Adding a New Benchmark to DuckDB

This is the second project for the Advanced Data Systems (ADS) class of 2024. The project is expected to take 4-5 weeks to complete.

Note: You will need a command-line platform for this. If you are using Windows, install the Ubuntu subsystem. If you need help or have questions about this, reach out to us.

Goals

The high-level goals of the project are the following:

1. Navigate the codebase of a widely used database system.
2. Extending an existing codebase with additional capabilities.
3. Get familiar with writing scripts to clean data, profile a software system, and analyze results.
4. Reason about the scalability of a complex codebase.

These goals will be achieved by the following more implementation-specific goals:

1. Become familiar with an in-process data analytics system, DuckDB.
2. Implement a standardized benchmark in DuckDB, Star Schema Benchmark (SSB).
3. Design and conduct experiments to analyze scalability of DuckDB query operators with respect to data size and number of threads using the SSB benchmark.

Tasks

Build & Get Familiar with DuckDB

Unlike Snowflake, DuckDB (https://duckdb.org/) isn't built for the cloud. It is an in-process / embedded data analysis platform. This means that by default it doesn't use a client-server setting. On the other hand, similar to Snowflake, DuckDB is considered an OLAP system as it adopts a columnar data layout and is optimized mainly for data analytics rather than small inserts and updates.

You can download DuckDB from its github repo (https://github.com/duckdb/duckdb) using:

```
$ git clone https://github.com/duckdb/duckdb.git
```

To build using the source code, follow the instructions at

- https://duckdb.org/docs/dev/building/build_instructions.html

We recommend building with extensions mentioned in this page to get the full capabilities of DuckDB. Also, add “tpch” extension to the build.

After you build DuckDB, run

```
$  ./build/release/duckdb
```

and get familiar with its functionality.

You can run SQL statements in memory after creating a table, create a database and run statements over it, run TPC-H benchmark queries, add data from csv files into a table, get timings of and profile the queries you run, set memory and thread limits, etc. Try out the different commands DuckDB prompt offers and see DuckDB documentation:

- D .help # will print DuckDB command list in the prompt
- https://duckdb.org/docs/ # has documentation for SQL examples, profiling, etc.
- https://github.com/duckdb/duckdb/tree/main # has readme files for various components

Notes:

- If you use csv files to load data to a tables using DuckDB prompt, follow the instructions at https://duckdb.org/docs/data/csv/overview.html instead of using *.import* command. *.import* can neither automatically detect csv delimiter, data types, etc., nor give decent error messages. It is also much slower.

Questions to get you started

(1) What are the options for performance analysis and profiling in DuckDB?
Does using mode detailed profiling options increase query runtime?

(2) What are the ways to evaluate standardized benchmarks in DuckDB?
What are their pros / cons in terms of implementation complexity and benchmark flexibility?
Hint: You can get inspiration from the TPC-H and Clickbench implementations under DuckDB.

Adding Star Schema Benchmark to DuckDB

In Project 1, you performed experiments using the standardized TPC-H benchmark. The Star Schema Benchmark (SSB) is a simplified version of TPC-H. You can find SSB specification at

- https://cs.umb.edu/~poneil/StarSchemaB.pdf

You can pick one of the ways of adding a benchmark to DuckDB based on your answer to question#2 above and implement SSB in DuckDB. We don’t expect you to deliver the most flexible and efficient benchmark implementation. As long as you can answer the questions under *Performance Experiments* (next section), it is enough.

Notes:

- You can find info on how to generate data, tables, and queries for SSB at various sources online. You are welcome to copy and paste SQL statements from these sources as long as you cite them in your report. For example, one such source is
  - https://clickhouse.com/docs/en/getting-started/example-datasets/star-schema
- Many sources will have a version of SSB with denormalized tables and queries where there is one flat table for all the data. Please use the normalized / original tables and queries in this project.
- You need to use dbgen to generate SSB table data. For this, use the dbgen code shared with you in the following repository:

  $ git clone https://github.itu.dk/pito/ads2024-ssb-dbgen.git

  The dbgen code found online doesn't generate the *datekey* column in DATE table in the format DuckDB expects.
- Keep in mind that the type names in DuckDB may be different than the ones you find in these sources. Thus, change them accordingly when creating tables.

Performance Experiments

Analyze the performance of SSB in DuckDB by answering the following questions for three SSB queries of your choice.

(1) How does the query performance change as you increase the number of threads running the query?
Which parts of the query (i.e., which query operators) get impacted the most?
Are there results expected? Why? Why not?
Perform this experiment using thread counts of 1, 4, and 8.
Keep the scaling factor constant as you change the thread count.
You are welcome to pick any scaling factor you want here, but for the sake of comparison picking a scaling factor that matches one of the points in question #2 below would be good.

(2) How does the query performance change as you increase the size of the database?
Which parts of the query (i.e., which query operators) get impacted the most?
Are there results expected? Why? Why not?
Furthermore, how does the size of the database increase as the scaling factor increases?
Perform this experiment using SSB scaling factors (SF) of 1, 10, and 100.
Keep the thread count the same across runs with different scaling factors.
You are welcome to pick any thread count you want here, but for the sake of comparison picking a number that matches one of the points in question #1 above would be good.

Hardware. Since DuckDB is a lightweight system that can run on various software and hardware platforms, you should be able to run all experiments on your local computers, but if you prefer to use a bigger server, you can use ITU's HPC cluster (hpc.itu.dk) or we can give you access to a DASYA lab server (both accessible

over ITU's VPN). Simply reach out to us (at pito@itu.dk). We have to implement a time-allocation mechanism on this server though as several people may be sharing it.

Memory requirements. If you get out-of-memory errors, first check if you can increase the memory limit set by default by DuckDB. If your hardware platform don't have enough memory for a particular scaling factor, you are welcome to use a smaller one, just mention this in your report.

Deliverables

The hand-in for the assignment includes:

1. Your code for the SSB implementation in DuckDB. Please provide a link to a GitHub code repository, either from ITU or the public GitHub.
2. Your report.

The report should be approximately 5 pages and should include the following sections:

- **Answers to questions** under "Questions to get you started".
- Description of your **SSB implementation** in DuckDB and why you picked this particular approach.
- **Experimental Methodology and Results.**
    - Describe your experimental design for performance analysis (hardware specification, SSB queries you picked to analyze, profiling option used, the number of repetitions of the runs, values for scaling factor and thread count). You can use the experimental setup descriptions from the papers we have been reading for this as an inspiration.
    - Present the results of your experiments. If you use graphs, ensure the x-axis, y-axis, and graph legends have clear titles with corresponding units and explanations.
- **Discussion of Results.** Interpret the results and discuss additional findings. In other words, answer the two questions under "Performance Experiments".
- **Conclusion.** Summarize your main findings and learnings.

Feel free to ask us if you have questions either in class (during Monday's exercise session), in the class discussion forum (so that others can also learn from it), or directly to Pınar Tözün (pito@itu.dk).