Database - Support #6714

check if there is an optimization opportunity when copying temp-tables based on multiplex id

08/25/2022 12:24 PM - Greg Shah

Status:	Review	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Dănuț Filimon	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to Database - Bug #6997: OUTPUT BY-REFERENCE records are not cleared	**New**

History

#1 - 08/25/2022 12:30 PM - Greg Shah

It seems to me that when processing certain temp-table copying use cases, there may be an opportunity for optimization. We currently use a multiplex id to separate out records for different instances. We know there are cases where we overwrite the contents of one table from another while then deleting the source table. For example, an output temp-table parameter will do this "overwrite and delete source".

I was wondering if we already handle this in an optimized fashion. For example, we could just change the multiplex id of the caller's temp-table to match the data that is already there.

In other cases, (e.g. an input-output parameter case) where it isn't a full overwrite, perhaps there is a possible optimization to just modify the source table's multiplex id to match the target table's id without otherwise copying/inserting records.

If we've already done such things and/or my ideas are naive, it is OK to say so. I'm coming at this without having checked the implementation.

#2 - 10/06/2022 02:13 AM - Alexandru Lungu

- Status changed from New to WIP

- Assignee set to Dănuț Filimon

#3 - 10/13/2022 04:37 AM - Alexandru Lungu

Danut, you can try using some of the following scenarios to have a comprehensive context of where such optimization may fit:

- temp-table parameter to non-persistent external procedure (run "side.p"(input-output table tt).)
- temp-table parameter to persistent external procedure (run "side.p" persistent set hProc (input-output table tt).)
- temp-table parameter to persistent internal procedure (run "intern" in hProc (input-output table tt).)
- temp-table parameter to internal procedure (run "intern" input-output table tt).)
- check APPEND specification (run "side.p"(input-output table tt append).)
- check BY-REFERENCE specification (run "side.p"(input-output table tt by-reference).)
- check BIND specification (run "side.p"(input-output table tt bind).)
- the scenarios above with output instead of input-output
- at the end, check that COPY-TEMP-TABLE and EMPTY-TEMP-TABLE continue working as normally (nothing is deleted/appended unexpectedly)

#4 - 10/19/2022 07:42 AM - Dănuț Filimon

I have created tests with the mentioned specifications and all passed (same results were obtained from the executed statements when comparing 4GL results of the same test). I am working on expanding the test cases to make sure that the correct statements are executed and that the multiplex values of the temp-tables are flagged correctly (empty / not empty).

The current tests I am working on are:

- passing a table as an INPUT-OUTPUT/OUTPUT parameter between multiple persistent / non-persistent internal / external procedure, this includes the APPEND, BY-REFERENCE, BIND, EMPTY TEMP-TABLE, COPY-TEMP-TABLE tests.
- using APPEND BY-REFERENCE
- using both NO-UNDO (currently) and UNDO temp-tables
- using TABLE-HANDLE

#5 - 10/27/2022 06:09 AM - Dănuț Filimon

- File test-byref.7z added

- File 6714.patch added

Apart from the tests mentioned in #6714-4, I have also made tests in which the called external procedure passes the table using an internal procedure It revealed problems that needed to be patched. I attached a proposed patch (6714.patch) for this issue. This patch was tested on a larger application and I didn't discover any problems while browsing. Please review.

While working on this issue, I discovered a small issue regarding BY-REFERENCE keyword. When I was comparing the results of the tests using **meld**, I found out that records are not deleted from non-persistent procedures. Using the most recent version of 3821c and a 3-way comparison in meld revealed that the same results are obtained in both versions (recent 3821c and my modified version for this issue) and at the same time, the results are different from what the 4GL code outputs.

test-byref.7z contains a simple example that uses BY-REFERENCE.

#6 - 10/27/2022 08:03 AM - Greg Shah

Ovidiu: Please review.

#7 - 11/16/2022 05:47 AM - Dănuț Filimon

Any news about the review?

#8 - 12/09/2022 11:52 PM - Ovidiu Maxiniuc

Dănuț,

I am sorry for the delay, somehow the notification emails got lost at the bottom of the list and I only saw it now.

