

## Database - Feature #7320

### Extend H2 tester from Hotel GUI

05/02/2023 08:50 AM - Alexandru Lungu

|                        |                  |                        |           |
|------------------------|------------------|------------------------|-----------|
| <b>Status:</b>         | WIP              | <b>Start date:</b>     |           |
| <b>Priority:</b>       | Normal           | <b>Due date:</b>       |           |
| <b>Assignee:</b>       | Alexandru Donica | <b>% Done:</b>         | 60%       |
| <b>Category:</b>       |                  | <b>Estimated time:</b> | 0.00 hour |
| <b>Target version:</b> |                  | <b>version:</b>        |           |
| <b>billable:</b>       | No               |                        |           |
| <b>vendor_id:</b>      | GCD              |                        |           |
| <b>Description</b>     |                  |                        |           |

#### History

#1 - 05/02/2023 09:27 AM - Alexandru Lungu

- Assignee set to Ștefan Roman

- Subject changed from Extend H2 tested from Hotel GUI to Extend H2 tester from Hotel GUI

This task is related to the H2 tester from Hotel GUI, used to track performance of FWD-H2 in certain scenarios. We shall extend it to have a better overview of FWD-H2 performance.

#### Usage:

- Right now, the testing dialog is activated through CTRL-ALT-H. This is OK, as it allows somebody to use the tester in GUI mode.
- We need a CLI to activate this, so that we can skip the whole server/client overhead. This should be done last, but keep in mind that such thing is planned.
- Right now, one can only activate certain operations/test suites (CREATE, UPDATE, FIND, DELETE).
- We shall allow some complete logical scenarios (i.e. 5000 CREATES, 2000 FINDS, 1000 UPDATES, 2000 FINDS, 1 BATCH DELETE). Activating such scenario, fills the browse with the operations done: first row (5000 creates), second row (1000 updates) etc. Each such scenario is a new button. Maybe if we exceed 10 scenarios, we can use a selection list / tree widget.

#### Accuracy:

- Right now, the tester is victim to the warm/cold error threshold. That is, longer runs mean better and accurate results. Cold runs mean inconsistent slow results.
- We need each scenario executed 20 times. Extend the tester browse to include: cold run (the first), average (the average of all runs, except cold), best (the best run).
- Note that a scenario with 5 phases should generate a browse with exactly 5 rows. Each row contains the information on cold run, average and best. Basically, each row aggregates the times from each of the 20 runs.
- Compute some totals. While individual performance is insightful, we need to have a view on the bigger picture. Compute 3 totals in the end: the cold run, the average run, the best run.
- Use more columns / more operations if the times are low (< 200ms). We need to have an accurate final time.

#### Scenarios:

- Avoid using any random! This alters the consistency of the results.
- We certainly need more complex and realist scenarios to test:
  - CREATE / DROP: we need to evaluate the time to create and drop temp-tables at all. Create and drop ~10k dynamic tables. Use ~20 columns of different data types.
  - COMMIT / ROLLBACK: we need to evaluate the time of transaction commits. Create small/big transactions which are committed or rolledback. Use UNDO/NO-UNDO tables.
  - COPY-TEMP-TABLE: check performance of copying data from one temp-table to another. Use tables of ~20 columns with different data types. Copy around ~1k records.
  - BUFFER-COPY: check how fast is data copied from one buffer to another.
  - multi-table queries
    - PRESELECT: check how fast we can iterate a preselected query. Consider using WHERE and BY clauses.
    - COMPOUND: check how fast we can iterate a for-each query. Consider using EACH/FIRST/LAST clauses, WHERE and OUTER-JOIN.
    - CAN-FIND: nested CAN-FIND inside a WHERE clause
  - CAN-FIND: check how fast we solve stand-alone CAN-FIND (with/without WHERE, with/without FIRST).
  - REPOSITION: use reposition to move through a query several times.
  - ADAPTIVE: use FOR EACH queries. Consider using WHERE clause. Use SCROLLING/non-SCROLLING queries. Consider invalidating

- when iterating.
- FIND: use FIND queries. Consider using WHERE clause.

Each time you take WHERE into account, consider:

- no-WHERE clause
- recid look-up (WHERE recid(tt) = 1)
- unique index look-up (WHERE tt.pk = 1)
- index look-up (WHERE tt.name = 'abc')
- no index (WHERE tt.misc = 1)

Radu, Danut, please post whatever simple profiling test-cases you still have and fit this task.

