Database - Bug #7496

finish support for query:forward-only attribute

07/10/2023 06:12 PM - Eric Faulhaber

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Dănuț Filimon	**% Done:**	70%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

Related issues:

Related to Database - Bug #7737: QueryWrapper should either show or throw err...	**WIP**
Related to Database - Bug #7991: use ScrollingResults instead of ProgressiveR...	**WIP**

History

#1 - 07/10/2023 06:47 PM - Eric Faulhaber

- Assignee set to Alexandru Lungu

Legacy support for this attribute was added long ago, but it seems FWD does nothing internally with the information, other than to report it when interrogated, and to use its state for some legacy error processing in QueryWrapper.

As Constantin pointed out in a different task, FWD can use this information to set ResultSet.TYPE_FORWARD_ONLY when executing a SQL query. This will allow some (e.g., PostgreSQL and hopefully other) database dialects to honor JDBC fetch size in conjunction with a database server-side cursor to optimize memory use and performance of these queries.

In addition, the preselect mode of AdaptiveQuery should use ResultSet.TYPE_FORWARD_ONLY with ProgressiveResults, when that AdaptiveQuery represents a converted FOR EACH statement (i.e., not the 4GL query form). This is not a use of the attribute per se, because the attribute is not usable with the FOR EACH statement. However, I mention it because AdaptiveQuery backs both the FOR EACH statement form, as well as the static and handle-based query form of FOR EACH, and we will need to differentiate the two to apply ResultSet.TYPE_FORWARD_ONLY correctly, depending on the form of use and (in the query case), the state of the 4GL FORWARD-ONLY attribute.

The current support for this attribute should be refactored to account for this duality of use and to use ResultSet.TYPE_FORWARD_ONLY with the backing JDBC queries whenever possible/appropriate, without breaking the legacy attribute's semantics.

Alexandru, please re-assign this task as you see fit.

#3 - 07/11/2023 03:13 AM - Alexandru Lungu

- Status changed from New to WIP

- Assignee changed from Alexandru Lungu to Dănuț Filimon

Danut, please focus on this issue. Check if we already have some tests for query:forward-only. Otherwise, build a full suite and check what errors are thrown in different scenarios. Test with first, prev, next, last, reposition (scrolling and non-scrolling). After we have a complete overview of this, we can implement it to fully replicate the 4GL attribute.

#4 - 07/11/2023 05:22 AM - Dănuț Filimon

I did not found any tests that use forward-only so I created a test suite with 12 examples. From the 12 tests mentioned, 2 were missing an error but had correct results and 2 displayed the wrong error and wrong results. I am currently investigating the test scenario where doing GET-NEXT and REPOSITION-BACKWARD(1) causes the record to be unavailable, but it should be available since reposition should not happen on a non-scrolling query.

#5 - 07/11/2023 09:42 AM - Dănuț Filimon

Committed 7496a/rev.14650. Reposition can be done only for non-scrolling queries. From the test suite I noticed that by setting the forward-only attribute to true a scrolling query will become a non-scrolling query and only non-scrolling queries can use reposition.

There are still a few changes that need to be done:

- Dynamic queries are scrolling by default, if the forward-only flag is set to false, the scrolling value should not be modified at all (**omitted in the commit**).
- Write test cases for reposition by rowid.
- The documentation says that the forward-only attribute can't be set while a query is open or is being browsed. In this case we can use a query and close it, then set the attribute before opening and using the query again. We might want to keep track if the query can be scrolling from the start so that we can have it as such when possible.

#6 - 07/11/2023 09:53 AM - Alexandru Lungu

> The documentation says that the forward-only attribute can't be set while a query is open or is being browsed. In this case we can use a query and close it, then set the attribute before opening and using the query again. We might want to keep track if the query can be scrolling from the start so that we can have it as such when possible.

Danut, please double check 4GL if it actually remembers that the query was scrolling in the first place. Therefore, check if after close / set forward-only on false / open, 4GL sets the query back on scrolling.

#7 - 07/11/2023 09:59 AM - Dănuț Filimon

Alexandru Lungu wrote:

> Danut, please double check 4GL if it actually remembers that the query was scrolling in the first place. Therefore, check if after close / set forward-only on false / open, 4GL sets the query back on scrolling.

It does set it back to scrolling.

#8 - 07/12/2023 02:55 AM - Alexandru Lungu

Ok! Danut please take the extra mile to ensure that ResultSet.FORWARD_ONLY is used if query:forward-only is set. Therefore, please check **all** kind of results we have back there and make sure they are honoring ResultSet.FORWARD_ONLY.
Coordinate with the effort in #7241-113 (onward); Stefan is attempting to refactor ProgressiveResults in order to fully support ResultSet.FORWARD_ONLY. Danut, your goal is just to bring the forwardOnly flag till the results class, so it can be picked up there. Currently, Stefan is hard-coding ResultSet.FORWARD_ONLY inside ProgressiveResults to test it.

#9 - 07/12/2023 07:51 AM - Dănuț Filimon

Alexandru Lungu wrote:

> Ok! Danut please take the extra mile to ensure that ResultSet.FORWARD_ONLY is used if query:forward-only is set. Therefore, please check **all** kind of results we have back there and make sure they are honoring ResultSet.FORWARD_ONLY.
> Coordinate with the effort in #7241-113 (onward); Stefan is attempting to refactor ProgressiveResults in order to fully support ResultSet.FORWARD_ONLY. Danut, your goal is just to bring the forwardOnly flag till the results class, so it can be picked up there. Currently, Stefan is hard-coding ResultSet.FORWARD_ONLY inside ProgressiveResults to test it.

