Database - Feature #7060

Consider using wildcard selection to reduce SQL size

01/23/2023 04:17 AM - Alexandru Lungu

Status:	Test	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Alexandru Lungu	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Database - Bug #6829: H2 forces re-parse of all prepared statement...	**Closed**

History

#1 - 01/23/2023 04:35 AM - Alexandru Lungu

There is initial work in #6829 on introducing wildcard selection (using * symbol). This can reduce the size of an SQL and boost the performance of string manipulation, SQL parsing, GC and caching, while also reducing memory consumption. The motivation comes from the large number of queries over tables with tens of columns, resulting in kilobytes of SQL strings.

The current effort is targeting only the H2 temporary database as we have a better understanding and control of the order of the selected fields using *. Not all queries can use * as some temp-tables have "computed/hidden columns": character indexed fields generate 2 columns, datetime fields generate an extra column, some extent field resolution may change the legacy table structure, etc. Therefore, we need some **quick** feedback on:

- How many SQL queries can actually use this wildcard technique in large customer applications? If there are few, we should find other better ways of reducing the selection field list size inside the SQL.
- Which is the performance/memory increase on using the wildcard? If this is negligible, then we can drop this effort.
- What are the causes for the queries which can't use a wildcard selection?

Only if we see a good performance from the wilcard selection and a low rate of usage: consider adding a HIDDEN keyword which can be used for the columns on a CREATE TABLE statement. Each time a wildcard selection is used, avoid returning the columns marked as HIDDEN. This way, we may significantly increase the wildcard usage rate on the back of some slim H2 internal changes.

#2 - 01/23/2023 04:35 AM - Alexandru Lungu

- Related to Bug #6829: H2 forces re-parse of all prepared statements when metadata is changed added

#3 - 01/23/2023 04:50 AM - Igor Skornyakov

Please note that this change may require extensive testing and changes in some non-obvious places.
For example in my recent changes for the recursive data-relations support I heavily rely on the fact the full list of fields with aliases is generated.
I think that we must have really good reasons based on extensive testing and profiling to make such a risky changes.

#4 - 03/16/2023 10:28 AM - Alexandru Lungu

- Assignee set to Alexandru Lungu

- % Done changed from 0 to 80

- Status changed from New to WIP

I committed 7026b/rev. 14506 for wildcard selection only for P2JH2 dialect which already supports INVISIBLE keyword for computed columns. Note that the wildcard is used only for a subset of cases (all fields are requested, H2 backend and there are no computed columns or they are marked with INVISIBLE). H2 is expanding the wildcard anyways in the query parser, but the goal here is to have smaller SQL strings which are cached faster.

This should be safe enough as it targets only FqlToSqlConverter.expandAlias, which expands all properties. Without wildcard (*), the aliases generated by FqlToSqlConverter.expandAlias are quite arbitrary (i.e. column1_5_ or other2_3_). The expandAlias generator is not used anywhere else.

> For example in my recent changes for the recursive data-relations support I heavily rely on the fact the full list of fields with aliases is generated.

Do you mean the column1_5_ or name2_3_ kind of aliases? Or do you explicitly generate some column alises and bypass expandAlias?
To use these results back in Java, I guess you can use ResultSet.getMetaData, so you retrieve the column names. This are retrieved even if you use *. However, I don't see any usage of the column names from the result-set metadata in FWD (except for cases where explicit SQLs are run, not generated by FqlToSqlConverter anyways).

> I think that we must have really good reasons based on extensive testing and profiling to make such a risky changes.

This is purely a performance optimization. I have a ~0.3% performance increase on a large POC just due to 7026b/rev. 14506. It can be extended to recover a bit more time (support extents, increase query cache size, etc.).
I didn't have any obvious regression in my tests or while navigating through some large applications. I've done some debugging and things look right.

#5 - 03/28/2023 07:16 AM - Alexandru Lungu

- Status changed from WIP to Review

- % Done changed from 80 to 100

#6 - 04/20/2023 07:12 AM - Alexandru Lungu

- Status changed from Review to Test

This was merged in trunk as rev. 14523 and can be closed.