The changes are not extensive but I am not sure I fully understand all the aspects of it. For example, at the end of TemporaryBuffer.associate(), for the output parameters the OutputTableCopier is constructed with the additional parameter, shouldInsert. Particularly, you force all records from after tables to be inserted. I do not say this is incorrect but a bit surprising: I do not expect those to be different from SIMPLE buffers (that is those which do not have a pair BEFORE buffer, even if they are members of a dataset).

In the new method performOutputCopy(TableParameter outputParameter, boolean shouldInsert), I see that, based on the above parameter, the records are removed. I think the second removeRecords which does not enforce the records to be deleted could be call on else branch, since the matching records are already 'moved' to targetMultiplex table.

You mentioned that you *"discovered a small issue regarding BY-REFERENCE"*, but I did not see anything to relate the changes to such parameters. Have you done tests without this option?

I think the update is a bit risky and I recommend putting it to test with a project which uses table parameters and especially sent BY-REFERENCE.

Eric,
which would be the most appropriate one? The old testsets from devsrv01?

#9 - 12/14/2022 05:47 AM - Dănuț Filimon

- Related to Bug #6997: OUTPUT BY-REFERENCE records are not cleared added

#10 - 12/14/2022 05:55 AM - Dănuț Filimon

Originally, my intention was to delete the records from a table based on the targetMultiplex when the outputParameter wasn't in append mode and then update the records by replacing the multiplex with targetMultiplex. I retest the patch and from a total of 108 tests that I used when working on this issue, a total of 12 failed. I looked a bit into the problem and it seems that the second removeRecords call returns because the table is empty which should not be.

Ovidiu Maxiniuc wrote:

> You mentioned that you "discovered a small issue regarding BY-REFERENCE", but I did not see anything to relate the changes to such parameters. Have you done tests without this option?

I've done tests without this option and found no problems at the time. Only 8 tests from my test set turned out to have an issue with BY-REFERENCE. Please note that my changes did not cause this problem, I only happened to find it by chance while working on the current issue. I created #6997 and explained the issue with BY-REFERENCE, please take a look at it.

> I think the update is a bit risky and I recommend putting it to test with a project which uses table parameters and especially sent BY-REFERENCE.

I agree. Since the patch is not working anymore, I will continue working on it.

#11 - 12/16/2022 08:06 AM - Dănuț Filimon

I've made a mistake and used the wrong patch when testing previously. The second removeRecords call didn't execute because the flags were set incorrectly in the patch that I had. I can confirm that the attached patch (6714.patch) still works correctly as it used to. We can proceed with testing the changes.

#12 - 01/10/2023 09:18 AM - Dănuț Filimon

I've made a comparison between **3821c/rev.14467** and the patched version from #6714-5 using different **patched versions of h2** (revision 6 - current, 7 and 8). I changed my tests to only check their execution time (this included create statements) and continued with a single test. The test was modified further and since measuring sql statements time was introduced recently, it was possible to filter out the create statements. I wrote a small script to calculate the time executed by each statement type (update, delete, insert) from the data collected.

Test specifications:

- temp-table parameter to non-persistent external procedure (using INPUT-OUTPUT) which is then passed to another non-persistent external procedure.
- 10000 initial create statement in each file (start.p and external procedures)
- 100 iterations of the test

Here are the results of the comparison, the value in the table is the sum of all operations measured in milliseconds:

Patched H2	3821c-14467-patched (ms)	3821c-14467 (ms)	Difference
h2-1.4.200-6.jar	25483.643	29372.805	-13.24%
h2-1.4.200-7.jar	25271.644	28445.966	-11.16%
h2-1.4.200-8.jar	23579.766	25734.358	-8.37%

Here is another table with a close lookup at each operation total execution time.

	h2-1.4.200-6.jar		h2-1.4.200-7.jar		h2-1.4.200-8.jar	
	3821c-14467-patched (ms)	3821c-14467 (ms)	3821c-14467-patched (ms)	3821c-14467 (ms)	3821c-14467-patched (ms)	3821c-14467 (ms)
insert	8380.807	18548.934	8552.507	17701.435	8423.291	16631.863
delete	5229.992	10823.871	5117.743	10744.53	5038.871	9102.495
update	11872.844	0	11601.393	0	10117.603	0

#13 - 01/11/2023 04:35 AM - Alexandru Lungu

Danut, can you add a counter JMX for your modification? I think we can benefit from finding out how many times this scenario is reached. Consider re-posting the patch over 6129b integrating the JMX.

