Database - Bug #8196

Reduce number of AbstractTempTable._hasRecords calls to avoid BufferManager.activeBuffers

01/15/2024 04:18 AM - Dănuț Filimon

Status:	Test	**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		

Description

Related issues:

Related to Database - Bug #7167: Associating records from opened buffers to n...	**Closed**

History

#1 - 01/15/2024 04:25 AM - Dănuț Filimon

While retesting different session reclaiming lifespans for #7167, it was noted that there is almost no difference between "No records to associate" and "Records associated" counts (note the table below that was copied from #7167-10).

Counts	Reclaiming disabled	1s lifespan	10s lifespan
No matching persistence	285708435	5874605	3564224
No records to associate	31051569	26227813	26226169
Records associated	2517852	614027	613197
Sessions created	124912	1127	89
Sessions reclaimed	0	123767	124805

After a retest of the same application and scenario with the default session lifespan, I compared the counters obtained from AbstractTempTable._hasRecords calls and separated them from other sources in the table below (also copied from #7167-12).

Counts	from AbstractTempTable	from BufferManager / other
Number of calls	11312	1188
No matching persistence	3534544	2457786
No records to associate	26237166	1974
Records asso	601546	1123

ciate d		

The purpose of this issue is to find a way to reduce the number of calls to BufferManager.activeBuffers that are resulted from AbstractTempTable._hasRecords.

#2 - 01/15/2024 04:25 AM - Dănuț Filimon

- Related to Bug #7167: Associating records from opened buffers to new sessions is slow added

#3 - 01/15/2024 08:04 AM - Dănuț Filimon

After a fix involving going over the buffers from AbstractTempTable.allBuffers and execute the same logic from activeBuffers, I noticed that whenever _hasRecords is called and the activeBuffers is reached, false is always returned. This means that the condition where the temp table is checked to be equal to the current object is never true. The logic was first introduced in trunk/rev.10452, since then it was improved by direct access.

#4 - 01/15/2024 08:38 AM - Alexandru Lungu

Danut, the code using allBuffers is meant to check records that were not yet flushed to the database (transient) and are stored only in FWD. Thus, we need to check all buffers to see if there is a record that was not flushed yet.

This can be improved in two ways:

- Check if we really need to check the transient records for some use-cases.
  - The code is called from TempTableBuilder.clear - is this actually considering transient records?
  - It is also called by AbstractTempTable.setCanUndo - is this a bottleneck (?) how often is _hasRecords hit from this spot?
- Iterate only the buffers over that table; not **all** open buffers.

#5 - 01/15/2024 09:03 AM - Dănuț Filimon

The POC application calls AbstractTempTable.hasRecords and clear/setCanUndo are not called at all. I noticed that this method can be directly called from the converted code (in this case, in a for each loop).

#6 - 01/15/2024 09:04 AM - Dănuț Filimon

- Assignee set to Dănuț Filimon

- Status changed from New to WIP

#7 - 01/15/2024 09:24 AM - Dănuț Filimon

Alexandru Lungu wrote:

> This can be improved in two ways:
>
> - Check if we really need to check the transient records for some use-cases.
>   - The code is called from TempTableBuilder.clear - is this actually considering transient records?
>   - It is also called by AbstractTempTable.setCanUndo - is this a bottleneck (?) how often is _hasRecords hit from this spot?

Both methods should check for transient records because they involve checking if there are any records into the table (and if any, the methods will throw an error).

#8 - 01/15/2024 09:28 AM - Alexandru Lungu

I wasn't aware this could be called directly from the converted code.

#9 - 01/22/2024 07:21 AM - Dănuț Filimon

Created 8196a.

#10 - 01/22/2024 08:50 AM - Dănuț Filimon

- Status changed from WIP to Review

Committed 8196a/rev.14936. Removed the call to activeBuffers from AbstractTempTable._hasRecords and integrated it's functionality into the method, now only the buffers of the current table will be iterated.

#11 - 01/30/2024 05:38 AM - Alexandru Lungu

I suspect the code duplicate to be quite bad architecturally.

Lets overload activeBuffers(Persistence, boolean) with activeBuffers(Persistence, boolean, Iterator<Buffer>). The first one will call by default the second one with **all** buffers. However, AbstractTempTable will call it only with the iterator of that table's buffers.

Please refactor.

#12 - 01/30/2024 06:26 AM - Dănuț Filimon

Note that activeBuffers iterates RecordBuffer while we want to iterate Buffer instances, so those need to be casted like so: RecordBuffer next = ((BufferReference) iter.next()).buffer();. This was the main reason I did not create another method, but we can pass Iterator<RecordBuffer> as an additional parameter and prepare the Iterator beforehand.

#13 - 01/30/2024 07:06 AM - Dănuț Filimon

- % Done changed from 0 to 100

Committed 8196a/rev.14937. I've created a separate method to check each buffer individually and it can be used directly by _hasRecords().

Alexandru, please review and let me know if I can go ahead with the POC performance tests.

#14 - 01/30/2024 08:19 AM - Alexandru Lungu

- Status changed from Review to Internal Test

- % Done changed from 100 to 0

The changes look good to me!
Danut, you can inline isActiveBuffer by doing return <large-if-condition> && buffer.getCurrentRecord() != null;. This can be done in AbstractTempTable as well if (...isActiveBuffer() && next.getParentTable().equals(this)) return true;
Please go ahead with the testing. This requires some massive testing as we are changing the way we analyze if a temp-table is empty or not:

- test POC for performance
- test large customer application regression tests
- test large customer application

#15 - 01/30/2024 08:31 AM - Dănuț Filimon

- % Done changed from 0 to 100

Alexandru Lungu wrote:

> The changes look good to me!
> Danut, you can inline isActiveBuffer by doing return <large-if-condition> && buffer.getCurrentRecord() != null;. This can be done in AbstractTempTable as well if (...isActiveBuffer() && next.getParentTable().equals(this)) return true;

I inlined the conditions in **8196a/rev.14938** and will now begin testing.

#16 - 01/31/2024 07:07 AM - Dănuț Filimon

POC regression tests passed and the results of the performance tests showed an improvement of **5.27%** for the average of the last 20 runs and **0.91%** for the total average.

I also did a retest on the scenario mentioned in #8196-1 and got the following results:

Counts	AbstractTempTable	BufferManager
Total calls	11312	747
Found active buffer	0	994
No matching persistence	0	1346652
No records to associate	59758	1401
Records associated	0	994

A few explanations of the counters used:

- Total calls: number of calls to AbstractTempTable._hasRecords() / BufferManager.activeBuffers().
- Found active buffer: isActiveBuffer() returns **true**.
- No matching persistence / No records to associate / Records associated: as before, but the counters are split based on which method is called.

#17 - 02/01/2024 03:29 AM - Alexandru Lungu

This is good news. The changes are quite safe from my POV.

This is ready for merge. Greg, this is a performance improvement we need before rebasing 7156b.

#18 - 02/01/2024 07:28 AM - Greg Shah

- Status changed from Internal Test to Merge Pending

8196a can merge after 8107a.

#19 - 02/01/2024 07:46 AM - Dănuț Filimon

- Status changed from Merge Pending to Test

Branch 8196a was merged into trunk as rev.14961 and archived.