Database - Feature #7466

Using a different database connection to work around sequence specific issues

06/23/2023 08:21 PM - Ovidiu Maxiniuc

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

Description

History

#2 - 06/23/2023 08:46 PM - Ovidiu Maxiniuc

This task is related to #7270 so that the approaches used for working around sequences issues can be discussed and documented in public.

The root problem in the related task was a deadlock of the client threads while accessing MariaDb dialect. It seems to be an older issue and was documented here: https://jira.mariadb.org/browse/MDEV-13713.

I encountered some difficulties in isolating the testcase to force a timed synchronization to make the issue observable. My initial procedure(s) involved only sequences, because that was how I successfully duplicated the problem using solely the MariaDb console client (that is, without FWD). I only been able to get the deadlock of the Conversation threads only when I added a basic operations with a permanent table. The final isolated testcase is this:

```
MESSAGE "initial:    " NEXT-VALUE(seq1).

DO TRANSACTION:
    FIND FIRST Book. MESSAGE isbn. isbn="333". RELEASE Book.
    MESSAGE "~t~tin trans:   " NEXT-VALUE(seq1).
    CURRENT-VALUE(seq1) = 0.
    MESSAGE "~t~tafter reset:" NEXT-VALUE(seq1).

    PAUSE. // waiting for other to act

    MESSAGE "~t~tbefore undo:" NEXT-VALUE(seq1).
    UNDO, LEAVE.
END.

MESSAGE "after trans:" NEXT-VALUE(seq1).
FIND FIRST Book. MESSAGE isbn.
```

If a secondary client was executed during the WAIT statement, neither of them would ever finished because they will deadlock each other. The initial one waiting a response from database, the second waiting the Java lock of the first to be released.

The solution added in 7270a makes use of temporary sessions for exclusive use of sequences on MariaDb dialect. The flaw was not observed with other dialects but this might be an opportunity to switch all sequence handlers to use similar solutions for freeing the main transactional session form sequence operations which are not bound to transactions/undo events.

Notes:

- The r14633 is not optimized, just a prof that the original problem can be avoided.
- There might be another issue related to undo events. The initial value of Book.isbn was abc but after running the testcase is gets saved as 333. I does not seem correct to me, I was expecting the final output of Book.isbn to remain unchanged after the transaction block was rolled back by UNDO, LEAVE. I did not do investigations in this regard because my focus was at the sequence deadlock issue.

#3 - 06/26/2023 08:17 AM - Greg Shah

You are saying that in the 4GL, all sequence usage is outside of a transaction and never will be rolled back if there is an error?

#4 - 06/26/2023 03:06 PM - Ovidiu Maxiniuc

Yes. Here is the quote from the official OpenEdge manual (Working with sequences):

> *"A sequence number is generated independently of the transaction which is committing or rolling back. It is possible that individual sequence numbers will appear to be skipped because they were generated and used in a transaction that ultimately rolled back. Sequence numbers are generated independently of tables so they can be used for more than one table. "*