will take care of the APPEND clause set at the source table. Also, you will need a TableParameter.getTable emitted in Block.init, to save the DMO in a instance field.

#22 - 02/23/2013 03:49 AM - Stanislav Lomany

- File svl_upd20130223a.zip added

I'm sending the update in its current state: INPUT/OUTPUT parameters and APPEND option are not handled, createDynamicTable should be moved to init(). TABLE-HANDLE parameters should be converted without crashes.

#23 - 02/26/2013 01:19 PM - Greg Shah

- File progress.g added

Attached is the parser change to accept the APPEND syntax on a function call.

#24 - 02/26/2013 03:34 PM - Stanislav Lomany

- File svl_testcases20130226a.zip added

#25 - 02/26/2013 03:44 PM - Stanislav Lomany

- File svl_test20130226a.zip added

I'll add javadocs later.

#26 - 02/26/2013 04:00 PM - Eric Faulhaber

Stanislav Lomany wrote:

> What remains is:
> 1. procedure TABLE-HANDLE calls using TableParameters, calling side - probably, not a problem

You are looking at this one, right?

> 2. function TABLE calls using TableParameters, calling side - problem with APPEND option since I have to dig grammar file.

Greg provided the parser update for this one. What's left to do?

> 3. problem with referencing parameters in inner classes of the called function - can't say if it is a problem or not.

Based on your earlier email, this one is working now, right?

> 4. SHARED handles are not supported for TABLE-HANDLE FOR - probably a problem.

Which test case goes with this issue? th-proc.p?

#27 - 02/26/2013 04:08 PM - Stanislav Lomany

#1 - yes, working on it
#2 - emit TableParameter when calling a function, working on it
#3 - yes, it works fine
#4 - yes, th-proc.p

#28 - 02/26/2013 04:09 PM - Eric Faulhaber

Did you intend to remove this from the define_stmt rule in progress.g?

```
Aast child = (Aast) ##.getFirstChild();
while (child != null)
{
   if (child.getType() == KW_TAB_HAND)
      setOpts = false;

   child = (Aast) child.getNextSibling();
}
```

#29 - 02/26/2013 04:12 PM - Stanislav Lomany

> Did you intend to remove this from the define_stmt rule in progress.g?

Yes. I don't know why it was there, probably to work around conversion fail when we wasn't supporting table handles.

#30 - 02/26/2013 04:37 PM - Eric Faulhaber

In record_scoping.rules, you changed the bufclassname annotation to TableParameter from Temporary. Isn't that going to break the places we emit calls to the following old methods?

```
TemporaryBuffer.associate(Temporary, Temporary, boolean, boolean)
TemporaryBuffer.associate(Temporary, Temporary, boolean, boolean, boolean)
```

#31 - 02/26/2013 04:43 PM - Stanislav Lomany

> In record_scoping.rules, you changed the bufclassname annotation to TableParameter from Temporary. Isn't that going to break the places we emit calls to the following old methods?

There should be no converted calls to this functions. Since TABLE and TABLE-HANDLE are interchangable, parameters of these types should be passed using TableParameter.

#32 - 02/26/2013 04:47 PM - Greg Shah

In regards to issue number 4, can you explain what the code in th-proc.p is supposed to do?

```
DEFINE INPUT PARAMETER TABLE-HANDLE FOR f-h-in.
DEFINE OUTPUT PARAMETER TABLE-HANDLE FOR f-h-out.
DEFINE INPUT-OUTPUT PARAMETER TABLE-HANDLE FOR f-h-inout APPEND.

DEFINE INPUT PARAMETER TABLE-HANDLE FOR sh-f-h-in.
DEFINE OUTPUT PARAMETER TABLE-HANDLE FOR sh-f-h-out.
DEFINE INPUT-OUTPUT PARAMETER TABLE-HANDLE FOR sh-f-h-inout APPEND.
```

This code seems like it would hide the local vars with the same name.

What happens in the 4GL and what is the problem in P2J?

I have to step out for a few hours, but I can try to look at this later tonight.