Committed 7496a/rev.14651. Improved the support for the FORWARD-ONLY attribute. I managed to make it available in PreselectQuery and added an additional condition in AdaptiveQuery.executeQuery which will eventually create ProgressiveResults instead of ScrollingResults.

The only thing to do that is left is to throw an error when using BROWSE with a forward-only query and finish up the test suite. Another important thing to note a query that includes a BREAK BY phrase becomes a FORWARD-ONLY query (as specified in the documentation), but support for this is not necessary since using BREAK BY will generate a PresortQuery with ResultSet.TYPE_FORWARD_ONLY by default.

#10 - 07/13/2023 07:49 AM - Dănuț Filimon

I did a few tests with browse and found a few problems:

- Between 7421a/rev.14649 and 7421a/rev.14651, the following error occurs when defining a query for a browse with FORWARD-ONLY attribute set to true: BROWSE requires the QUERY to be DEFINEd SCROLLING. (3330) which is correct since a scrolling query will become non-scrolling if this flag is set to true.
- In 4GL, setting the FORWARD-ONLY attribute to either true or false will result in this error being thrown: Forward only status may not be changed while query q is open or being browsed. (12354) followed by Unable to set attribute FORWARD-ONLY in widget of type QUERY. (3131).. When eliminating the DEFINE BROWSE which uses the query, these errors do not appear. In FWD, the previous error is replaced by error 3330, but only when FORWARD-ONLY is true.

Apart from the points mentioned above, I don't think there is a huge problem. The only way for error 12354 and 3131 to be thrown in FWD is by setting the forward-only attribute when the query is open. I see that there is a QueryWrapper.browsed flag available that can be used for this case.

In FWD, FORWARD-ONLY is by default false, which makes it possible to create a query which will set the value of the attribute to false and still be able to create a BROWSE with that query. This happens because the QueryWrapper.browsed value is set later, when the query is attached to the browse, after it is opened (please correct me if I am wrong).

While testing, I stumbled upon an issue related to GET PREV and GET LAST where after creating a few records and opening a query with forward-only enabled, will cause the GET LAST to throw an error (correct) and if you try to message data from the buffer it will manage to print it. This does not happen in 4GL, once the error is thrown, no records are found. I will take a look and create a separate issue for this after checking if it was not already reported.

Committed 7496a.rev/14652. Since the forwardOnly flag might be useful in the future in other queries, I added it as a member to AbstractQuery and added the required methods to use it. I also added the missing BROWSE errors mentioned above.

Now I will test the changes on a customer application and add my test suite to testcases. I will follow up with an update after I am done.

#11 - 07/13/2023 09:11 AM - Dănuț Filimon

I didn't find any problem while testing a customer application.

Committed testcases/rev.2380. Added FORWARD-ONLY tests.

#12 - 07/17/2023 04:33 AM - Alexandru Lungu

- % Done changed from 0 to 100

- Status changed from WIP to Review

Review of 7496a

- We still need support for forwardOnly in lazy AdaptiveQuery. Therefore, it is fine to execute the lazy queries over H2 in forwardOnly manner - no need to rule it out.
- PreselectQuery still has forwardOnly = false; data member, is this still needed? If you remove it, remove all your history entries from that file as there won't be any changes there at the end.
- use Override annotation for @setForwardOnly and isForwardOnly wherever implemented

I am OK with the changes by now. Starting a testing round and performance check. If everything goes right, we can merge it as it is now.
With Stefan's changes from #7241, we will propagate this flag to the lower part of the persistence layer (Results).

#13 - 07/17/2023 04:58 AM - Dănuț Filimon

Committed 7496a/rev.14653. Quick changes made based on the #7496-12 review. Similar to PreselectQuery, I removed the changes and logs from AdaptiveQuery.

#14 - 07/17/2023 05:44 AM - Alexandru Lungu

- Status changed from Review to WIP

- % Done changed from 100 to 80

Danut, I think the following code is quite interesting:

```
      boolean forwardOnlyValue = forwardOnly.booleanValue();
      if (forwardOnlyValue)
      {
         this.scrolling = false;
      }
      else
      {
         this.scrolling = originalScrolling;
      }
      setForwardOnly(forwardOnlyValue);
```

I guess it is ok to change the scrolling flag when forwardOnly is changed, but note that the delegate won't be affected. Therefore, this.scrolling = false; won't affect the delegate, but it won't ever could (as scrolling can be set only on true, considering that false is the default - and this can't be reverted).

The same goes for this.scrolling = originalScrolling;, but with extra steps.

I think that we can't rely on QueryWrapper to properly do the propagation in the delegates. I mean it is fine that it does this whole state transitions from scrolling to non-scrolling / forward-only to non-forward-only and vice-versa. However, we need to replicate this behavior in the underlying queries. Consider changing PreselectQuery.isScrolling implementation (and other isScrolling implementations) to (scrolling && !typeForwardOnly) || isNativelyScrolling(). This way, we propagate TYPE_FORWARD_ONLY implicitly to the Results, so this is fine.

#15 - 07/17/2023 10:07 AM - Dănuț Filimon

Alexandru Lungu wrote:

> I think that we can't rely on QueryWrapper to properly do the propagation in the delegates. I mean it is fine that it does this whole state transitions from scrolling to non-scrolling / forward-only to non-forward-only and vice-versa. However, we need to replicate this behavior in the underlying queries. Consider changing PreselectQuery.isScrolling implementation (and other isScrolling implementations) to (scrolling && !typeForwardOnly) || isNativelyScrolling(). This way, we propagate TYPE_FORWARD_ONLY implicitly to the Results, so this is fine.

