Database - Feature #2175

add support for -rereadnolock startup parameter

08/28/2013 03:44 PM - Eric Faulhaber

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Ovidiu Maxiniuc	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	Runtime Support for Server Features		
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Database - Feature #1594: add conversion and runtime support for C...	**Closed**	**10/17/2012**	**07/22/2013**
Related to Database - Bug #2176: exceptions to rereadnolock	**New**		

History

#1 - 08/28/2013 03:46 PM - Eric Faulhaber

The current persistence implementation does not support the feature provided by the -rereadnolock startup parameter. This must be configurable in the directory, and should be disabled by default (as in Progress).

See http://knowledgebase.progress.com/articles/Article/P109264 and related articles for information on how this parameter works.

#2 - 08/29/2013 10:04 AM - Ovidiu Maxiniuc

Testing environment. The article is not very explicit so I will explain the testcase.
Suppose we have a table person with last-name a column that is not part of the primary index with the first record last-name = "alpha", and the following two clients:
A:

```
DEFINE BUFFER p2 FOR person.
FIND FIRST p2 NO-LOCK.
DISPLAY "1: " + p2.last-name.

PAUSE. /* wait for B here */

FIND FIRST person NO-LOCK.
DISPLAY "2: " + person.last-name.
```

B:

```
FIND FIRST person.
DISPLAY "1: " + last-name.
ASSIGN last-name = "beta".
DISPLAY "2: " + last-name.
```

We first launch the A client and while it is paused, the second client is executed, then A is let to do the rest of the job.
If a is launched without -rereadnolock command-line parameter, it will print
1: alpha 2: alpha
With the above parameter, A will re-read the buffer so it will output:
1: alpha 2: beta.

It is important to note that A has two buffers. If A had been written like:

```
FIND FIRST person NO-LOCK.
DISPLAY "1: " + person.last-name.

PAUSE. /* wait for B here */

FIND FIRST person NO-LOCK.
DISPLAY "2: " + person.last-name.
```

Then the change is visible (A prints 1: alpha 2: beta) whether or not the -rereadnolock was supplied.

#3 - 08/30/2013 11:57 AM - Ovidiu Maxiniuc

This is a command-line parameter for the client, it's proper location is not the directory.xml.
In 4GL I can have 2 different clients, simultaneously connected to same server, displaying different results due the extra -rereadnolock parameter of one of them.
How can I specify it ? I mean... this flag should get to P2J server in the context of the respective user to be available at the moment of processing the queries. Is there any mechanism at connection handshake to pass this piece of information I do not know?

#4 - 08/30/2013 12:35 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> This is a command-line parameter for the client, it's proper location is not the directory.xml.

All command-line client parameters we currently support are configured via the directory.

> In 4GL I can have 2 different clients, simultaneously connected to same server, displaying different results due the extra -rereadnolock parameter of one of them.
> How can I specify it ? I mean... this flag should get to P2J server in the context of the respective user to be available at the moment of processing the queries. Is there any mechanism at connection handshake to pass this piece of information I do not know?

At some point in the future, we plan to provide a feature to honor command line arguments, allowing them to override startup parameters specified in the directory for both the server and the client, but for now the directory is the only way to configure this. See, for example, how we deal with the -d (date format) client internationalization parameter (directory path is server/default/runtime/default/dateFormat).

See also the P2J Runtime Installation Configuration and Administration Guide (Part 2 -> Configuration -> Directory -> Progress 4GL Runtime Compatibility Options).

#5 - 08/30/2013 03:07 PM - Ovidiu Maxiniuc

My first instinct was to put this flag somewhere in the database branch which was a little cumbersome to extract. I finally read the value from the same location as date/number format and other client-side system-related parameters.

There is only one issue. How can I know if the respective record have been already retrieved into another buffer ?
I need to know this because if there is only one buffer that request the same record, the value obtained is always up-to-date, whether rereadnolock was specified. On the other hand, if a second buffer is expected to retrieve a record existing into another buffer it will not be refreshed except if the rereadnolock is active. Apparently this we for each record we need to know if there is another buffer that holds already it. If there is, we use the current value, if I am the only buffer then do a refresh.

