Database - Bug #8259

Reduce the work done by PreselectQuery.coreFetch

02/08/2024 04:29 AM - Alexandru Lungu

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#1 - 02/08/2024 05:55 AM - Alexandru Lungu

Eduard detected recently that there is some work being done in PreselectQuery.coreFetch regarding an extra fetch.

After some analysis, there is the following conditional that triggers an extra fetch (eventually from session cache):

```
// if we have an ID only, a cached DMO, or we are working with a temp-table, fetch
// the current version of the record (dirty DMO copy excepted)
boolean doFetch = (!fullRec || isResultSetCached() || !buffer.isTemporary()) && !dirtyCopy;
```

- If the record is not full, I think it is clear that we should fetch it from the session cache / database. **This can be kept**
- If the result-set is cached, I don't think there should be much worry. Why would it make a difference if the results are an in-memory SimpleResults or a serialized ScrollingResults - they can both become stale. This stale problem is already resolved before-hand. **This can be removed**
- If the buffer is persistent, the only problem is in regard to the locking. However, is the whole session.get logic still needed? This does a session cache look-up to retrieve the same DMO we already have in coreFetch.
  - The only problem may come when needing to upgrade lock - this can be handled specially.
  - Another problem may be if the DMO is stale. This can also be handled directly inside coreFetch.
  - Otherwise, we may use the DMO we have from the results right away.

My whole concern here is that we do an extra session cache look-up for almost any record that comes into a PreselectQuery buffer (from persistent database). I need to measure first how slow is this process. The rough idea is to replace that code with:

```
// if we have an ID only, a cached DMO, or we are working with a temp-table, fetch
// the current version of the record (dirty DMO copy excepted)
boolean doFetch = (!fullRec || updateLock || dmo.isStale()) && !dirtyCopy;
```

Mind that the temporary condition is implicitly negated by updateLock and isStale is a natural condition here anyway.

#2 - 02/08/2024 07:20 AM - Eduard Soltan

Alexandru Lungu wrote:

> My whole concern here is that we do an extra session cache look-up for almost any record that comes into a PreselectQuery buffer (from persistent database). I need to measure first how slow is this process. The rough idea is to replace that code with:
>
> ```
> // if we have an ID only, a cached DMO, or we are working with a temp-table, fetch
> // the current version of the record (dirty DMO copy excepted)
> boolean doFetch = (!fullRec || updateLock || dmo.isStale()) && !dirtyCopy;
> ```
>
> Mind that the temporary condition is implicitly negated by updateLock and isStale is a natural condition here anyway.

I think that assignment for updateLock should also be upadated, as another fetch from cache/database should happen only if updateLock is EXCLUSIVE.

```
updateLock = (lt == LockType.EXCLUSIVE || lt == LockType.EXCLUSIVE_NO_WAIT);
```

#3 - 02/08/2024 08:03 AM - Alexandru Lungu

I wonder if NO-LOCK to SHARE-LOCK is considered an lock update.

#4 - 02/08/2024 08:49 AM - Eric Faulhaber

Alexandru Lungu wrote:

> I wonder if NO-LOCK to SHARE-LOCK is considered an lock update.

It is. Any change in lock level (none/share/exclusive) is considered a lock update.

#5 - 02/08/2024 09:08 AM - Alexandru Lungu

Eduard, this basically means that the condition in #8259-2 is incorrect.

Please post here a cut-out to the problem you have now with the locking.

#6 - 02/09/2024 03:58 AM - Eduard Soltan

For now in we assign to updateLock the following expression: updateLock = (lt != LockType.NONE);.

I think that in order to see if a lock should be changed we should verify if the current lock on the record is different that that send as parameter. updateLock = (lt != compLock);

#7 - 02/09/2024 04:04 AM - Alexandru Lungu

Eduard Soltan wrote:

> For now in we assign to updateLock the following expression: updateLock = (lt != LockType.NONE);.
>
> I think that in order to see if a lock should be changed we should verify if the current lock on the record is different that that send as parameter. updateLock = (lt != compLock);

Eduard, we need a more large overview of a case that is no working with the current implementation and requires updateLock = (lt != compLock);. Can this be replicated with trunk or only with your changes.

#8 - 02/09/2024 04:57 AM - Eduard Soltan

Alexandru Lungu wrote:

> Eduard Soltan wrote:
>
> > For now in we assign to updateLock the following expression: updateLock = (lt != LockType.NONE);.
> >
> > I think that in order to see if a lock should be changed we should verify if the current lock on the record is different that that send as parameter. updateLock = (lt != compLock);
>
> Eduard, we need a more large overview of a case that is no working with the current implementation and requires updateLock = (lt != compLock);. Can this be replicated with trunk or only with your changes.

It only can be seen with changes in #7999.
In case of a for each book fields (isbn) query, an select book.isbn from book sql query is created. And on any of retrieve operation operation (first/last/next/prev), a record with just a field is retrieved from database. But in coreFetch, it enters in if (doFetch) and persistance.load loads the entire record.