I made a few changes to allow the transition between scrolling states in **7496a**/**rev.14654**. I also tested 1/4 of my test suite independently and a customer application. Everything works fine and the results of the test suite were all correct, with the exception of one which turned from a **No record available** error to the record hold by the buffer after creating all the records (which is already an existent problem) and I still need to look into.

#16 - 07/18/2023 03:30 AM - Alexandru Lungu

Review of 7496a

- I don't think setScrolling should be called when calling setForwardOnly. This will make all queries that are forward-only scrolling - this is not good.
- When setting isForwardOnly in AbstractQuery (or overrides):
  - on true: if the query is scrolling, the cursor should be dropped.
  - on false: if the query is scrolling, the cursor should be created (but only if the query was "originally" scrolling.

#17 - 07/19/2023 07:08 AM - Alexandru Lungu

Danut, please check what happens if you actually do first, prev or last when type-forward only is set (in 4GL and FWD). I want just to make sure we do it right even for these "non-reposition" cases. I see that first is simply calling delegate's first, which calls results' first (which may be incorrect due to forward only flag. Mind that Stefan has an implementation of results that logs if such attempts are made, but we should ensure they are captured earlier to throw proper 4GL exceptions.

Just as a fact: in ForwardResults (from #7241), first and last are supported as long as first can be done in special conditions (we are still before first) and last can be always done.

#18 - 07/19/2023 09:48 AM - Dănuț Filimon

Alexandru Lungu wrote:

> Danut, please check what happens if you actually do first, prev or last when type-forward only is set (in 4GL and FWD). I want just to make sure we do it right even for these "non-reposition" cases. I see that first is simply calling delegate's first, which calls results' first (which may be incorrect due to forward only flag. Mind that Stefan has an implementation of results that logs if such attempts are made, but we should ensure they are captured earlier to throw proper 4GL exceptions.

I tested the following cases with FORWARD-ONLY=TRUE and both 4GL and FWD showed the same results:
- **GET-FIRST**: retrieves the first record correctly;
- **GET-NEXT > GET-FIRST**: Shows error 12394;
- **GET-LAST**: Shows error 12393;
- **GET-NEXT > GET-LAST**: Shows error 12393;
- **GET-PREV**: Shows error 12393;
- **GET-NEXT > GET-PREV**: Shows error 12393;

With FORWARD-ONLY=FALSE, 4GL and FWD again showed the same results:
- **GET-FIRST**: retrieves the first record;
- **GET-NEXT > GET-FIRST**: retrieves the first record;
- **GET-LAST**: retrieves the last record;
- **GET-NEXT > GET-LAST**: retrieves the last record;
- **GET-PREV**: retrieves the previous record (the last record created, buffer was released);
- **GET-NEXT > GET-PREV**: no record available after GET-NEXT;

I am currently working on #7496-16, originalScrolling is only available in QueryWrapper, but it should be made possible for other queries to make use of it through AbstractQuery. Some changes in 7496a/rev.14654 will be reverted, especially those that modified the functionality of the setScrolling method.

#19 - 07/19/2023 10:46 AM - Alexandru Lungu

That is great - please ensure that #7496-18 still holds after refactoring. Also, do some skim debugging on more complex cases:

- scrolling -> FORWARD-ONLY true -> FORWARD-ONLY false -> open (should be scrolling) -> close -> FORWARD-ONLY true -> open (should not be scrolling)
- scrolling -> FORWARD-ONLY true -> FORWARD-ONLY false -> open (should be scrolling) -> close -> FORWARD-ONLY false -> open (should be scrolling)
- scrolling -> FORWARD-ONLY true -> FORWARD-ONLY true -> open (should not be scrolling) -> close -> FORWARD-ONLY true -> open (should not be scrolling)
- scrolling -> FORWARD-ONLY true -> FORWARD-ONLY true -> open (should not be scrolling) -> close -> FORWARD-ONLY false -> open (should be scrolling)
- scrolling -> FORWARD-ONLY false -> FORWARD-ONLY false -> open (should be scrolling) -> close -> FORWARD-ONLY true -> open (should not be scrolling)
- scrolling -> FORWARD-ONLY false -> FORWARD-ONLY false -> open (should be scrolling) -> close -> FORWARD-ONLY false -> open (should be scrolling)
- scrolling -> FORWARD-ONLY false -> FORWARD-ONLY true -> open (should not be scrolling) -> close -> FORWARD-ONLY true -> open

(should not be scrolling)
- scrolling -> FORWARD-ONLY false -> FORWARD-ONLY true -> open (should not be scrolling) -> close -> FORWARD-ONLY false -> open (should be scrolling)

By "should be scrolling" I mean that isScrolling should return true. scrolling variable can be true even if isScrolling return false (as it represents the "original status of the query). Also, cursor != null should always hold when the query is truly scrolling (isScrolling is true).

#20 - 07/20/2023 10:49 AM - Alexandru Lungu

Danut, please do your best to get this done asap. I intent to have it merged intro trunk by the end of this week, so we can open new tasks of adapting this new mode to new use-cases: FOR EACH loops, honoring in different kind of results, etc.

#21 - 07/21/2023 07:24 AM - Dănuț Filimon

Alexandru Lungu wrote:

> Danut, please do your best to get this done asap. I intent to have it merged intro trunk by the end of this week, so we can open new tasks of adapting this new mode to new use-cases: FOR EACH loops, honoring in different kind of results, etc.