Here is an interesting piece of code:

```
A:
DO TRANSACTION:
   FIND FIRST person EXCLUSIVE-LOCK.
   ASSIGN person.last-name = "alpha".
END.
FIND FIRST person NO-LOCK.
DISPLAY "BEFORE: " person.last-name + "" SKIP.
PAUSE. /* wait for B here */

DEFINE BUFFER p1 FOR person.
FIND FIRST p1 NO-LOCK.
DISPLAY "AFTER1: " p1.last-name + "" SKIP.

FIND FIRST person WHERE person.last-name EQ "Beta" NO-LOCK.
DISPLAY "AFTER2: " person.last-name + "" SKIP.
```

B:

```
FIND FIRST person.
ASSIGN person.last-name = "Beta".
```

Running B while A pauses, you'll get 3 alpha s even though in the last query of A a record with person.last-name EQ "Beta" was explicitly requested.

#6 - 09/02/2013 11:37 AM - Ovidiu Maxiniuc

- File om_upd20130902a.zip added

Update pack for review. This update is built over the om_upd20130829a.zip of #1594.
The implementation looks into BufferManger for the queried dmo to see if it is referred in multiple buffers and emulate the no-rereadnolock behavior of 4GL of using the old cached record by not refreshing it from db server. If the parameter is specified, the refresh is forced in load with no-lock / no-lock case.

#7 - 09/03/2013 01:42 AM - Eric Faulhaber

I haven't finished a detailed review yet, but I have a basic question on your approach: are you trying to determine whether a DMO is in use by another buffer *across contexts* or *within the current context*? If it's the former, BufferManager isn't going to tell you what you want. There is a separate instance of BufferManager for each session. It doesn't know anything about buffers in other users' sessions.

There is a class in the persist.dirty package, GlobalEventManager, which is a bit closer to what we would need for determining change across sessions, but it's not quite right for that either, since it only tracks changes to fields contained by indexes.

If I've misunderstood your intent, please clarify.

#8 - 09/03/2013 09:48 AM - Ovidiu Maxiniuc

From my researches on 4GL and docs, there is nothing to indicate that Progress looks into other user's buffers to see if the record is referred there. The docs specify that the parameter will *"resolve client-server currency conflicts"*; so one client is not aware of the buffers of the others, as the client-server P2J architecture is slightly different than 4GL's.
I also did the following test that proves that only buffers of the current user are checked.
A:

```
DO TRANSACTION:
  /* make sure the customer and person with 0 id are set to [alpha] */
  FIND FIRST customer WHERE customer.id EQ 0 EXCLUSIVE-LOCK.
  FIND FIRST person WHERE person.id EQ 0 EXCLUSIVE-LOCK.
  ASSIGN
    customer.name = "alpha"
    person.last-name = "alpha".
END.

FIND FIRST person WHERE person.id EQ 0 NO-LOCK.
FIND FIRST customer WHERE customer.id EQ 0 NO-LOCK.
DISPLAY "Before" SKIP
        "p1: " person.last-name + "" SKIP
        "c:  " customer.name + "" SKIP WITH NO-LABELS.

PAUSE. /* now wait for B to alter the records */

DEFINE BUFFER p1 FOR person.
FIND FIRST p1 WHERE p1.id EQ 0 NO-LOCK.
FIND FIRST customer WHERE customer.id EQ 0 NO-LOCK.

DISPLAY "After B changed it" SKIP
        "p1: " person.last-name + "" SKIP
        "p2: " p1.last-name + "" SKIP
        "customer: " customer.name + "" SKIP.

RELEASE p1.
FIND FIRST person WHERE person.id EQ 0 NO-LOCK.

DISPLAY "After release" SKIP
  "person: " person.last-name + "" SKIP.
```

And B:

```
DO TRANSACTION:
   FIND FIRST person WHERE person.id EQ 0.
   FIND FIRST customer WHERE customer.id EQ 0.
   DISPLAY "Before: " person.last-name + "" customer.NAME + "" SKIP.
   ASSIGN 
         person.last-name = "Beta".
         customer.name = "Beta".
   DISPLAY "After:  " person.last-name + "" customer.NAME + "" SKIP.
END.

DEFINE BUFFER p1 FOR person.
DEFINE BUFFER c1 FOR customer.
FIND FIRST p1 WHERE p1.id EQ 0 NO-LOCK.
FIND FIRST c1 WHERE c1.id EQ 0 NO-LOCK.

PAUSE. /* keep p1/c1 buffers holding the reference to respective person / customer */

DISPLAY "Still holding reference to: " p1.last-name + "" c1.NAME + "" SKIP.
```

We run the clients this way:

- launch A, it will reset the person.last-name and customer.name to "alpha", display them with the label "before".
- while A pauses, launch B. I will find those records, display the initial value ("alpha"), change both fields to "Beta" and re-display them. Then, create a second buffer for each record, load them then pauses.
- now allow A to continue. In first step, both buffers for person will display the old content while the one for customer will reflect the correct one. After releasing the p1 buffer and re-applying the query, we can see that 4GL will print the correct value for person, too.
- we can let client B finish and printing again the actual values of the records held by p1/c1 buffers.