#33 - 02/26/2013 04:56 PM - Eric Faulhaber

Stanislav Lomany wrote:

> > In record_scoping.rules, you changed the bufclassname annotation to TableParameter from Temporary. Isn't that going to break the places we emit calls to the following old methods?
>
> There should be no converted calls to this functions. Since TABLE and TABLE-HANDLE are interchangable, parameters of these types should be passed using TableParameter.

What about BUFFER parameters on temp-tables?

The templates associate_temp_tables and associate_temp_tables_append are still being grafted around line 335 in your version of buffer_definitions.rules. These will emit calls to those old TemporaryBuffer.associate methods.

#34 - 02/26/2013 04:58 PM - Stanislav Lomany

> What about BUFFER parameters on temp-tables?

Yes, I missed this type of parameters when I was analyzing the impact. I'll fix it.

#35 - 02/26/2013 05:28 PM - Eric Faulhaber

Stanislav Lomany wrote:

> > What about BUFFER parameters on temp-tables?
>
> Yes, I missed this type of parameters when I was analyzing the impact. I'll fix it.

Hm...actually, a BUFFER parameter on a temp-table won't cause a call to TemporaryBuffer.associate. Is there still a path to emit those old associate calls?

#36 - 02/26/2013 05:33 PM - Stanislav Lomany

> Hm...actually, a BUFFER parameter on a temp-table won't cause a call to TemporaryBuffer.associate. Is there still a path to emit those old associate calls?

It looks like I'm ruined conversion of BUFFER parameters. So if you don't see TemporaryBuffer.associate with my update applied that can be a bug.

#37 - 02/26/2013 06:08 PM - Stanislav Lomany

Heh, it look like BUFFER parameter is misconverted, but not because of me:

```
def temp-table tt field f1 as integer.
```

```
define parameter buffer buf for book.
define parameter buffer tt1 for tt.
```

Is converted to:

```
 public void execute(final Book.Buf bufbuf, final TempRecord1.Buf tt1tt1)
```

It's to late for me to pin down the reason so please someone check and distribute the information.

> Is there still a path to emit those old associate calls?

associate is emitted if parent.type == prog.kw_table. For BUFFER calls the tree is a bit different. So the question is open, I leave everything as is.

#38 - 02/26/2013 06:53 PM - Stanislav Lomany

> In regards to issue number 4, can you explain what the code in th-proc.p is supposed to do?
> This code seems like it would hide the local vars with the same name.
> What happens in the 4GL and what is the problem in P2J?

TABLE-HANDLE FOR assigns a value to an existing variable rather creates a new one. For non-shared variables these conditions are equivalent and there was no need to make any difference in P2J conversion, but for shared variables the value assigned by TABLE-HANDLE FOR can be obtained outside the calling procedure and therefore [NEW] SHARED variable should be declared using SharedVariableManager.

#39 - 02/26/2013 07:23 PM - Stanislav Lomany

- File database_references.rules added

Fix for #2.

#40 - 02/27/2013 12:11 AM - Eric Faulhaber

Stanislav Lomany wrote:

> Heh, it look like BUFFER parameter is misconverted, but not because of me:
>
> [...]

Fixed, see #2062.

#41 - 02/27/2013 02:07 PM - Stanislav Lomany

- File svl_upd20130226a.zip added

Everything except SHARED variables for TABLE-HANDLE FOR. I recommend to distribute the remaining part later as a separate update.

#42 - 02/27/2013 02:10 PM - Eric Faulhaber

Have you run conversion on Majic with this to check for unexpected changes?

#43 - 02/27/2013 02:17 PM - Stanislav Lomany

> Have you run conversion on Majic with this to check for unexpected changes?

No. I can run, but it will take 2.5 h.

#44 - 02/27/2013 02:29 PM - Eric Faulhaber

I'll do it here. Do you expect any differences? If I find anything unusual, I'll ask.

#45 - 02/27/2013 03:08 PM - Stanislav Lomany

> I'll do it here. Do you expect any differences? If I find anything unusual, I'll ask.

TABLE parameters should use TableParameters now.

#46 - 02/27/2013 03:47 PM - Eric Faulhaber

Majic did not compile after this update. Please see #2067 for the results.