Committed 7496a/rev.14655. I've previously mentioned that other queries can make use of originalScrolling value through AbstractQuery but I made a mistake, it isn't necessary to use it's value at all and it should not even exist! Instead of doing a complicated process of toggling scrolling when setting FORWARD-ONLY to true/false, we can simply use an additional condition when calling isScrolling() (which was added previously and completed in 7496a/rev.14654).

The changes resume to the following:

- removed QueryWrapper.originalScrolling
- Always delegate forwardOnly value when opening the query
- removed the delegation process for scrolling when setting FORWARD-ONLY
- Reverted all the changes to PrelectQuery in 7496a/rev.14654, including history entries.
- Added setForwardOnly() to DynamicQuery and AdaptiveQuery in order to reset the cursor or disable it if necessary

All the tests mentioned in #7496-19 pass with these changes. I still need to test a customer application right now and check if everything works properly with the previous test suite.

Please review.

#22 - 08/07/2023 10:01 AM - Dănuț Filimon

I previously tested a large customer application using the changes from 7241b/rev.14655, 7496a/rev.14655 and a few changes to make the actual call that generates ForwardResults. A total of **2686** ForwardResults instances were created and the application ran without any problems. I've

experimented with the example provided in #7241-133 and created smaller tests. SessionListener class mentioned in #7241-138 is exactly what was missing from the implementation, SessionListener.Event.TRANSACTION_COMMITTING exists and it is handled but this event is **never notified**. By notifying this event in Persistence.Context.commit(), I got better results between FWD and 4GL (previously 2/4 showed **No ... record is available**, now all results match).

```
=== modified file 'src/com/goldencode/p2j/persist/Persistence.java'
--- old/src/com/goldencode/p2j/persist/Persistence.java    2023-05-23 11:54:21 +0000
+++ new/src/com/goldencode/p2j/persist/Persistence.java    2023-08-07 13:55:29 +0000
@@ -4778,6 +4778,11 @@

          try
          {
+            if (!sessionListeners.isEmpty())
+            {
+               notifySessionEvent(SessionListener.Event.TRANSACTION_COMMITTING);
+            }
+
             session.commit();

             if (LOG.isLoggable(Level.FINER))
```

I will retest the customer application with a few changes to generate more ForwardResults instances. I did not check how many No ... record is available errors might have been caused by ForwardResults since I only looked for SEVERE log types related to it.

#23 - 08/08/2023 03:27 AM - Alexandru Lungu

Danut, please find a concrete example where 7496a (together with 7241b) is failing before SessionListener.Event.TRANSACTION_COMMITTING and is fixed now. My point is that the PG server-cursor should have thrown something like "Portal ... does not exist". After the changes, it shouldn't be thrown. Also, please double check that the "implicit transaction" is created properly and commit is iterating all results till the end to fetch the remaining records and avoid PG server-cursor invalidation.

I agree that No ... record is available may have been caused by "Portal ... does not exist" in the first place. If this is the case, ignore my first comment; just double-check.

As for PreselectQuery, please use your type-forward-only tests to see if PRESELECT influences the forward-only and scrolling is any way, especially according to #7496-19. My final concern is if that FOR EACH is different from PRESELECT in regard to SCROLLING and FORWARD-ONLY behavior. If they are the same, I think we can remove isNativelyScrolling. Is a PRESELECT query that is defined as non-scrolling made automatically SCROLLING, just because it is preselect?

All queries should emit TYPE_SCROLL_INSENSITIVE (no matter if in 4GL we have SCROLLING or not). Only queries that are explicitly set with FORWARD-ONLY should emit TYPE_FORWARD_ONLY and determine PG server-cursor. Also, queries that are TYPE_FORWARD_ONLY should always use ForwardResults, while TYPE_SCROLL_INSENSITIVE should use ScrollingResults. Both wrap ScrollableResults and can be wrapped by a ProgressiveResults.

Danut, please port the changes from 7241b into 7496a, so I can dead-archive 7241b. Also, commit the changes from #7496-22 into 7496a. The goal

here is to have only 7496a fully ready to fix FORWARD-ONLY (including implicit transactions, SCROLLING and ForwardResults).

Once completely finished, let Stefan test the changes as he already has a testing scenario already set-up form #7241. Stefan, consider testing 7496a when ready over the trunk (without 7241c).

#24 - 08/08/2023 03:53 AM - Dănuț Filimon

There are a few things wrong with the patch from #7496-22 which I just found out while testing.

1. TemporaryBuffer does not have a case for TRANSACTION_COMMITTING in sessionEvent(), but other implementations eventually call cleanup()/sessionClosing() and releaseSession() which will decrease the value of Persistence.sessionUseCount. This value is decreased **once** for each commit, but notifying multiple listeners can decrease it even further.
2. When the session event is processed, it will close the results too early and will throw the following error:

```
org.h2.jdbc.JdbcSQLNonTransientException: The object is already closed [90007-200]
        at org.h2.message.DbException.getJdbcSQLException(DbException.java:505)
        at org.h2.message.DbException.getJdbcSQLException(DbException.java:429)
        at org.h2.message.DbException.get(DbException.java:205)
        at org.h2.message.DbException.get(DbException.java:181)
        at org.h2.message.DbException.get(DbException.java:170)
        at org.h2.jdbc.JdbcResultSet.checkClosed(JdbcResultSet.java:3252)
        at org.h2.jdbc.JdbcResultSet.next(JdbcResultSet.java:133)
        at com.goldencode.p2j.persist.orm.ScrollableResults.execute(ScrollableResults.java:440)
        at com.goldencode.p2j.persist.orm.ScrollableResults.next(ScrollableResults.java:167)
        at com.goldencode.p2j.persist.ScrollingResults.next(ScrollingResults.java:156)
```

#25 - 08/17/2023 07:20 AM - Dănuț Filimon

Committed 7496a/rev.14656. I brought the changes made by Stefan in #7241 (7241b) to 7496a.