We can see that for customer the buffer is always up-to-date as there is only one buffer holding the record with id = 0. For person, as long as there is a second buffer referring that record, the value is the cached one. As the second buffer releases the record, the correct value will be found. And this while the second client keeps references to both records.

The conclusion is that in 4GL client does not see the records buffered by other clients, only buffers form current client are taken into consideration. In P2J parlance, this is equivalent of handling buffers from one client context are rather independent from the other context.

#9 - 09/03/2013 11:06 AM - Eric Faulhaber

OK, thanks for the explanation. The implementation looks good to me, just minor change requests:

- For consistency of naming, in Persistence$Context, please change the variable name reRead_NoLock to rereadNoLock and the method name rereadNoLock() to getRereadNoLock().
- Typo at line 2111 in Persistence ("hols" -> "holds")

Assuming these are the only changes you make, I don't need to review this again before you regression test. If it passes regression testing, please check it in and distribute.

#10 - 09/03/2013 11:23 AM - Eric Faulhaber

One more thing: please use proper logging in your RecordBuffer changes instead of System.err.println.

#11 - 09/03/2013 12:13 PM - Eric Faulhaber

Eric Faulhaber wrote:

> rereadNoLock() to getRereadNoLock().

Sorry, for consistency, I guess that should be isRereadNoLock().

#12 - 09/03/2013 01:20 PM - Ovidiu Maxiniuc

I found another issue. When the search for a record is done using the ROWID or RECID, the record is not re-fetched from db, instead the value found in any existing buffer. Only if there is only one buffer the value is refreshed from db.
In other words if a record is looked up using its ROWID or RECID Progress will behave like the rereadnolock is off even if it was explicitly added to command-line.
Current implementation does not honor this and, for the moment I don't know how to deal with this. I am thinking of checking for these functions ROWID or RECID in the hql parser. The difficulty here is to detect the form of the where hql expression the ROWID or RECID appear in order to qualify as the above exception.
For example:
FIND FIRST person WHERE crecid EQ RECID(person) NO-LOCK. - will not re-fetch in rereadnolock and multi-buffer
FIND FIRST person WHERE crecid EQ RECID(person) OR person.id EQ 0 NO-LOCK. - will not re-fetch in rereadnolock and multi-buffer
but
FIND FIRST person WHERE crecid EQ RECID(person) OR otherid EQ RECID(person) NO-LOCK. - will do re-fetch in rereadnolock and multi-buffer

Do you have any idea here ?

#13 - 09/03/2013 01:54 PM - Eric Faulhaber

<sigh> I can't make any sense of the "logic" that must be behind these results. Looks like a quirk/bug of their parser that will take some larger range

of test cases to pinpoint. Kudos for finding it.

Ultimately, we'll have to track this down and emulate it, but I don't want to hold up integrating the good work you've done so far on this feature, which I think handles the majority of use cases. Please open a separate (bug) issue for this and make it related to this one, but don't assign a target version or start date. Document the limitation in the code as well (in the place you feel makes the most sense), then continue with the clean-up and regression testing.

#14 - 09/04/2013 10:53 AM - Ovidiu Maxiniuc

- File om_upd20130904a.zip added

Update merged with latest bzr sources and changes from review. I already re-tested it with my sample tests.
I am running at this moment the runtime regression on it.

#15 - 09/05/2013 02:32 PM - Ovidiu Maxiniuc

The runtime test failed.
Some CTRL+C test failed and 3 reports show results different than expected.
I started discussing with Constantin to help me detect the cause.

#16 - 09/17/2013 04:11 PM - Eric Faulhaber

- Status changed from New to WIP

#17 - 09/19/2013 03:56 AM - Ovidiu Maxiniuc

The update om_upd20130912e.zip added in #1594-46 of the parent issue contains the needed changes for this issue, too.
It passed the regression testing, was committed to bzr as revision 10381 and distributed by email.

#18 - 09/19/2013 01:37 PM - Eric Faulhaber

- Status changed from WIP to Closed

#19 - 11/16/2016 11:43 AM - Greg Shah

- Target version changed from Milestone 7 to Runtime Support for Server Features

#20 - 02/01/2023 01:10 PM - Greg Shah

- Related to Bug #2176: exceptions to rereadnolock added

Files

om_upd20130902a.zip	198 KB	09/02/2013	Ovidiu Maxiniuc
om_upd20130904a.zip	198 KB	09/04/2013	Ovidiu Maxiniuc