Database - Feature #4019

rework dirty database implementation to use persistent database

04/01/2019 11:06 AM - Eric Faulhaber

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

Description

History

#1 - 04/01/2019 11:35 AM - Eric Faulhaber

This task is meant to address two issues with the current dirty database implementation:

- it is limited, in that it can't support queries which involve more than one table (such as a query on one table which has an embedded CAN-FIND on another table in its WHERE clause);
- it introduced a performance bottleneck, in that it requires separate queries into an embedded H2 database both for validation of inserted/updated records, and to augment the results of a primary query (e.g., FIND FIRST).

The idea of this task is to explore the feasibility of an alternate implementation. Dirty database inserts and updates currently made into the embedded H2 dirty database would instead be made into the same persistent database in which the inserted or updated record's primary table exists. However, these "dirty changes" would be made into a parallel table, in a separate, smaller-scoped transaction.

Queries and validations which rely on the dirty database today would be made against a view which encompasses a join between the primary table and this parallel table, such that a result would be found with a single query, rather than the two queries and client-side "join" it takes today (one query against the primary table, the other against the dirty database table).

An important consideration is whether the use of a joined table view and the need to push data over the network to the persistent database (vs. an embedded database) outweigh any performance benefits derived from eliminating the separate query and any logic currently used to join this data with data from the primary table.

A complication of this implementation (not unlike the current one) is that it will require a separate connection and a smaller transaction scope (i.e., not the application-level transaction scope used for business logic), to ensure dirty data is visible to other sessions before the current application-level transaction is committed.

Another consideration is the application of the joined table views. This will need additional DDL, presumably generated during conversion. When is it applied? Server startup? Just in time?