I've done tests with a customer application and there are still problems when using ForwardResults and TRANSACTION_COMMITTING. Initially I had problem with the results being closed to early or with record hydration, but now I get a overlapping recids when trying to navigate the application basic pages. It logs a SEVERE error which doesn't actually shows the cause of the error, so I still need to investigate it.

I've tried to set and use isNativelyScrolling with false/true and most of my test suite runs properly with either with the exception of a single test that already has a wrong output in FWD compared to 4GL.

Currently there are 15 tests that show different outputs between 4GL and FWD, some of the problems are:

- Trying to display a record after a failed Cannot LAST or Prev Query q when FORWARD-ONLY ... shows No record available. The expected scenario is not showing anything if GET-LAST fails.
- Not releasing the buffer after creating records will keep the last record and will be made available even if GET-LAST fails due to using FORWARD-ONLY. (I think this is a separate issue and the test suite will be modified to release after creating the records)
- Additional error being displayed when using SCROLLING, FORWARD-ONLY and GET-FIRST on a PRESELECT EACH (even if the query uses forward-only, it should be able to retrieve precedent records due to it being preselect)
- Many error such as

```
** No tmp1 record is available. (91)
** No tmp1 record is available. (91)
? ?
```

which do not display the ? ?, just an empty line. (These were not included in the 15 tests I am currently talking about, but the overall number of occurrences is greater and is very noticeable)

- A few more instances of wrong output that are similar to the ones mentioned above, but on a larger test which causes them to mix and not easily identify the problem.

#26 - 08/22/2023 08:19 AM - Dănuț Filimon

- Related to Bug #7737: QueryWrapper should either show or throw errors depending on the query type added

#27 - 08/25/2023 09:56 AM - Dănuț Filimon

Committed 7496a/rev.14705. After fixing #7737 (#7737-12) and investigating TRANSACTION_COMMITTING, I managed to fix 12 out of the 15 failing tests, now I am left with 3 tests that are not caused by the the current changes, some of which are mentioned in the related issue. The problem with using the TRANSACTION_COMMITTING event was that it was possible to delete a record in a temporary table, the results would be cached (including the deleted record which wasn't marked as removed). If you tried to create a new record, it would use the reclaimed recid and would be found in the cache when flushing. This should've been solved only for persistent tables, for which the initial testing was done.

The problems that I found while testing the customer application were solved. I will update the application to fix an existent regression and retest if necessary.

The changes should also be tested in a more detailed manner, most of the TRANSACTION_COMMITTING events that I tested were handled in the AdaptiveQuery.sessionEvent() method, while I rarely got to see the event being handled in PreselectQuery or never (Presorter).

#28 - 08/30/2023 04:31 AM - Alexandru Lungu

- % Done changed from 80 to 100

- Status changed from WIP to Review

Danut, please let me know if there is something left in this task.

Please do a full round of testing with this:

- personal test-cases
- customer applications
- attempt to do a database dump according to #7241
  - make sure you follow the steps in #7241-42. Incrementally convert #7241-33 (export file for each table) and #7241-28 (master file to trigger export). Please make sure it converts correctly. You need to open the server and run #7241-42.
  - attempt the dump - you can do a bit of debugging to see that ProgressiveResults with ForwardResults is used. Extend your debug in SQLQuery and make sure the dumping statements are set for auto-commit false, TYPE_FORWARD_ONLY and fetch-size is non-zero - so a PG server-cursor is used. Currently, I think the auto-commit is by default true when there is no transaction. Consider calling conn.setAutoCommit(false) when you open a session (in the c'tor maybe). This will be fixed separately.
  - attach VisualVM and keep track of the CPU and memory. Report back if the dumping works properly.
  - report if there is any PG portal related error (reading from a invalidated database-cursor).

I will start profiling / testing the changes here on a customer application. If everything works fine, I will consider merging this.

#29 - 08/31/2023 07:17 AM - Dănuț Filimon

Personal tests and customer application were tested previously. I've retested them just to make sure, but I was not successful with the customer application since it required a new setup. Currently I do not need to browse the application, but I had some problems with the incremental version since the archive was outdated (old tables were removed and new ones were added with the new database). The current problem is with the incremental version which I finally made it work, but even after making sure that name_map.xml is updated with the new converted files, FileSystemOps.searchPath() still can't find the file anywhere.

The other problem with the customer application is being addressed in #7710. I previously encountered 403 Forbidden (exactly like the Swing client in #7710-12), but after following the instructions in the issue I managed to display the window, only to be redirected to the login page right after closing it.

#30 - 09/04/2023 06:45 AM - Alexandru Lungu

Rebased 7496a to trunk and is now at rev. 14732.

#31 - 09/04/2023 09:55 AM - Dănuț Filimon

> The current problem is with the incremental version which I finally made it work, but even after making sure that name_map.xml is updated with the new converted files, FileSystemOps.searchPath() still can't find the file anywhere.

Somehow the dumping works properly now, it finds all the files. I tried converting a test file for another issue which didn't work since it couldn't find it and tested the dump out of curiosity. I am currently dumping the database of a customer application and will return with the results after it is done.

