Database - Bug #5281

BUFFER-COPY does not report changes in some modes

04/23/2021 02:47 PM - Eric Faulhaber

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	**version:**	

Description

History

#1 - 04/23/2021 02:48 PM - Eric Faulhaber

- Project changed from Regression Testing to Database

#2 - 04/23/2021 03:03 PM - Eric Faulhaber

When any change is made to a record, RecordBuffer must invoke reportChange to notify registered listeners of the change via the ChangeBroker. This occurs in the RecordBuffer invocation handler code (for individual property updates) or during endBatch processing (for property updates collected in a batch assign or in a buffer-copy operation).

We recently introduced two optimizations which need to do this notification as well: the separate, fast path buffer-copy implementation (RecordBuffer.fastCopy), and the direct access mode within the "classic" buffer-copy implementation.

Adrian posted the following in a separate task, while we were working a related issue:

> Eric Faulhaber wrote:
>
> > But there is more... In RecordBuffer.fastCopy, we are not collecting the unreported changes and then reporting them after the batch copy is finished. We are setting and unsetting the bulkCopy flag in that method, but that is only useful for the invocation handler path, which we are bypassing in fastCopy. We need a "bulk" way to collect the unreported changes (i.e., pairs of before/after field diffs), or we may have other problems resulting from going down the fastCopy path. It looks like this would need to be collected in BaseRecord.setAllDatum (which we probably should rename to setAllData).
>
> Right, a fast-copy won't report the changes. However, this means that the "regular" copy RecordBuffer.copy(DataModelObject,DataModelObject,Map<DatumAccess,DatumAccess>) also has the same issue. Some fields are updated using OrmUtils.setField(), but those changes aren't reported. We should find a way to fix both.
>
> - For the "one-by-one" buffer-copy, we can simply use BaseRecord.getActiveUpdateDiffs() and RecordBuffer.addUnreportedChanges().
> - For the fast-copy, I agree with you. BaseRecord.setAllDatum() collects them, but the reporting should be done by RecordBuffer.fastCopy().
>
> To be fixed: right now I see that setter used inside RecordBuffer$Handler.invoke() can invalidate activeBuffer (currentRecord.setActiveBuffer(null);). This can be dangerous for RecordBuffer.copy as activeBuffer can be invalidated right in the middle of the buffer copy.

#3 - 04/23/2021 03:15 PM - Eric Faulhaber

Adrian Lungu wrote:

> To be fixed: right now I see that setter used inside RecordBuffer$Handler.invoke() can invalidate activeBuffer (currentRecord.setActiveBuffer(null);). This can be dangerous for RecordBuffer.copy as activeBuffer can be invalidated right in the middle of the buffer copy.

I wrote setActiveBuffer with the intention of only using it from the invocation handler, so the invocation handler set it and unset it by design. BaseRecord.setDatum was the only method that was supposed to have a dependency on activeBuffer, and setDatum was only meant to be called by DMO setter methods. It was not possible to pass the buffer as a parameter to setDatum, because this method is called from the subclass DMO's setter methods which have no knowledge of RecordBuffer. I tried to express this restriction in the javadoc, but I was not clear enough. When a call to setDatum from OrmUtils.setField was later added, this fragile requirement was broken. The use of activeBuffer has since been expanded beyond this original intent.