Database - Bug #11705

Inner undoable block rollbacks the outer transactional block

08/04/2026 02:34 AM - Artur Şcolnic

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Artur Şcolnic	**% Done:**	90%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**reviewer:**	Ovidiu Maxiniuc
vendor_id:	GCD	**production:**	No
case_num:		**env_name:**	
version_reported:		**topics:**	
version_resolved:			

Description

History

#2 - 08/04/2026 02:38 AM - Artur Şcolnic

This is only one of the bugs Constantin found in #11625, whereas others seem more like corner cases, this bug is a pretty obvious one and must be solved urgently. The test case is

```
do transaction:
   create Book.
   assign Book.book-id = 1 Book.isbn = "ISBN1" Book.book-title = "outer".
   do on error undo, leave:
      assign Book.book-title = "inner".
      undo, leave.
   end.
end.

find first book.
if avail book then message book.book-title.
```

The inner block causes the outer transaction to rollback, thus no record is created.

#3 - 08/04/2026 02:38 AM - Artur Şcolnic

- Status changed from New to WIP

- Assignee set to Artur Școlnic

#4 - 08/05/2026 06:24 AM - Artur Școlnic

- Status changed from WIP to Review

- reviewer Ovidiu Maxiniuc added

The conclusion that the inner undo rollbacks the outer transaction was a bit hasty on my part, in fact the issue is different, but with a similar effect.
In the logs I observed

```
26/08/05 13:13:28.897+0300 | WARNING | com.goldencode.p2j.persist.orm.Persister | ThreadName:Conversation [000
00001:bogus-as], Session:00000001, ThreadId:00000005, User:bogus | Failed to UPDATE #2 of com.goldencode.datas
et.dmo.fwd.Book__Impl__._com.mchange.v2.c3p0.impl.NewProxyPreparedStatement@395eb732 [wrapping: update book se
t book_id=('1'::int4), book_title=('outer'), isbn=('ISBN1') where recid=('2'::int8)]
```

which means the insert somehow was emitted as an update, which rightfully failed since the record does not exist. The issue is records state, not necessarily the rollback/write semantics, after the inner undo, the DMO state was CHANGED only, so FWD decided the record exists and needs to be updated.
The changes in 11705a address this issue by correctly tracking the DMO state throughout all nesting levels.
The main changes are:
- ChangeSet.java — logStateChange is now specified to take the state after the change, matching logDataChange, which already records post-change values.
- BaseRecord.java — the three callers now pass the real post-change state word instead of a flag mask. In updateState that meant preparing the change set before applying the flags (so the baseline snapshot stays pre-change) and logging after.
- SavepointManager.java + Session.java — record the post-write state at the nesting level that owns the write, not the level that happened to run the DML. DmlSavepoint gains ownerTxLevel; Session.save() passes it down.
Ovidiu, please review 11705a.

#5 - 08/05/2026 02:56 PM - Greg Shah

- % Done changed from 0 to 100

#7 - 08/05/2026 03:39 PM - Ovidiu Maxiniuc

I read the code a few times and it looks sane in diff/meld. However, something else bothers me and I would like to debate, staring from this note fragment:

Artur Școlnic wrote:

> - ChangeSet.java — logStateChange is now specified to take the state after the change, matching logDataChange, which already records post-change values.

I do not think this is correct. The states should reflect the state of the record as it was before the (sub-)transaction t. In the event of the undo/rollback of t that is the state to which we want to set back before 'stepping' into t. And the values from changes should also honour this rule: save the 'before' image so that in the event t is rolled back we use the values from changes for restoring the fields in the record as they were 'before'.

Note: the marked array is not affected. It stores the positions where a change occurred, so walk is possible in both directions.

#8 - 08/06/2026 03:04 AM - Artur Școlnic

- Status changed from Review to WIP

- % Done changed from 100 to 90

#9 - 08/10/2026 08:38 AM - Alexandru Lungu

This looks like #6923.

#10 - 08/10/2026 09:09 AM - Ovidiu Maxiniuc

Yes. It looks like it.

I also encountered the same problem while working on a different task. I tried to get it fix in the respective task, but this is not that simple like a 'secondary' fix. And I had to revert my changes, finally working around the issue as much as I could.

Therefore, I wonder whether we can intensify work in this known, rather complex issue, while it is in normal priority. Having a customer depending on it would raise the priority and the stress associated with it.