Database - Feature #5685

optimization: limit iteration of RecordBuffers in Commitable hooks by scope

09/27/2021 05:07 PM - 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 - 09/27/2021 05:20 PM - Eric Faulhaber

From previous profiling, I recall we spend too much time iterating through the BufferManager.openBuffers scoped list on calls from TransactionManager Commitable callbacks, especially for the commit and validate hooks. In large applications with deep call stacks, openBuffers contains a large number of RecordBuffer instances at outer scopes, most of which I think don't need to be processed during these hooks.

It seems to me we that for any given Commitable hook, we only need to process those buffers which are "in scope" up to and including the nearest enclosing external procedure scope. If we just iterated buffers from this scope to the current scope, we would handle all the buffers we need.

In most cases, a buffer is explicitly opened with RecordBuffer.openScope or RecordBuffer.openScopeAt. There may currently be exceptions to this rule, such as buffers passed to the external procedure or to a previously loaded procedure by handle. I don't recall if these are added to the openBuffers list in the scope which they are used, or if we just rely on the fact that we currently iterate ALL the openBuffers.

Can anyone think of use cases where a RecordBuffer may need to process Commitables callbacks, but it is not in the openBuffers list in a scope at or more deeply nested than the current external procedure scope? I need to get some test cases together to determine whether some additional work is needed in these cases, to be able to iterate only those buffers in the openBuffers list at the current external procedure scope or higher.

#2 - 09/28/2021 07:37 AM - Greg Shah

Anything passed as a named parameter must be kept in the list I think:

- buffer parameters
- table parameters
- table-handle parameters
- buffers referenced by dataset parameters
- buffers referenced by dataset-handle parameters

As you noted the other day, access to shared buffers is explicitly "imported", so that may naturally be kept in scope.

Questions:

- Is there any special behavior for a <buffer>:handle which is passed as a generic handle parameter and not as a table-handle or dataset-handle? This is a kind of "hidden" reference that can be arbitrarily passed downstream or exposed as a shared variable.
- I know we notify related buffers when something happens to a record that they may be holding. Are there scenarios where we need the scope notifications to process this?

#3 - 09/28/2021 04:24 PM - Eric Faulhaber

Greg Shah wrote:

> I know we notify related buffers when something happens to a record that they may be holding. Are there scenarios where we need the scope notifications to process this?

No, such notifications go through ChangeBroker, an unrelated mechanism. These already are pretty efficient, as they are targeted to listeners specifically registered to a specific record. They would not be affected by this proposed change.