Join Algorithms: Nested Loop Join vs Hash Join

The goal of this project is to Implement nested-loop join and hash join, and benchmark their performance using a standard dataset. We don't use SQL for this project; we implement the algorithm using old-fashioned language with familiar control logic (e.g., if, for, functions, and so on).

- **Submission Deadline:** November 27, 2021 (Saturday).
    - This due date is set so you still have enough time for the final demo of our course project (on December 2). If you are busy due to other commitments, please start and submit early. The time management is up to you.
    - The submission time is based on the latest commit of your GitHub repository. To this end, please don't make additional commits after the due date, which will complicate our grading. If you still want to commit something (for your own sake), you can simply create a branch and continue to work in that branch. Simply don't commit to the master/main branch.
- **How to submit:** Create a GitHub repository (either in your account or our course project organization) and send us the link to the repository via email.
    - If you wish to use your own repository, you may need to make it public so that we have access to it.
    - Email addresses: yongjoo@illinois.edu; dm42@illinois.edu;
- **Credits:** Upon successful completion of this project, you receive up to 3% extra credit in addition to the credits you would have earned normally.
    - This extra credit goes toward the total score you earn for this course. 3% is set to recover from missing one SQL problem in Midterm 1, not any more than that (that is, Midterm 1 is 15% while a SQL problem in the midterm is 20% of it; thus, $0.15 \times 0.2 = 0.03 = 3\%$).

Project Specification

At a high level, your program takes (1) the names of two input CSV files (containing data for relations/tables), (2) the join method, and (3) the file name for an output CSV file; then, your program writes the join result to the output CSV file (overwrite if the file already exists), and

prints out (to the standard output) the time it took (in microseconds) for producing the output file.

For example:

```
java -cp "*.jar" edu.illinois.netid.Main input1.csv left_col_name1
input2.csv join_col_name2 NESTED_LOOP output.csv
```

will read data from input1.csv and input2.csv, join them using the NESTED_LOOP algorithm (with the join condition of join_col_name1 = join_col_name2), and write the joined data to output.csv. Also, the program prints (to standard output) the time it took for generating the output file.

> Note: in the above example, the parts starting from java up to "edu.illinois.netid.Main" is due to the way Java works. Depending on your choice of programming language, it may require a different format for running your program. Please specify this in your README.md file.

Regarding the join method, there are two options (case insensitive): NESTED_LOOP and HASH. To understand how they work, please attend the class or read textbooks. For both, you only need to implement INNER JOIN (not LEFT/RIGHT/FULL OUTER JOINS).

