Database - Bug #5306

AdaptiveQuery sets a found record into the buffer twice in dynamic mode

05/03/2021 03:44 PM - Eric Faulhaber

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

Description

Related issues:

Related to Database - Feature #5355: _Usertablestat metadata improvements	**WIP**

History

#1 - 05/03/2021 03:53 PM - Eric Faulhaber

When AdaptiveQuery finds a record in dynamic mode, it sets it into RecordBuffer via setRecord twice. This was discovered while working on #3814. From that task:

For example for GET-FIRST it is first called at:

```
Daemon Thread [Conversation [0000000F:bogus]] (Suspended (breakpoint at line 191 in UserTableStatUpdater))
    UserTableStatUpdater.retrieve(Database, RecordBuffer) line: 191
    RecordBuffer.setRecord(Record, LockType, boolean, SortIndex, OffEnd, boolean, boolean, boolean) line: 9325

    RandomAccessQuery.updateBuffer(Record, LockType, boolean, OffEnd) line: 3252
    RandomAccessQuery.next(Object[], LockType) line: 1849
    RandomAccessQuery.next() line: 1717
    AdaptiveQuery$DynamicResults.next() line: 4239
    ResultsAdapter.next() line: 161
    AdaptiveQuery.next(LockType) line: 2136
    AdaptiveQuery.first(LockType) line: 1680
    AdaptiveQuery(PreselectQuery).first() line: 2395
    AdaptiveQuery(AbstractQuery).getFirst() line: 2362
    QueryWrapper.lambda$getFirst$4() line: 4889
    167937990.get() line: not available
    QueryWrapper.handleQueryOffEnd(Supplier<logical>) line: 6577
    QueryWrapper.getFirst() line: 4889
    Stat1.lambda$execute$14() line: 231
```

and then at

```
Daemon Thread [Conversation [0000000F:bogus]] (Suspended (breakpoint at line 191 in UserTableStatUpdater))
    UserTableStatUpdater.retrieve(Database, RecordBuffer) line: 191
    RecordBuffer.setRecord(Record, LockType, boolean, SortIndex, OffEnd, boolean, boolean, boolean) line: 9325

    AdaptiveQuery(PreselectQuery).coreFetch(Object[], LockType, boolean, boolean) line: 5866
    AdaptiveQuery(PreselectQuery).fetch(boolean, LockType, boolean) line: 5631
    AdaptiveQuery.fetch(boolean, LockType, boolean) line: 3762
    AdaptiveQuery.next(LockType) line: 2158
    AdaptiveQuery.first(LockType) line: 1680
    AdaptiveQuery(PreselectQuery).first() line: 2395
    AdaptiveQuery(AbstractQuery).getFirst() line: 2362
    QueryWrapper.lambda$getFirst$4() line: 4889
    167937990.get() line: not available
    QueryWrapper.handleQueryOffEnd(Supplier<logical>) line: 6577
```

```
    QueryWrapper.getFirst() line: 4889
    Stat1.lambda$execute$14() line: 231
```

This presents two problems:

- more work is being done that is necessary, so this represents a potential optimization;
- a record retrieval counter for the meta_usertablestat VST implementation (the counter is implemented within RecordBuffer.setRecord; see #3814 for details) is being invoked twice for each record when we follow this code path, so this count is wrong.

#2 - 05/10/2021 01:31 PM - Eric Faulhaber

- Related to Feature #5355: _Usertablestat metadata improvements added

#3 - 05/11/2021 03:10 PM - Ovidiu Maxiniuc

I have investigated this issue but I cannot reproduce it. I took the isolated testcase from #3814-406, converted and executed it with 3821c/12403. I put a breakpoint in UserTableStatUpdater.retrieve(). It hit only once for me. The stack trace is this:

```
"Conversation [00000001:bogus]@5584" daemon prio=5 tid=0x2a nid=NA runnable
  java.lang.Thread.State: RUNNABLE
      at com.goldencode.p2j.persist.meta.UserTableStatUpdater.retrieve(UserTableStatUpdater.java:206)
      at com.goldencode.p2j.persist.RecordBuffer.setRecord(RecordBuffer.java:9342)
      at com.goldencode.p2j.persist.PreselectQuery.coreFetch(PreselectQuery.java:5863)
      at com.goldencode.p2j.persist.PreselectQuery.fetch(PreselectQuery.java:5632)
      at com.goldencode.p2j.persist.AdaptiveQuery.fetch(AdaptiveQuery.java:3758)
      at com.goldencode.p2j.persist.PreselectQuery.first(PreselectQuery.java:2434)
      at com.goldencode.p2j.persist.AdaptiveQuery.first(AdaptiveQuery.java:1706)
      at com.goldencode.p2j.persist.PreselectQuery.first(PreselectQuery.java:2397)
      at com.goldencode.p2j.persist.AbstractQuery.getFirst(AbstractQuery.java:2362)
      at com.goldencode.p2j.persist.QueryWrapper.lambda$4(QueryWrapper.java:5014)
      at com.goldencode.p2j.persist.QueryWrapper$$Lambda$466.1544689002.get(Unknown Source:-1)
      at com.goldencode.p2j.persist.QueryWrapper.handleQueryOffEnd(QueryWrapper.java:6742)
      at com.goldencode.p2j.persist.QueryWrapper.getFirst(QueryWrapper.java:5014)
      at com.goldencode.testcases.RecTwiceSet.lambda$0(RecTwiceSet.java:35)
      [...]
```

