Database - Bug #4931

possible ProgressiveResults performance improvement

09/29/2020 12:30 PM - Eric Faulhaber

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

Description

Related issues:

Related to Database - Bug #4917: eliminate redundant ORDER BY elements in mul...	**New**
Related to Database - Support #6707: evaluate the effectiveness of the curren...	**New**
Related to Database - Feature #6582: implement multi-table AdaptiveQuery	**WIP**

History

#1 - 09/29/2020 12:51 PM - Eric Faulhaber

AdaptiveQuery uses ProgressiveResults internally to fetch a progressively larger set of query results. The original idea was to get just 1 result at first, presumably at minimal cost. If the query loop was terminated, or some data in the table was changed, such that the query had to go from preselect to dynamic mode, we would minimize our cost. After that result was consumed, the query would fetch progressively larger sets.

A possible performance problem with this approach is that when we have a query which cannot be executed quickly, even with a limit of 1, we can have an expensive initial fetch, followed by additional, expensive fetches for the progressively larger batches. This can happen if, for instance, the query's results must be sorted in a way that does not match the index selected at query conversion, such that an efficient query plan cannot be chosen. This happens because the whole result set must be determined by the database, before the 1 record to be returned is found.

If we can determine when we have such a query (e.g., the sort phrase does not match the selected index, or we have a multi-table AdaptiveQuery generated from an optimized CompoundQuery), we may be better off fetching a much larger batch up front (possibly the whole result set), rather than going through the progressive steps. I think the progressive steps are still a good idea when we have a query for which we believe it will be easy for the database to pick a good, fast query plan. This is actually the common case for most single table AdaptiveQuery cases.

This idea is all theoretical at this point and needs some testing to be borne out. In practice, the cost of fetching batches after the initial one may already be mitigated by caching done by the database server. I'll write some test cases and see if there is something to be gained by changing the behavior of ProgressiveResults. The tricky part will be to determine when we have a query which is likely to be slow, but I have some starting ideas:

- multi-table queries whose sort phrase represents more than just the selected index of the first table (currently, this is limited to the optimized CompoundQuery case, as we don't convert multi-table queries directly to AdaptiveQuery);
- queries with a BY clause that does not match the selected index of the (first) table (actually, these might be converted to PreselectQuery instead of AdaptiveQuery already).

For the first one, we have to take into consideration any changes driven by #4917.

#2 - 09/29/2020 12:51 PM - Eric Faulhaber

- Related to Bug #4917: eliminate redundant ORDER BY elements in multi-table queries added

#3 - 08/25/2022 11:40 AM - Alexandru Lungu

- Related to Support #6707: evaluate the effectiveness of the current approach to ProgressiveResults added

#4 - 09/23/2022 10:52 AM - Eric Faulhaber

- Related to Feature #6582: implement multi-table AdaptiveQuery added