#32 - 09/06/2023 02:35 AM - Dănuț Filimon

I dumped the customer application database in **9 hours and 17 minutes**. While monitoring the CPU and Memory using VisualVM, there was nothing unusual. For an hour, it was using between **2-4GB** for small tables and **max 6.6GB** for large tables. I left the process to execute overnight and a total of **940** tables were dumped with ~**110** million records dumped. The largest table had **17** million records and was dumped in **35 minutes**. There were no PG portal related errors and the test was done using auto commit set to false.

#33 - 09/06/2023 03:03 AM - Alexandru Lungu

Dănuț Filimon wrote:

> I dumped the customer application database in **9 hours and 17 minutes**. While monitoring the CPU and Memory using VisualVM, there was nothing unusual. For an hour, it was using between **2-4GB** for small tables and **max 6.6GB** for large tables. I left the process to execute overnight and a total of **940** tables were dumped with ~**110** million records dumped. The largest table had **17** million records and was dumped in **35 minutes**. There were no PG portal related errors and the test was done using auto commit set to false.

Danut, 9h and a bit (#7241-140, #7241-164) is the baseline time without changes. I am quite surprised that with #7496 changes, you are facing the same dumping time. I will request a double check here. AFAIK, a dump with server-cursor was done before and achieved a really low time (#7241-92, but with lots of "temporary patches" both in 4GL and FWD).

Therefore, please do some System.out and debugging kind of job to check the server-cursor is actually used. It should: set auto-commit on false,

results type set on TYPE_FORWARD_ONLY and a fetch-size of 256 (you can even test with more, like 1024). Anyway, you should definitely see some kind of improvement comparing with TYPE_SCROLL_INSENSITIVE.

If the second test fails, I suspect this is due to AdaptiveQuery still being used. However, PreselectQuery can't me made forward-only ... Still need to think about it. Good news is that the dumping still works with no regression.

#34 - 09/06/2023 04:39 AM - Dănuț Filimon

The dump was done using TYPE_SCROLL_INSENSITIVE and ForwardResults wasn't used, I had to modify he original dump file to use a dynamic query with FORWARD-ONLY instead of a simple FOR EACH. I will retest the dump now.

#35 - 09/06/2023 04:55 AM - Alexandru Lungu

Danut, as a debugging hint, I downloaded the PG driver sources and debugged them when ResultSet.next was called. This way, I could identify in rows how many rows were actually fetched from the database. If the database cursor is used, this rows has only <256 records at once. If the server-cursor is not used, the ResultSet may have thousand of records at once.

#36 - 09/06/2023 08:13 AM - Dănuț Filimon

Alexandru Lungu wrote:

> Danut, as a debugging hint, I downloaded the PG driver sources and debugged them when ResultSet.next was called. This way, I could identify in rows how many rows were actually fetched from the database. If the database cursor is used, this rows has only <256 records at once. If the server-cursor is not used, the ResultSet may have thousand of records at once.

Thank you for the hint, the fetch size is 256 and the size of the rows is 256 as well when debugging. I stopped the process closely to the 3 hour mark and managed to dump 21.8GB. I took the dump.log file results and compared them to the previous ones that used TYPE_SCROLL_INSENSITIVE and there is not much of a difference, 2h and 42 minutes when using TYPE_FORWARD_ONLY and 2h and 57 minutes when using TYPE_SCROLL_INSENSITIVE. 710 tables were dumped in this time.

#37 - 12/15/2023 04:07 AM - Dănuț Filimon

I am picking up work on this issue for #7991 and will be testing the changes first to see if there are any obvious problems since 7496a is pretty far behind current trunk.

#38 - 12/15/2023 07:10 AM - Dănuț Filimon

First things first, there are a lot of 23/12/15 12:47:11.619+0200 | SEVERE | com.goldencode.p2j.persist.ForwardResults | ThreadName:Conversation [00000015:bogus], Session:00000021, Thread:00000010, User:bogus | Cannot call setRowNumber on empty or fully iterated results in FORWARD_ONLY mode logs while testing a customer application. During testing, I did not find anything problematic and went ahead and started a POC run. POC threw the following error:

```
Caused by: com.goldencode.p2j.persist.PersistenceException: Failed to populate record for class ...
Caused by: org.postgresql.util.PSQLException: ResultSet not positioned properly, perhaps you need to call next
.
    at org.postgresql.jdbc.PgResultSet.checkResultSet(PgResultSet.java:2772)
    at org.postgresql.jdbc.PgResultSet.getLong(PgResultSet.java:2078)
    at com.mchange.v2.c3p0.impl.NewProxyResultSet.getLong(NewProxyResultSet.java:503)
    at com.goldencode.p2j.persist.orm.SQLQuery.hydrateRecordImpl(SQLQuery.java:899)
```

```
    at com.goldencode.p2j.persist.orm.SQLQuery.lambda$hydrateRecord$5(SQLQuery.java:762)
    at com.goldencode.p2j.jmx.NanoTimer.timer(NanoTimer.java:131)
    at com.goldencode.p2j.persist.orm.SQLQuery.hydrateRecord(SQLQuery.java:762)
    at com.goldencode.p2j.persist.orm.ScrollableResults.lambda$get$0(ScrollableResults.java:243)
    at com.goldencode.p2j.persist.orm.ScrollableResults.execute(ScrollableResults.java:440)
    at com.goldencode.p2j.persist.orm.ScrollableResults.get(ScrollableResults.java:212)
    at com.goldencode.p2j.persist.ScrollingResults.get(ScrollingResults.java:197)
    at com.goldencode.p2j.persist.PreselectQuery.cacheResults(PreselectQuery.java:5592)
    at com.goldencode.p2j.persist.AdaptiveQuery.sessionEvent(AdaptiveQuery.java:2230)
    at com.goldencode.p2j.persist.Persistence$Context.notifySessionEvent(Persistence.java:5374)
    at com.goldencode.p2j.persist.Persistence$Context.commit(Persistence.java:4768)
    at com.goldencode.p2j.persist.TxWrapper.commit(TxWrapper.java:307)
    at com.goldencode.p2j.util.TransactionManager$WorkArea.notifyMasterCommit(TransactionManager.java:10943)
    at com.goldencode.p2j.util.TransactionManager$WorkArea.access$12200(TransactionManager.java:10389)
    at com.goldencode.p2j.util.TransactionManager.lambda$processCommit$3(TransactionManager.java:6876)
    at com.goldencode.p2j.jmx.NanoTimer.timer(NanoTimer.java:131)
    at com.goldencode.p2j.util.TransactionManager.processCommit(TransactionManager.java:6836)
    at com.goldencode.p2j.util.TransactionManager.popScope(TransactionManager.java:4659)
    at com.goldencode.p2j.util.TransactionManager.access$7200(TransactionManager.java:711)
    at com.goldencode.p2j.util.TransactionManager$TransactionHelper.popScope(TransactionManager.java:8831)
    at com.goldencode.p2j.util.BlockManager.doBlockWorker(BlockManager.java:10184)
    at com.goldencode.p2j.util.BlockManager.doBlock(BlockManager.java:1532)
    ...
    at com.goldencode.p2j.util.Block.body(Block.java:636)
```