#14 - 01/11/2023 06:23 AM - Dănuț Filimon

- File 6714-6129b.patch added

There already is a JMX for OutputTableCopier with the same name and it measures the execution time and call count of the performOutputCopy method. I made a patch with the changes for 6129b after solving a minor conflict.

#15 - 01/12/2023 09:33 AM - Constantin Asofiei

Some concerns on moving the source records to the destination table just by updating the multiplex ID to the destination table:

- of course, this works only if the same SQL table is at both the src and dest
- Alexandru notes in #7026-12 that:

  However, it is restricted only to the cases where the output parameter is part of a non-persistent external procedure, which is going to empty the multiplex (if there are any concerns on this, we can continue discussion on #6714). Anyways, I used the output table copier JMX and the POC tests trigger less than 10 such operations that are resolved in less than 1ms in total. Therefore, I won't go ahead with profiling and integrating it with 6129b effort.

This is a problem: you can't assume that a non-persistent procedure exists only once on the stack. You can have multiple internal procedures/functions for the same non-persistent external program active on the stack, at the same time.

#16 - 01/16/2023 07:08 AM - Alexandru Lungu

Constantin Asofiei wrote:

> Some concerns on moving the source records to the destination table just by updating the multiplex ID to the destination table:
>
> - of course, this works only if the same SQL table is at both the src and dest
> - Alexandru notes in #7026-12 that:
>
>   However, it is restricted only to the cases where the output parameter is part of a non-persistent external procedure, which is going to empty the multiplex (if there are any concerns on this, we can continue discussion on #6714). Anyways, I used the output table copier JMX and the POC tests trigger less than 10 such operations that are resolved in less than 1ms in total. Therefore, I won't go ahead with profiling and integrating it with 6129b effort.
>
> This is a problem: you can't assume that a non-persistent procedure exists only once on the stack. You can have multiple internal procedures/functions for the same non-persistent external program active on the stack, at the same time.

AFAIK, only the OutputTableCopier associated with an external block of a non-persistent program triggers an "update-multiplex" type of copy. How does internal procedures/functions affect this process? I suspect that once the external block finishes, there can't be any internal procedures/functions of that exact program instance still on the stack.

#17 - 01/20/2023 10:34 AM - Alexandru Lungu

- % Done changed from 0 to 100

- Status changed from WIP to Review

#18 - 01/23/2024 04:36 PM - Eric Faulhaber

Can this be closed? Is there anything to be reviewed and used from this task, or has it been absorbed/superseded by similar tasks?

#19 - 01/25/2024 07:51 AM - Dănuț Filimon

Eric Faulhaber wrote:

> Can this be closed? Is there anything to be reviewed and used from this task, or has it been absorbed/superseded by similar tasks?

The changes were tested but it handled a very particular case. Over the time, multiple changes were made to H2 and it is possible that the changes might not be useful. I'll retest the changes using POC and return with an appropriate answer.

#20 - 01/26/2024 05:38 AM - Dănuț Filimon

I retested the changes (the patch from #6714-14) but had to make two small adjustments:

- The usage of the ProcedureManager where the OutputTableCopier is instantiated.
- The multiplex argument which is now an argument for a prepared statement and not added as it is.

I ran the POC performance tests and obtained an improvement but after introducing the JMXs to get a few counters and timers, I found out that OutputTableCopier is not used at all so the POC results are invalid.

After testing another large application using JMXs, I got the following results:

Case	Count
copyAllRows (default)	5362
only update	6
delete-update	37

shouldInsert	Count	Total Call Time (ms)	Avg call time (ms)
true	5510	1140	0.2068
false	43	4	0.093

There were a total of **5553** calls to performOutputCopy where **5510** were with the default logic, and only **43** were using delete-update (some calls were already processed and short-circuited: **148**). Only **0.774%** of the total performOutputCopy call used this optimization, while the average call time can't be used because of the low number of calls. In my opinion, unless we can expand the area where delete-update is used, we should not proceed with the changes and close/reject this issue.

Files

6714.patch	12.1 KB	10/27/2022	Dănuț Filimon
test-byref.7z	658 Bytes	10/27/2022	Dănuț Filimon
6714-6129b.patch	11.4 KB	01/11/2023	Dănuț Filimon