Database indexes and general performance

The test data consists of 29,552 rows in total.

(Clear query cache after every SELECT statement, “RESET QUERY CACHE”)

Case:

- Customer wants data between specific date ranges, or after x date
- Customer wants data specific to a given component
- Customer wants data according to a given test type (e.g.: integration, system, …)
- Customer wants data specific to trigger type (e.g.: push to GitHub, manual testing, …)

Solution:

- Non-clustered indexes on possible frequently used WHERE statements in the backend (componentName, testtrigger, startTime, testType)

Indexes:

CREATE INDEX IX_componentName on outerSuite (componentName) USING BTREE;

SELECT * FROM outerSuite WHERE componentName = "componentA";

Retrieved Rows:	
Performance without index:	0.022 seconds.
Performance with index:	0.001 seconds.

CREATE INDEX IX_startTime on outerSuite (startTime) using BTREE;

SELECT * FROM outerSuite WHERE (startTime Between "2020-09-13 00:00:00" AND "2020-12-30 00:00:00");

Retrieved Rows:	49
Performance without index:	0.028 seconds.
Performance with index:	0.002 seconds.

CREATE INDEX IX_testType on outerSuite (testType) using BTREE;

SELECT * FROM outerSuite where testType = "int";

Retrieved Rows:	9
Performance without index:	0.021 seconds.
Performance with index:	0.001 seconds.

CREATE INDEX IX_testtrigger on outerSuite (testtrigger) using BTREE;

SELECT * FROm outerSuite WHERE testtrigger = “testTriggerA”;

Retrieved Rows:	20
Performance without index:	0.021 seconds.
Performance with index:	0.003 seconds.

Index Name	**Index Type**	**Column**	**Table**
IX_componentName	B-TREE	componentName	outerSuite
IX_startTime	B-TREE	startTime	outerSuite
IX_testType	B-TREE	testType	outerSuite
IX_testtrigger	B-TREE	testtrigger	outerSuite

Slow Query Log:

To help with measuring future performance, we have introduced a slow query log that logs all queries that surpass a specified time limit to the slow query log. This can help us pinpoint areas of improvement in our database.

Below is a step-by-step guide in how we set the slow query log in our database.

1. Log in to the database server
2. Log in to MariaDB using the admin log in details
3. To enable the slow query log, type the following command:

   SET GLOBAL slow_query_log=1;

4. To specify the file name of the slow query log, use the following command:

   SET GLOBAL slow_query_log_file='mariadb-slow.log';

5. By default the output is saved to a file. However, you can also specify that the output is saved to a table. You can set either with one of the following commands:

   ```
   SET GLOBAL log_output='FILE';
   ```

   OR

   ```
   SET GLOBAL log_output='TABLE';
   ```

   NOTE: As the default is that the output is saved to a file, we have not used this command but it could be useful moving forward.

6. To set the time limit that is considered slow, we can use the following command:

   ```
   SET GLOBAL long_query_time=1;
   ```

   NOTE: As we are in development phase, we have defined a long query as a query taking 1 second, as above. This will need to be changed when large amounts of data are added to the db as performance will slow naturally. To change the slow query time, simply run the command again and replace 1 with the number of seconds you consider to be a long query.

7. We can also log queries that don't use indexes as these queries are highly likely to be able to be optimized with an index or a rewrite of the query. We do this with the following command:

   ```
   SET GLOBAL log_queries_not_using_indexes=ON;
   ```

These commands are the most relevant to our project and therefore are the commands we have used. However, there are more options available with slow query log in MariaDB. Please see https://mariadb.com/kb/en/slow-query-log-overview/ for more details.

Conclusion:

Performance increase at low amounts of data is almost neglectable, but as the DB grows it becomes important.

Performance increases should be remeasured once more valid test data is inserted into the DBMS.