## #2 - 05/05/2023 05:32 AM - Alexandru Lungu

Some of our recent work on FWD-H2 performance can be captured with tests like the following. Please make sure you include in your testing suite:

- [#7062](#): FIND FIRST tt WHERE tt.f1 = ?, where f1 is uniquely indexed, FIND FIRST tt WHERE tt.f1 = ? AND tt.f2 = ?, where (f1, f2) is uniquely indexed and FIND FIRST tt WHERE recid(tt) = ?
  - Try several such "unique look-ups", but ensure you use different tables each time, other-wise a specific FWD cache (ffCache which stands for fast-find) will kick in.
- [#7061](#): Any single-table AdaptiveQuery which is invalidated.
- [#7060](#): Any FOR EACH over tables with a lot of columns (~100).
  - Try several such loops, but ensure you use different tables / WHERE clauses each time, other-wise the statement will be prepared only once.
- [#7174](#): Use CAN-FIND on empty-tables (now is faster), CAN-FIND without WHERE on non-empty-tables (now is faster) and CAN-FIND tt WHERE tt.f1 = ?, where f1 is uniquely indexed.
  - You can do the same tests with CAN-FIND FIRST syntax.
- [#7108](#): Tables with lots of indexed character columns (CASE-SENSITIVE or CASE-INSENSITIVE). Do any kind of queries/finds on the indexed fields.
- [#6829](#): Intertwine table creation and queries/finds. Basically, execute the **same** query/find multiple times, but, between each execution, create a table on which you insert a row. Make the query/find as big as possible (WHERE / JOINS).

## #3 - 06/22/2023 03:13 AM - Alexandru Lungu

- Status changed from New to WIP

- % Done changed from 0 to 60

Stefan, please commit your extension to Hotel GUI.

AFAIK, you developed exclusively in 4GL for this task. Move your changes to the Hotel GUI project, convert with FWD and run. Make sure the tests are running in a decent time. If everything is right, commit to Hotel GUI / H2 tester.

#### #4 - 06/23/2023 07:31 AM - Ștefan Roman

The changes that I did to the H2 tester are :

- I removed the initial create;
- I removed the delete operation that was made before every other test in order to get the average, cold and best times;
- Added copy-temp-table test with 8 scenarios (append/replace/loose-copy over tables with none/one/two indexes);
- Added buffer-copy test with 3 scenarios (none/one/two indexes);
- Added 2 pre-made scenarios
  - 6 x create, 3 x find, 2 x update, 2 x find, 1 x delete;
  - 5 x ( 5 x create, 1 x delete);
- I removed find button and added queries instead, which opens a new window. This new windows creates 10k values on start, and the tests that it is containing are :
  - Find test with 18 variations;
  - For with 10 variations, including one 100 fields table;
  - Do/Repeat with 10 scenarios;
  - Define/Open query with 18 variations;

I finished working on this feature quite abrupt, so notice me in case you notice something that is not right.

I committed it, revision 268.

#### #5 - 08/22/2023 05:12 AM - Alexandru Lungu

- Assignee changed from Ștefan Roman to Alexandru Donica

There are lots of insightful tests generated for FWD-H2 right from the H2 tester. However, these are unusable in FWD as some of the scenarios are way slower in FWD:

- Generating the initial data takes a lot of time in FWD (~2min); we need to lower the population order. Doing a second population attempt is really slow.
- Some tests take > 5min (so I had to stop them). We need to carefully choose the parameters for which this test is run, so we have a decent measurable FWD time in the end.

First and fore-most we need:

- a run-time mechanism of reading the testing parameters. I am think of reading the testing parameters from a file at run-time (eventually allowing re-reading on a button press). This way we can tune the tests without re-conversion.
- a stop button to avoid restarting the application. Also, I am interested into a time-out feature. I imagine this as a combo-box. If this is checked, the tests will automatically time-out after some predefined static time.
- a uniform way of exporting the data. This way, we can compare H2 performance between FWD/FWD-H2 revisions **and** between 4GL and FWD.

Alex, please update your-self with the changes done in [#7320](#). They are already in the hotel\_gui project. Configure Hotel GUI in your 4GL VM and do some experimenting. Afterward, address the points above.