which I suspect is caused by the processing of the TRANSACTION_COMMITTING event. I will attempt to remove this event (as it was not originally used) then retest. If there are still errors, I'll go ahead and investigate it.

#39 - 12/15/2023 09:10 AM - Dănuț Filimon

- Status changed from Review to WIP

POC tests still fail, I am trying to setup the fwdtests project in Eclipse and debug.

#40 - 12/19/2023 03:06 AM - Dănuț Filimon

Dănuț Filimon wrote:

> First things first, there are a lot of 23/12/15 12:47:11.619+0200 | SEVERE | com.goldencode.p2j.persist.ForwardResults | ThreadName:Conversation [00000015:bogus], Session:00000021, Thread:00000010, User:bogus | Cannot call setRowNumber on empty or fully iterated results in FORWARD_ONLY mode logs while testing a customer application.

It seems that a row is searched in a loop and when it is not found, a new Locator is needed to find that row. The problem is that results used to find the record (which did not find it) and the results from the next bracket are the same, meaning that the Locator used will return the results from the only bracket available. The size of the bracket is 10 and the number of available records is 3, so when searching the row 4 it will be located in the same bracket (**1**).

This can be easily solved by comparing the Locator bracket and the ProgressiveResults bracket. When both are the same, **false** should be returned since the bracket was already searched.

#41 - 12/19/2023 07:38 AM - Dănuț Filimon

Alexandru, I tested POC with the solution mentioned in #7496-40 after we discussed it could provide a performance improvement.
The following patch:

```
=== modified file 'src/com/goldencode/p2j/persist/ProgressiveResults.java'
--- old/src/com/goldencode/p2j/persist/ProgressiveResults.java    2023-10-24 08:12:12 +0000
+++ new/src/com/goldencode/p2j/persist/ProgressiveResults.java    2023-12-19 08:54:44 +0000
@@ -980,6 +980,11 @@
          // to get the proper bracket and move to the target row within it.
          Locator locator = locateRow(row);

+         if (results != null && locator.getBracket() == this.bracket)
+         {
+            return false;
+         }
+
          // If we are caching, retrieve every result set and cache results.
          if (cache != null)
          {
```

got the following results:

	Baseline (4 cold tests avg)	7156b + patch (5 cold tests avg)	Difference (%)
average of the last 20 runs	15113.25	14751.4	-2.39
total average	14821.5	14707.8	-0.76

I also tested a customer application with this change while looking for other bugs. I was not able to identify the issue with the POC tests that cause wrong results, ForwardResults is created and records are retrieved correctly but without the actual code behind it I am not able to find the problem. Still investigating...

#42 - 12/19/2023 07:41 AM - Alexandru Lungu

- Status changed from WIP to Internal Test

This change is invasive as it affects all ProgressiveResults across any application. I am OK with the change: we shouldn't look again in a bracket that we have analyzed already. If we failed the first time through, we are going to fail the second time as well. However, this should be intensively tested.

- Run large application regression tests.
- Run large application smoke tests. Talk with Andrei [ab] / Stefan-Andrei [sat] or Radu to do the same tests on their application.
- Run Majic regression tests with 6667e and this change.
- Run custom tests you wrote.
- Run adaptive-scrolling tests.

I would like to have this in trunk asap.

#43 - 12/20/2023 05:40 AM - Dănuț Filimon

I finished testing the patch from #7496-41:

- POC tests: passed;
- POC regression tests: passed (with the expected number of failing tests);
- Smoke tested a large application: passed;
- Radu helped me test another large application: passed;
- Majic regression tests: passed;
- Custom tests: passed;
- Adaptive-scrolling tests: passed.

This change is good to go. Alexandru, where should I commit the patch so that it can be merged asap?

#44 - 12/20/2023 08:25 AM - Dănuț Filimon

Committed 7496b/rev.14879 Added the patch from #7496-41.

#45 - 12/21/2023 04:57 AM - Alexandru Lungu

Greg, this seems like an easy performance gain. It was fully tested (except ETF). We can go ahead with the merge of 7496b.