#47 - 02/27/2013 04:37 PM - Eric Faulhaber

Actually, I think we're OK here. All the failures are in the Config class, which is hand-written and will need to be updated to accept TableParameter instead of Temporary. Go ahead and check this update into bzr and distribute. After you finish with the last feature (as a separate update), please fix up the Config class in Majic and post that to #2067.

#48 - 02/27/2013 05:04 PM - Greg Shah

Please distribute the Config change as a separate update zip file BUT at the SAME TIME as the svl_upd20130226a.zip. The reason we cannot wait is because there are many people on the team that are all trying to run conversion testing against the bzr head + some changes. If you check in your change and it breaks Majic, then we can no longer do conversion testing in the normal way and everyone will have a real mess on their hands.

#49 - 02/27/2013 05:13 PM - Stanislav Lomany

> Please distribute the Config change as a separate update zip file BUT at the SAME TIME as the svl_upd20130226a.zip. The reason we cannot wait is because there are many people on the team that are all trying to run conversion testing against the bzr head + some changes. If you check in your change and it breaks Majic, then we can no longer do conversion testing in the normal way and everyone will have a real mess on their hands.

OK, I'll quickly make the Config update (I guess you noticed that I checked in svl_upd20130226a).

#50 - 02/27/2013 05:17 PM - Stanislav Lomany

I uncommited my last commit.

#51 - 03/04/2013 05:39 AM - Stanislav Lomany

I found that DEF PARAMETER TABLE-HANDLE FOR th where th is a [new] shared variable is handled differently that if th is a regular variable:

1.
OUTPUT case will produce compile-time error.

```
DEFINE SHARED VARIABLE th AS HANDLE.

DEFINE OUTPUT PARAMETER TABLE-HANDLE FOR th.
```

2.
In INPUT-OUTPUT case an invalid handle value (negative integer) will be produced as the output value of the procedure. Value of the shared variable will be correct.

```
DEF NEW SHARED VARIABLE th AS HANDLE.
DEF NEW SHARED TEMP-TABLE tt FIELD f1 AS INTEGER.

DEF VAR th2 AS HANDLE.
```

```
RUN proc.p(INPUT-OUTPUT TABLE-HANDLE th2).

MESSAGE (IF th = ? THEN "?" ELSE STRING(th)) +
         " " +
        (IF th2 = ? THEN "?" ELSE STRING(th2)).
```

proc.p:

```
DEFINE SHARED VARIABLE th AS HANDLE.
DEFINE SHARED TEMP-TABLE tt FIELD f1 AS INTEGER.

DEFINE INPUT-OUTPUT PARAMETER TABLE-HANDLE FOR th.

th = TEMP-TABLE tt:HANDLE.
```

Output:

```
9203 -62557
```

As the current solution, value of the handle passed to the procedure will be assigned to the shared parameter variable at the beginning of the procedure.

#52 - 03/04/2013 09:31 AM - Greg Shah

Is there any difference in behavior if you switch the order of the DEF SHARED VAR and DEF PARAMETER (if the DEF SHARED VAR comes 2nd)?

#53 - 03/04/2013 09:33 AM - Stanislav Lomany

> Is there any difference in behavior if you switch the order of the DEF SHARED VAR and DEF PARAMETER (if the DEF SHARED VAR comes 2nd)?

DEF PARAMETER FOR cannot come first because it should reference an already declared variable.

#54 - 03/04/2013 09:37 AM - Greg Shah

Got it. I was mis-reading the meaning of the syntax.

#55 - 03/07/2013 04:44 AM - Stanislav Lomany

About DEF PARAMETER TABLE-HANDLE FOR th where th is a regular variable:

1. OUTPUT case will produce compile-time error.
2. In INPUT-OUTPUT case a valid handle will be produced as the output value of the procedure.

The current solution will be the same as in the non-FOR case.

#56 - 03/07/2013 08:46 AM - Greg Shah

When will the solution be ready for review?

#57 - 03/07/2013 08:48 AM - Stanislav Lomany

> When will the solution be ready for review?