This is basically the second stack from #3814-399 but there are some differences in line numbers. For some reason, all files are affected.
My conclusion is that the switch to dynamic mode of the adaptive query which causes the record to be set twice into the buffer is a regression from the changed code.

#4 - 10/26/2022 06:28 AM - Alexandru Lungu

The same issue I encountered when implemented #6582. When AdaptiveQuery was in dynamic mode, the record was fetched twice: by the underlying query used for dynamic query (RandomAccessQuery or CompoundQuery) and by the AdaptiveQuery itself.

When I used CompoundQuery as part of multi-table AdaptiveQuery implementation, the record was also fetched twice. This is not just a performance issue in my case, but also breaks the logic. The buffer is snapshot in order to keep track of a cursor reference-row:

- when using next of the compound query, the field references are resolved in regard to the snapshots, not the actual buffer records. In WHERE tt.f1 = tt2.f2, tt.f1 is from a snapshot. This means that changing tt in the meantime won't reflect in the evaluation of the WHERE clause, until we move to another tt.
- this is done in order to mimic 4GL behavior. There are also other particularities when considering SCROLLING, FIRST, LAST etc.
- when the second fetching is done (by AdaptiveQuery), the snapshots are invalidated, so the query works incorrectly. Before WHERE tt.f1 = tt2.f2, tt is re-fetched, so tt.f1 is no longer a snapshot.
- the performance issue is not negligible for multi-table AdaptiveQuery, as we don't want to re-fetch all the joined records, which most probable didn't changed anyways. As an example: in FOR EACH tt, EACH tt2, EACH tt3, we may fetch tt, tt2 and tt3 twice (so 6 fetches in total per next). tt and tt2 most probably are still identical, so 5 out of 6 fetches are on average not needed.

I fixed this with 6582a, by adding Results.isAutoFetch() which is by default false. AdaptiveQuery$DynamicResults.isAutoFetch is true, as the delegate query is fetching the records into buffers "by itself".

#5 - 03/13/2023 10:32 AM - Alexandru Lungu

- Assignee set to Alexandru Lungu

- Status changed from New to WIP

#6 - 03/15/2023 07:00 AM - Alexandru Lungu

- File auto_fetch_results.patch added

- % Done changed from 0 to 100

- Status changed from WIP to Review

Committed fix for repetitive fetching in 7026b/rev. 14505. I also attached the changes here.

The fix is the same as the one I previously implemented for #6582 (which didn't reached trunk yet): a isAutoFetch method of results to state if they also fetch the DMO into the buffer. This way, the AdaptiveQuery is not required to fetch it again in the buffer.

Tested with 2 large applications, there is no regression. I will do some tests with adaptive_scrolling test suite, which is really good for testing adaptive query invalidation (scrolling and non-scrolling).

#5355 is dependent upon this fix. Please consider it cleared after reviewing/testing this task.

#7 - 03/16/2023 10:58 AM - Alexandru Lungu

Radu, I think you have both 7026b (at least rev. 14505) and the adaptive_scrolling tests (scrolling and non-scrolling) already set up from your recent tests with #6444. It is not relevant if it is run with temporary or persistent database - the change targets the AdaptiveQuery logic alone.

Can you do a run on the latest rev. 14507?

#8 - 03/17/2023 05:07 AM - Radu Apetrii

Alexandru Lungu wrote:

> Can you do a run on the latest rev. 14507?

Since the tests were already set up, I ran through all of them. There were no errors found and from the little investigation I've done, there were no records fetched twice.
I tested with 7026b, rev. 14508.

#9 - 03/17/2023 05:51 AM - Alexandru Lungu

Radu Apetrii wrote:

> Alexandru Lungu wrote:
>
> > Can you do a run on the latest rev. 14507?
>
> Since the tests were already set up, I ran through all of them. There were no errors found and from the little investigation I've done, there were no records fetched twice.
> I tested with 7026b, rev. 14508.

Thank you!
7026b/rev. 14505 is ready for review.

#10 - 04/20/2023 07:13 AM - Alexandru Lungu

- Status changed from Review to Test

This was merged in trunk as rev. 14523 and can be closed.

Files

auto_fetch_results.patch	5.22 KB	03/15/2023	Alexandru Lungu