#46 - 12/21/2023 07:52 AM - Greg Shah

You can merge 7496b to trunk now.

#47 - 12/21/2023 10:07 AM - Alexandru Lungu

Doing it now - just 10 mins to check it out.

#48 - 12/21/2023 10:50 AM - Alexandru Lungu

Branch 7496b was merged into trunk revision 14891 and archived.

This task is still open due to 7496a.

#49 - 12/22/2023 12:31 AM - Dănuț Filimon

- % Done changed from 100 to 50

- Status changed from Internal Test to WIP

Since 7496b was merged, I'll put this issue into WIP. I've managed to find a few other issues with the implementation when running the adaptive-scrolling test set. I've also found 3 failing tests at the beginning of the ChUI regression tests, but I did not manage to connect using the configuration specified in the Wiki and examine them. Another large application has no problem when debugging using a similar configuration so I am thinking this is a configuration issue.

#50 - 01/17/2024 09:45 AM - Dănuț Filimon

- % Done changed from 50 to 70

I managed to solve all the regressions from the adaptive-scrolling test set and identify other problems in the POC while debugging a create test (but only partially solving them).

The issues include:

- resetting the scrolling when setting the forwardOnly flag;
- CompoundQuery components that had a different value for forwardOnly when being added through addCompoundComponent;
- ~~Result ids not being added to the cursor due to isScrolling() && !isPresorted() condition (the record was found and it was available, but the cursor did not update accordingly);~~

I am still debugging the POC project and I am working on fixing the fetch test, which still does not return the expected result. I also narrowed down the problem to the isScrolling() method (maybe the !isForwardOnly() check is not actually correctly used here). The problem is that there is no instance where forwardOnly is set to true for the fetch test when using 7496a, but it is used 3 times without those changes, so I suspect that another query that was previously created is spoiling the test.

POC tests were done with 7156b/rev.14933 + 7496a (and the changes mentioned above).

#51 - 01/30/2024 03:48 AM - Dănuț Filimon

- File 7496a-OffEnd.p added

- File fwd.df added

Rebased 7496a with **trunk/rev.14950**. **7496a** is not at revision **14959**. I also committed the adjustments which fixes most of the problems in **7496a/rev.14960**. The changes include:

- Removing the **TRANSACTION_COMMITTING** event;
- The cursor will not be reset when the forwardOnly flag is assigned.
- New CompoundComponent(s) added should also set the forwardOnly flag.
- When creating a DynamicQuery, the forwardOnly flag should also be set. (I'm also now looking into more instances where a query is created in this way).

I've been debugging POC for a while and finally managed to find the main problem of a create test. The main clue was a DYN AdaptiveQuery which becomes a PRE and a batch create that should make it back into DYN is not executed. I've attached a minimal test for this problem and the .df used.

The test executes GET-FIRST and creates a record since none is available. At this point the query is not OFF-END and it executed GET-NEXT which should make it OFF-END and finally create a new record. With 7496a, the record is not created at the end. From my investigation, the first record is not found at the point of executing GET-NEXT and this causes the query to find that record (in this case the query will not be OFF-END). I am not sure how to fix it, this looks like a problem related to the way the created record is considered to already be found at the time of executing GET-NEXT.

#52 - 01/30/2024 04:14 AM - Alexandru Lungu

Danut, can you cut-out it even more.

1. Please check if this can be reproduced with temp-tables - it is a bit easier to follow. Remove the unnecessary fields. Is the index relevant?
2. Is this a QUERY-OFF-END problem or an AdaptiveQuery problem? Please replace the "<buffer-create and assign>" with "message" where possible. It is easier to follow and test.

#53 - 01/30/2024 04:55 AM - Dănuț Filimon

- File 7496a-minimal-fixed.p added

Alexandru Lungu wrote:

> Danut, can you cut-out it even more.
>
> 1. Please check if this can be reproduced with temp-tables - it is a bit easier to follow. Remove the unnecessary fields. Is the index relevant?

It can't be reproduced with temp-tables, the persistent table, index, transaction, batch create are important.

> 2. Is this a QUERY-OFF-END problem or an AdaptiveQuery problem? Please replace the "<buffer-create and assign>" with "message" where possible. It is easier to follow and test.

This is related to AdaptiveQuery and mostly how it handles the get-first result. I adjusted the test as you mentioned and attached it. I've reduced the number of fields, the prepare string and removed the second create.

#55 - 01/31/2024 04:41 AM - Alexandru Lungu

- Related to Bug #7991: use ScrollingResults instead of ProgressiveResults when FORWARD-ONLY mode and database cursors are available. added

#56 - 02/02/2024 02:15 AM - Dănuț Filimon

After taking the minimal test case and looking into it, I noted the following:

- with no changes, the cursor already found the first record and running cursor.getNext() will return **null**. In this case the current record of the buffer is replaced with **null**;
- with 7496, a RandomAccessQuery is created. The current record of the RecordBuffer is not replaced with null after calling results.next(), it is cached before any statement is executed and the record is found afterwards (since it was made visible through honorRecordNursery).

I am not sure how to approach this problem. In the first place, the cursor would have no rows to show and in this case (with changes), the record is made available too soon.

#57 - 02/12/2024 02:39 AM - Dănuț Filimon

Rebased **7496a** with **trunk/rev.14981**. **7496a** is not at revision **14991**.

Files

7496a-OffEnd.p	912 Bytes	01/30/2024	Dănuț Filimon
fwd.df	561 Bytes	01/30/2024	Dănuț Filimon
7496a-minimal-fixed.p	838 Bytes	01/30/2024	Dănuț Filimon