Database - Bug #2261

improve server startup performance

03/17/2014 04:39 PM - Eric Faulhaber

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Eric Faulhaber	**% Done:**	50%
Category:		**Estimated time:**	0.00 hour
Target version:	Performance and Scalability Improvements		
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#1 - 03/17/2014 04:45 PM - Eric Faulhaber

Startup of the server for an application with a lot of permanent tables is very slow. While this probably is not a big problem for production use (where the server is restarted rarely), it is very annoying for development use and debugging.

I believe the main reason for the long startup time is the many JDBC metadata queries we make to gather index information (one per table). A potential fix is to make a smaller number of queries (a single one, if possible), store the information temporarily during server startup, then discard the information to recover memory once the persistence framework has been configured.

#2 - 03/17/2014 05:38 PM - Eric Faulhaber

I've done a little bit of research and trial and error testing. I've found that unfortunately, the JDBC DatabaseMetaData API for retrieving index information is not scalable in terms of querying metadata for multiple tables at once. The API requires a specific table name be passed; null and wildcards are not valid. So, the solution is not as simple as requesting a broader result set.

It seems now that there are several possible ways of dealing with this (aside from simply reducing the number of tables in use):

- defer the work of querying index info so that it is done lazily, on demand;
- multi-thread the startup, so that multiple metadata queries can be executed simultaneously;
- access the information directly from the database using system table queries.

None of these options is very attractive. Doing the work lazily would involve refactoring quite a bit of code. Multi-threading the startup may be the cleanest option, though the benefit will depend heavily on the number of CPUs and I/O throughput available. Accessing the information from system tables is IMO the least attractive option, in that it is complicated, dialect-specific, and fragile across database versions.

I will look a bit more into the impact/effort/risk of refactoring the code to implement the lazy option and the multi-threading option. I am avoiding the system table option for now.

#3 - 03/17/2014 07:24 PM - Eric Faulhaber

We have redundant index metadata queries coming from DMOIndex$DMOMetadata.collectIndexData and DirtyTempTableHelper.getIndexes. The latter needs to work with a copy, because it modifies the indexes to work with the dirty database manager, but we probably can reroute the call to collect the initial metadata to leverage the cache DMOIndex maintains of this information, then make copies of the P2JIndex objects that come back, and modify those copies. The copy work may still take some time, but we will avoid many round trips to the database, which has to provide a significant benefit to server startup time.

#4 - 03/18/2014 02:24 PM - Eric Faulhaber

- File ecf_upd20140318a.zip added

The attached update implements the plan in note 3 above. Empirical testing show it trims ~33% off the server startup time. It has not yet been regression tested. It also uses slightly less memory for the index data we cache.

#5 - 03/18/2014 04:03 PM - Eric Faulhaber

- File deleted (ecf_upd20140318a.zip)

#6 - 03/18/2014 04:04 PM - Eric Faulhaber

- File ecf_upd20140318a.zip added

Replaced with a version that does better error handling when retrieving index metadata.

#7 - 03/19/2014 10:43 PM - Eric Faulhaber

- % Done changed from 0 to 50

- Target version changed from Milestone 11 to Milestone 17

The 0318a update has passed regression testing and is committed to bzr rev. 10494.

The performance improvement is enough to make development use more tolerable, but further improvements wouldn't hurt. However, I don't consider this a critical requirement for M11, so I am changing the milestone to M17, which is about performance and scalability.

#8 - 11/16/2016 12:30 PM - Greg Shah

- Target version changed from Milestone 17 to Performance and Scalability Improvements

Files

ecf_upd20140318a.zip	54.1 KB	03/18/2014	Eric Faulhaber