Hopefully will finish it today.

#58 - 03/08/2013 12:20 PM - Stanislav Lomany

About the state of work: I changed the parser so TABLE-HANDLE FOR is recognized as a variable reference rather than a variable definition. Then I basically needed to handle conversion of regular variables (converted in the same way as if no FOR option is specified) and shared variables which are converted differently. I had three options:
1. make parser recognize if it is a shared and or regular variable, ignore FOR option for regular variables and implement conversion for shared variables only;
2. leave parser alone, but instead rewrite generated nodes in the similar manner on annotations stage;
3. handle either shared and regular variables as variable references, implement conversion for shared variables, correct conversion for regular variables.

I chose the last option, but unfortunately it turned out technically more trickier than I've expected because of conversion for regular variables (you don't have to work on it into options #1 or #2).

I don't have any problems with solution, I'm just trying to explain why I need more time that I've expected. If you can say that sure I had to go with the #1 or #2 instead that is an option too.

#59 - 03/11/2013 02:32 PM - Stanislav Lomany

- File svl_upd20130311a.zip added

Update for TABLE-HANDLE FOR case. Should I regression-test it?

#60 - 03/11/2013 02:34 PM - Stanislav Lomany

- File svl_upd20130311a.zip added

#61 - 03/11/2013 02:59 PM - Greg Shah

Is this merged up to the latest bzr revision?

If so, please do run conversion regression testing. Eric will do a code review shortly.

If not yet merged, please merge and upload the merged result.

#62 - 03/11/2013 03:42 PM - Stanislav Lomany

> Is this merged up to the latest bzr revision?

Yes, it is merged.

#63 - 03/11/2013 06:20 PM - Stanislav Lomany

Regression conversion completed without errors.

#64 - 03/11/2013 10:15 PM - Eric Faulhaber

Code looks good, AFAICT, but without test cases and being able to look at the ASTs, the changes were hard to follow, especially in variable_definitions.rules. You must have test cases?

Go ahead and check it in and distribute, and please post and check in your test cases, too.

#65 - 03/12/2013 05:22 AM - Stanislav Lomany

- File svl_testcases20130311a.zip added

#66 - 03/21/2013 09:51 AM - Stanislav Lomany

- Status changed from WIP to Review

#67 - 03/22/2013 07:38 AM - Constantin Asofiei

Stanislav: did this convert properly after you added TABLE-HANDLE parameter support?

```
def output param table-handle h.
```

Now it converts to this:

```
   public void execute(final TableParameter _h)
   {
      externalProcedure(new Block()
      {
         handle h = _h.getTableHandle();

         public void init()
         {
            h.assign(new handle(), TemporaryBuffer.createDynamicTable(_h, false, true));
            TransactionManager.registerUndo(h);
         }

         public void body()
         {
         }

      });
   }
```

Can you point me how the converted code should look?

#68 - 03/22/2013 08:15 AM - Stanislav Lomany

It has been regressed since revision 10279.

Correct output is:

```
public void execute(final TableParameter _h)
   {
      externalProcedure(new Block()
      {
         handle h = _h.getTableHandle();

         public void init()
         {
            h.assign(new handle());
            TransactionManager.registerUndo(h);
         }

         public void body()
         {
            TemporaryBuffer.createDynamicTable(_h, false, true);
         }
      });
   }
```

Testcases for table handles are named th-* in bzr.

#69 - 03/22/2013 08:55 AM - Constantin Asofiei

OK, I'll look at it.

#70 - 03/22/2013 09:11 AM - Constantin Asofiei

- File ca_upd20130322g.zip added

Fixes a regression in the TABLE-HANDLE parameter conversion, introduced by 10293.

#71 - 03/22/2013 09:15 AM - Greg Shah

The code looks fine. If it passes testing, check it in and distribute it.

#72 - 03/22/2013 10:24 AM - Constantin Asofiei

ca_upd20130322g.zip committed to bzr revision 10214.

