MDS
Winter Term 2024/25
miniHive, Milestone 4

In the fourth milestone, we optimize *miniHive*. The goal is to reduce the total amount of data stored in intermediary files within HDFS.

1 Evaluation Data

We use data from the TPC-H benchmark, describing customers and their orders. This is a popular benchmark where synthetic data can be generated based on a scale factor (SF).

Figure 1 describes the database schema and has been taken from the official benchmark description. We assume that all data adhere to this schema.

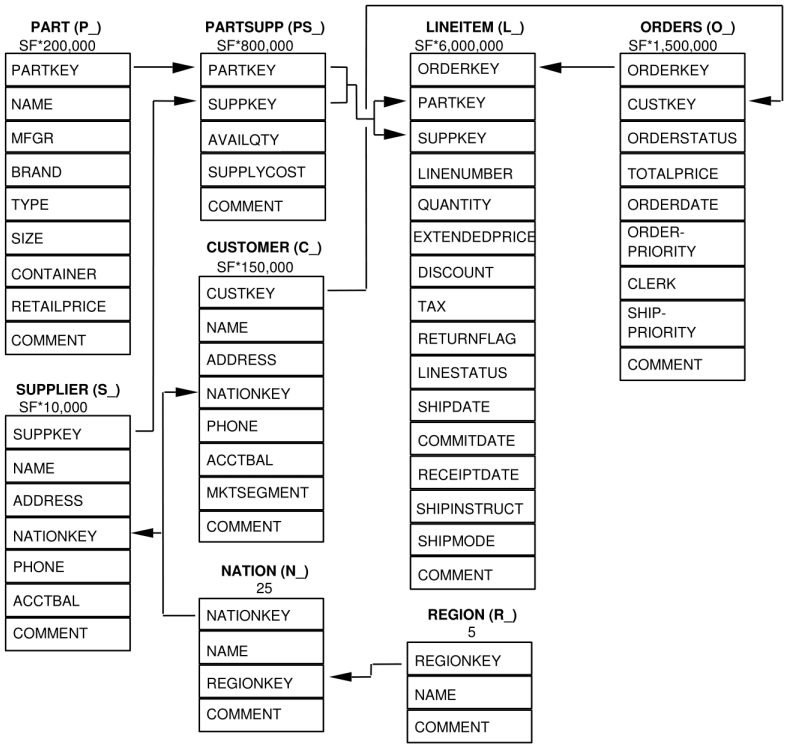

Figure 1: The TCP-H benchmark data [Source: `www.tpc.org`.]

The parentheses following each table name contain the prefix of the column names for that table. The arrows point in the direction of the one-to-many relationships between tables. The number/formula below each table name represents the cardinality (number of rows) of the table. Some are factored by SF, the scale factor.

2 Calling *miniHive*

For evaluating a single query, *miniHive* gets

- an optional flag telling miniHive to switch optimization on,
- the optional scale factor, allowing you to derive the *approximate* sizes of tables,
- the input files, already stored in HDFS or available as local files,
- the optional execution environment (LOCAL or HDFS), set to HDFS by default,
- the SQL query over TPC-H data to evaluate.

This is how we call the optimized *miniHive* for execution on local files:

```
python3 miniHive.py --O --SF 1 --env LOCAL \
"select distinct N_NAME from NATION"
```

If called in `LOCAL` mode, this outputs an approximation of the HDFS storage costs (more on this later). **Make sure your submitted version does not write any other information to standard out, as this will confuse the test scripts.**

3 Material Available on GitHub

In our GitHub repository at `https://github.com/miniHive/assignment/tree/master/milestone4` you find additional material:

- A sample dataset that was generated with scale factor 0.01 (`data_0_01.zip`).
- The file `miniHive.py`. You may alter this file.
- The file `costcouter.py`. You must not alter this file.
- A list of SQL test queries in `miniHive.q`.

4 What to Submit in Praktomat

- All files to run miniHive: `sql2ra.py`, `raopt.py`, `ra2mr.py` and `miniHive.py` (upload individually or as ZIP file).
- A README file (`README.md`) with a 1-paragraph description of the optimization(s) that you have implemented. The README must have this format:

```
# Author: <your lastname>, <your firstname>

# My approach:
<1-paragraph description of your optimizations>
```

5 Earning Points

Your submission will be tested by a fully automated Python script. **Make sure you do not break the functionality of the script.**

The script parses the local temporary files, assuming that they have the format familiar from Milestone 3, namely "`some-key json-encoded-value`". **Do not change this format. Do not tinker with the JSON encoding of the value**.

The script then determines the total number of characters encoding MapReduce values in all output and intermediary files (the input files are the same for everybody, so we ignore them). **Do not tinker with the intermediary files.**

The README file must be comprehensible. Your implementation must run in the unoptimized and the optimized mode.

You must **implement at least one genuine optimization**. We recommend to implement chain folding, but projection pushing is also a good strategy, as is cost-based join reordering. You may also implement your own optimizations.

Optimizing the format of temporary files (e.g., using compression) will earn zero points.

The automated scripts must be able to run all test queries from `miniHive.q` with your implementation on HDFS and on LOCAL files with a secret scale factor, and measure the storage costs for temporary files.

In particular, your implementation has to fulfill the following criteria:

- For all of the queries in `miniHive.q`, your costs must be lower or equal than those of your (unoptimized) implementation of Milestone 3 (all implementations of Milestone 3 should have the same costs). This requirement ensures that your optimization does not actually make things worse.
- For at least half of the queries in `miniHive.q`, your costs must be lower than those of the (unoptimized) implementation of Milestone 3. That means that you can't just resubmit the same code as in Milestone 3.
- For at least three queries in `miniHive.q`, your optimization must be able to reduce the costs by one third.

Remarks: A successful submission that (1) passes all public tests from Milestone 3 on Praktomat (in unoptimized and optimized mode), (2) correctly processes the same queries on Hadoop MapReduce inside the miniHive Docker container (`--env HDFS`), (3) meets the optimization criteria stated above, and (4) passes the plagiarism check, earns 10 points.

The deadline for submitting Milestone 4 is December 16, 2024, 12 noon.
