Database - Bug #4703

investigate whether performance of TempTableDataSourceProvider can be improved

06/30/2020 05:50 AM - Eric Faulhaber

Status:	Hold	**Start date:**	
Priority:	High	**Due date:**	
Assignee:	Igor Skornyakov	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to Database - Support #4702: write temp-table performance test cases	**Closed**
Related to Database - Feature #4011: database/persistence layer performance i...	**Closed**

History

#1 - 06/30/2020 05:51 AM - Eric Faulhaber

- Related to Support #4702: write temp-table performance test cases added

#2 - 06/30/2020 05:52 AM - Eric Faulhaber

- Related to Feature #4011: database/persistence layer performance improvements added

#3 - 06/30/2020 05:53 AM - Igor Skornyakov

Is it a top-priority task?
Thank you.

#4 - 06/30/2020 06:00 AM - Eric Faulhaber

TempTableDataSourceProvider was introduced in branch 4011a to provide database connections to the in-memory H2 database used for temp-table support. It currently uses DriverManager.getConnection to establish a new connection. This is fine for very long-running temp-table database sessions. However, I suspect we can get into some use cases where we are "thrashing" the creation and closing of database sessions, along with their backing JDBC connections.

In #4702, I have requested some test cases be written that expose this "thrashing" use case, so we can see whether this is a bottleneck. Perhaps it is best that the assignee of this issue write those test cases and provide them back for use with #4702.

If this proves to be a bottleneck, we would want to back TempTableDataSourceProvider with a c3p0 pool instead of using DriverManager.getConnection.

#5 - 06/30/2020 06:02 AM - Eric Faulhaber

- Priority changed from Normal to High

Igor Skornyakov wrote:

> Is it a top-priority task?

It is next highest priority behind any work needed to complete the rebase of the 4011a branch.

#6 - 07/02/2020 01:39 PM - Eric Faulhaber

FYI, 4011b is the branch for this work. 4011a and 4011a_rebase are now frozen.

#7 - 07/03/2020 02:00 PM - Igor Skornyakov

I've tested calling one of Stanislav's tests (with minor changes to suppress output) as an external program in loop:

```
DO i = 1 TO 1000000:
    RUN h2_performance/perf-create1.p.
END.
```

With FWD it runs several orders of magnitude faster than with 4GL but the connections are created only at the server startup. Which modifications are required to test "connection trashing"?
Thank you.

#8 - 07/06/2020 05:51 AM - Eric Faulhaber

This is good news, but it suggests that the test did not recreate the condition I thought it would, either because I am misunderstanding the way the data source provider is implemented, or because I misjudged the conditions needed to allow a temp-table related Session object to be closed.

I'm not sure what you mean by the connections are created at server startup. My understanding is that the first time TempTableDataSourceProvider needs to create a connection, it does so, and this connection is stored in a context-local instance of the inner Context class. When the Session for that connection is closed, the connection is released if that context's count of active temp-tables is 0. A new connection would be retrieved the next time a Session object is opened for that context.

I thought I had witnessed situations (like the looping test case above) where we would have short-lived temp-tables, and thus the associated session and connection would be closed and opened again in quick succession. It seems that scenario was not recreated by that test.

#9 - 07/06/2020 06:03 AM - Igor Skornyakov

Eric Faulhaber wrote:

> This is good news, but it suggests that the test did not recreate the condition I thought it would, either because I am misunderstanding the way the data source provider is implemented, or because I misjudged the conditions needed to allow a temp-table related Session object to be closed.
>
> I'm not sure what you mean by the connections are created at server startup. My understanding is that the first time TempTableDataSourceProvider needs to create a connection, it does so, and this connection is stored in a context-local instance of the inner Context class. When the Session for that connection is closed, the connection is released if that context's count of active temp-tables is 0. A new connection would be retrieved the next time a Session object is opened for that context.

Well, maybe I have missed something, but the debugger stops at the corresponding call only at the server startup.

> I thought I had witnessed situations (like the looping test case above) where we would have short-lived temp-tables, and thus the associated session and connection would be closed and opened again in quick succession. It seems that scenario was not recreated by that test.

Indeed. I was trying to figure out in which situations the converted code will close the connection, but w/o success so far. Maybe it makes sense to use pure Java test? Unfortunately, the way the 1TempTableDataSourceProvider is implemented makes this a little bit tricky.

#10 - 07/06/2020 06:16 AM - Eric Faulhaber

I don't think it is really worth it to contrive something in hand-written Java to recreate a problem that we haven't actually seen coming from converted code (at least not that I can recall or pinpoint now). I opened this task based on a perceived problem from my reading of the source for the data source provider. I recalled seeing behavior which I attributed to this potential problem, but through assumption, not debugging. Until we hit something that actually realizes this potential in real converted code, let's put this task on hold.

#11 - 07/06/2020 06:44 AM - Igor Skornyakov

Eric Faulhaber wrote:

> I don't think it is really worth it to contrive something in hand-written Java to recreate a problem that we haven't actually seen coming from converted code (at least not that I can recall or pinpoint now). I opened this task based on a perceived problem from my reading of the source for the data source provider. I recalled seeing behavior which I attributed to this potential problem, but through assumption, not debugging. Until we hit something that actually realizes this potential in real converted code, let's put this task on hold.

OK, thank you. Please note that I have not seen a single call to the TempTableDataSourceProvider during profiling the server with YourKit during the client session run.

#12 - 07/07/2020 03:31 PM - Eric Faulhaber

- Status changed from New to Hold