Database - Bug #11811

Deadlock in distributed LockManager implementation.

09/01/2026 03:46 AM - Eduard Soltan

Status:	Review	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**reviewer:**	Constantin Asofiei
vendor_id:	GCD	**production:**	No
case_num:		**env_name:**	
version_reported:		**topics:**	
version_resolved:			

Description

History

#1 - 09/01/2026 04:03 AM - Eduard Soltan

- Status changed from New to WIP

In cluster mode, FWD deliberately turns off the in-JVM notifyAll signal, because the wake-up is supposed to arrive as a Geode event instead. Dirty-share locks were moved to a local, non-Geode implementation — but the tap stayed switched off. So a release frees the lock and nobody ever tells the waiting thread.

Mode	How the waiter parks	Who wakes it
Single JVM	status.wait() on the lock-status object	The releaser calls status.notifyAll() — direct, same object
Cluster	same wait(), but lock state also lives in a shared Geode region	A release anywhere fires a Geode cache event; GeodeUpdateListener catches it and notifies

Because the cluster path has its own notifier, the local notifyAll() is intentionally disabled when clustering is on. That's the !ClusterConfig.isEnabled() guard in InMemoryLockManager and LockStatusImpl.

The problem comes when a InMemoryLockManager for dirty table is used. It is local, and require status.notifyAll() to be called. However it don't pass the !ClusterConfig.isEnabled() check. I have some changes to correct that.

#3 - 09/01/2026 04:07 AM - Eduard Soltan

- % Done changed from 0 to 100

- Status changed from WIP to Review

- reviewer Constantin Asofiei added

Committed on 11811a, rev. 16732.