Database - Bug #7420

Optimize Cursor for scrolling AdaptiveQuery

06/08/2023 08:13 AM - Alexandru Lungu

Status:	Test	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Dănuț Filimon	**% Done:**	20%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 06/08/2023 08:40 AM - Alexandru Lungu

- Status changed from New to WIP

- Assignee set to Alexandru Lungu

This is a task in regard to a hot-spot in AdaptiveQuery.setScrolling(). It is called 7506, but it takes 795ms own time.
This is mostly because setScrolling instantiates a Cursor (which also creates a results List, a RepositionCache with a Node root that has a LinkedHashMap children field). Therefore, there are 5 objects created on AdaptiveQuery.setScrolling().

First step here is to make LinkedHashMap lazily instantiated, as most nodes (AFAIK) are leaves - so this map will be mostly empty.
For single-table queries, the root is the only one that actually needs a LinkedHashMap children field.
For multi-table queries, nodes that aren't for the outermost table should have LinkedHashMap. Even so, there are more nodes that don't need an instantiated children than nodes that have children.

Committed this initial optimization in 7026e/rev. 14597. However, I think we can do better with Cursor, considering that all dynamic queries are scrolling by default.

#2 - 06/14/2023 03:02 AM - Alexandru Lungu

- % Done changed from 0 to 20

#3 - 06/15/2023 07:57 AM - Alexandru Lungu

- Assignee changed from Alexandru Lungu to Dănuț Filimon

Danut, can you take a look into this one?

It is quite bad that AdaptiveQuery.setScrolling() works that slow. Please test a customer application and check if you meet the same bottleneck. Try to navigate the customer application / a custom test-case with scrolling queries till you have close to ~7k setScrolling called. Let me know if you also reach ~800ms of time spent on aggregate there (in setScrolling and forward calls).

Please try without the changes in 7026e first - maybe use trunk.

#4 - 06/15/2023 09:52 AM - Dănuț Filimon

I tested a customer application using AdaptiveQueryTraceAspect. My results for setScrolling were ~5.5k calls with ~36 ms execution time. In my opinion, I think there is something wrong here and I will need to retest it using MethodTraceAspect.

#5 - 06/15/2023 10:44 AM - Alexandru Lungu

I wonder if our LTW is spoiled somehow by the GC. I mean, for a simple setScrolling, it is hard to believe that 7506 calls takes 795ms agg time. Maybe one of the setScrolling calls caught the GC in action. If you can't replicate, I guess we can reject this. I will redo my tests and keep you updated if the time changes.

#6 - 06/19/2023 05:04 AM - Dănuț Filimon

I retested the customer application using MethodTraceAspect instead of AdaptiveQueryTraceAspect and **8031** calls took ~**29 ms**.

Alexandru Lungu wrote:

> I wonder if our LTW is spoiled somehow by the GC. I mean, for a simple setScrolling, it is hard to believe that 7506 calls takes 795ms agg time. Maybe one of the setScrolling calls caught the GC in action. If you can't replicate, I guess we can reject this. I will redo my tests and keep you updated if the time changes.

From what you mentioned, this is probably a rare case and since I couldn't replicate the scenario, I'll wait for the test results from you.

#7 - 07/13/2023 09:02 AM - Alexandru Lungu

- Status changed from WIP to Review

7026e reached into trunk. This can be closed.
I couldn't replicate my times from #7420. I guess the tracing was spoiled by something that was really time consuming (a GC action?)

#8 - 11/01/2023 11:29 AM - Alexandru Lungu

- Status changed from Review to Test