#73 - 03/23/2013 10:10 PM - Eric Faulhaber

- Status changed from Review to Closed

- % Done changed from 0 to 100

#74 - 06/26/2013 01:54 PM - Constantin Asofiei

Stanislav, any idea why this code converts badly?

```
def temp-table tt1 field f1 as int field f2 as char.
procedure proc_table_input.
   def input param table for tt1.
   def var ch as char.

   ch = "".
   for each tt1:
      ch = ch + string(tt1.f1) + " " + string(tt1.f2).
   end.

   return ch.
end.
```

converts to (notice how the tt1Buf2 name is used in the query instead of tt1):

```
   public void procTableInput(final TableParameter tt1Buf2)
   {
      internalProcedure(new Block()
      {
         character ch = new character("");

         public void init()
         {
            TransactionManager.registerUndo(ch);
         }

         public void body()
         {
            TemporaryBuffer.associate(tt1Buf2, tt1, true, false);
            ch.assign("");

            forEach("loopLabel0", new Block()
            {
               AdaptiveQuery query0 = null;

               public void init()
               {
                  RecordBuffer.openScope(tt1Buf2);
                  query0 = new AdaptiveQuery(tt1Buf2, (String) null, null, "tt1Buf2.id asc");
               }
```

```
               public void body()
               {
                  query0.next();
                  ch.assign(concat(ch, valueOf(tt1Buf2.getF1()), " ", valueOf(tt1Buf2.getF2())));
               }
            });

            returnNormal(ch);
         }
      });
   }
```

#75 - 06/27/2013 07:52 AM - Stanislav Lomany

Constantin, I looked at it and have no quick idea how to fix it. It looks like TableParameters-related code just wasn't tested on internal procedures. If it is a prioritized issue, I can fix it (I don't think it will take much time).

#76 - 06/27/2013 07:55 AM - Constantin Asofiei

Stanislav: this is from one of my tests, I think is safe to put it in your low priority queue, as it doesn't affect the server code.

#77 - 11/16/2016 11:06 AM - Greg Shah

- Target version changed from Milestone 4 to Conversion Support for Server Features

#78 - 10/02/2017 11:56 PM - Eric Faulhaber

The issue reported in #2000-74 seems to be fixed now. The converted output of the internal procedure is:

```
   public void procTableInput(final TableParameter tt1Buf2)
   {
      character ch = UndoableFactory.character();

      internalProcedure(new Block((Body) () ->
      {
         TemporaryBuffer.associate(tt1Buf2, tt1, true, false);
         ch.assign("");

         AdaptiveQuery query0 = new AdaptiveQuery();

         forEach("loopLabel0", new Block((Init) () ->
         {
            RecordBuffer.openScope(tt1);
            query0.initialize(tt1, ((String) null), null, "tt1.id asc");
         },
         (Body) () ->
         {
            query0.next();
            ch.assign(concat(ch, valueOf(tt1.getF1()), " ", valueOf(tt1.getF2())));
         }));

         returnNormal(ch);
      }));
   }
```

Note that the name tt1 is now used for the query instead of tt1Buf2. As I understand it, this (i.e., tt1Buf2 being used incorrectly for the query) was the issue.

Stanislav, Constantin, please confirm whether this conversion looks correct now.

#79 - 10/03/2017 05:45 AM - Stanislav Lomany

Yes, it's correct.

Files

svl_upd20130223a.zip	309 KB	02/23/2013	Stanislav Lomany
progress.g	1.19 MB	02/26/2013	Greg Shah
svl_testcases20130226a.zip	1.14 KB	02/26/2013	Stanislav Lomany
svl_test20130226a.zip	338 KB	02/26/2013	Stanislav Lomany
database_references.rules	34.4 KB	02/27/2013	Stanislav Lomany
svl_upd20130226a.zip	346 KB	02/27/2013	Stanislav Lomany
svl_upd20130311a.zip	367 KB	03/11/2013	Stanislav Lomany
svl_upd20130311a.zip	367 KB	03/11/2013	Stanislav Lomany
svl_testcases20130311a.zip	1.28 KB	03/12/2013	Stanislav Lomany
ca_upd20130322g.zip	13.4 KB	03/22/2013	Constantin Asofiei