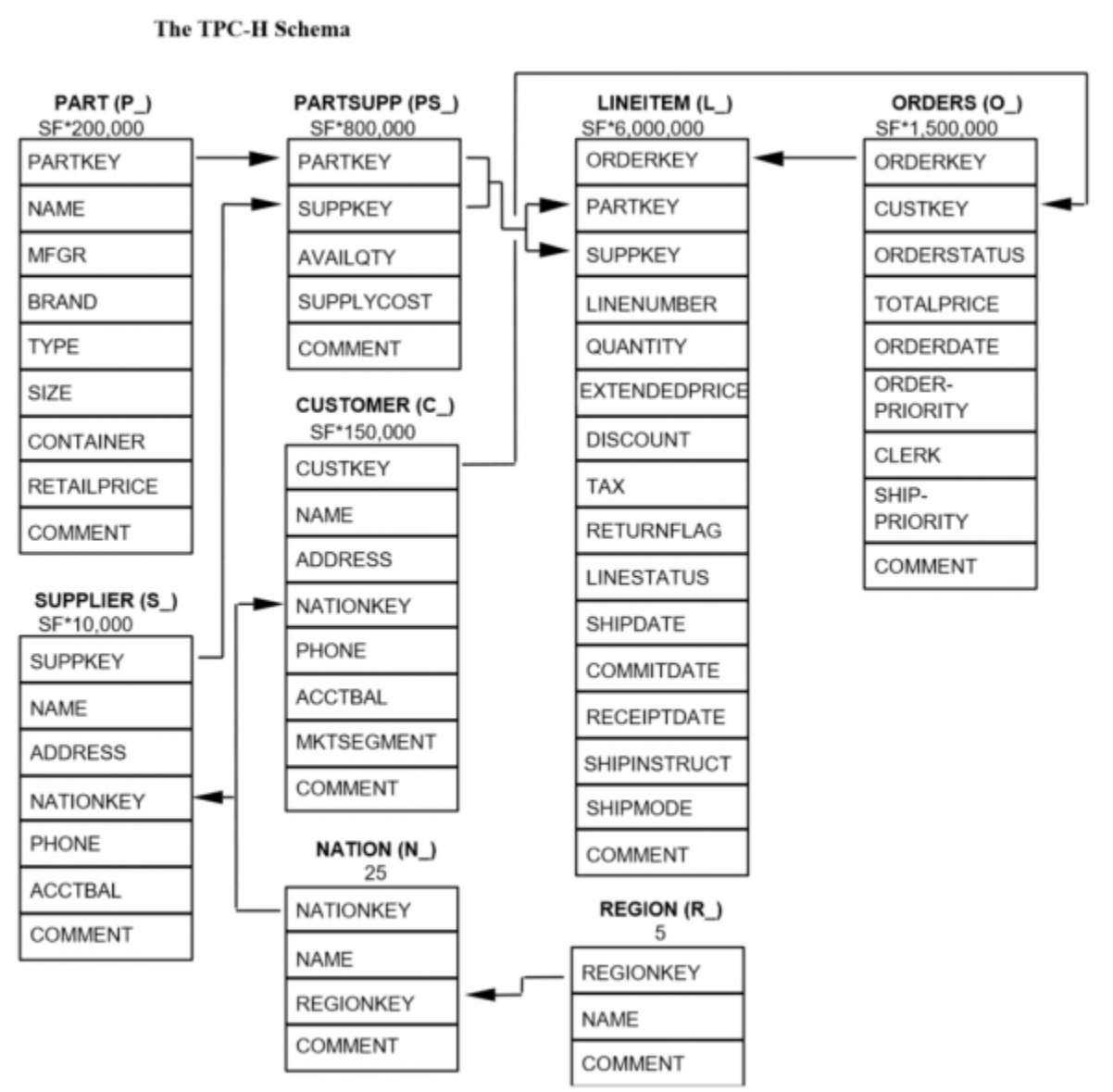
> Note: For both NESTED_LOOP and HASH joins, please assume that we have large-enough memory. That is, for NESTED_LOOP, we can read the entire relation (for the outer relation) and keep in memory, instead of reading it block by block. Likewise, for HASH, we can assume that a has table can be built for the entire relation.

The input CSV files are assumed to have a header row (i.e., column names) on the first line. You can assume that the input arguments do not include any special characters (e.g., single/double quotes, spaces) that would require more sophisticated logic for properly reading them.

Sample Input Files

For this project, we use a standard benchmark dataset called TPC-H. You can download the file from this link: https://drive.google.com/file/d/1TiVHfa0Pk4wD92wdh9gzVD3wtMUe0SBm/view?usp=sharing

The dataset includes multiple tables (e.g., lineitem, orders, partsupp, and so on). See the below diagram to understand table names, their attribute names, and the relationships. Please see the next section (Deliverables) to know how to use this dataset.



Deliverables

Send a link to your repository. The repository must include the following:

1. README.md: This file must include the following information:
    a. How to run your program for the sample CSV files (nation and region)
    b. Benchmark result
        i. Compare the performance (i.e., time) of nested loop join and hash join using two tables LINEITEM and ORDERS.
        ii. Include the results of 10 runs and the average of them, for each of nested loop join and hash join.
2. Sample CSV files
    a. Only two files: nation.csv and region.csv
3. Other files necessary for your program

Your instruction in README.md must state how to run your program in the terminal, without using IDEs such as IntelliJ or VS Code.

Additional Requirements:

1. For implementation, you must use a compiled language (e.g., Java, Scala, C, Rust, Go), not interpreted language (e.g., Python, Ruby, PHP).
2. The README.md file you include in your GitHub repository must be self-explanatory to install all dependencies and to run your program with sample files.
3. You may use standard or third-party libraries for almost all operations (e.g., parsing CSV files, sorting, reading from and writing to files, etc.) as long as they are not related to the core algorithms (i.e., nested loop join and hash join).

Questions?

Please use Campuswire for clarifying questions.