DB2

Forum: https://forum-db.informatik.uni-tuebingen.de/c/ss20-db2

Assignment 1 (28.04.2020)

Submission: Tuesday, 05.05.2020, 10:00 AM

Relevant videos: up to `DB2 - Chapter 02 - Video #08`.

https://tinyurl.com/DB2-2020

1. [0 Points] **Introduction**

   (a) **Before we grade your** team, you have to complete a few administrative tasks first. In your team's GitHub Classroom repository, there exists a file called `README.md`. Add each team member's name, surname, matrikel number, subject, field of study, e-mail as well as forum username to the incomplete table and commit+push the changes.
   Completing this task is a **requirement** for your team to be graded in the first place. Please make sure the file always exists with the correct information present.
   Note: Because of the high demand/low supply of exercise slots, please note that **not completing this task until the submission date (05.05.2020)** will prompt us to remove anyone not present in the `README.md` from the exercises, freeing a slot for students on the waiting list.

   (b) All of your submissions will be placed inside the `assignments/` directory of your team's GitHub Classroom repository. Each submission requires its own subdirectory called `solution<number>`, where number is the current assignment number.
   For example, the submission of this assignment will be located in the `assignments/solution01/` directory.

   (c) In general, the only accepted file formats are plain text files (.txt) and source files (.sql, .py, ...). Other formats may not be graded, unless stated otherwise.

   (d) Your submitted code has to work out of the box. If particular preparatory steps have to be taken to run your queries, document these steps properly.

   (e) Lastly, the usual rules for plagiarism and other academic integrity apply.

   (f) For any further questions about this lecture, assignment and other related topics, visit the forum at

   https://forum-db.informatik.uni-tuebingen.de/c/ss20-db2.

2. [20 Points] **FizzBuzz**
In the lectures, we introduced `generate_series(...)` for both *PostgreSQL* and *MonetDB*. Solve the following tasks in (1) *PostgreSQL* and (2) *MonetDB*.

   (a) Write a simple *SQL* query which produces a table with a single column of type `TEXT`.
   Use `generate_series(...)` to generate numbers $i \in \{1,2,\ldots,100\}$. The resulting table should contain each i (as a string). However, note the two following exceptions:

   - If i is divisible by 3, the table should contain `'FIZZ'` instead of i.
   - If i is divisble by 5, the table should contain `'BUZZ'` instead of i.
   - Lastly, if i is divisible by both 3 and 5, the table should contain `'FIZZBUZZ'` instead of i.

   Order the output so that you obtain a result as shown here:

'1'
'2'
'FIZZ'
⋮
'14'
'FIZZBUZZ'
'16'
⋮
'98'
'FIZZ'
'BUZZ'

   Note:

   - Consider to use `CASE WHEN`[1]-expressions, which are similar to if/else statements in other programming languages.
   - Do **not** use user-defined functions or common table expressions as part of your solution.

   (b) Create a table `fizzbuzz(v text)` then populate it with the result of the query of (a).

   (c) Will the result of the following query **always** return the exact same result as the query in (a)?

```
SELECT * FROM fizzbuzz;
```

   Explain briefly.

   (d) Table `fizzbuzz` is now persistently stored on your disk as a regular file. Name the path of this file on your system and describe the way you found it briefly.
   Consider the system catalogs table `pg_class`[2] in *PostgreSQL* and the *MAL* function `bat.info()` in *MonetDB*.

3. [10 Points] **Memory Scan**
For this assignment, you are given the file `transfer.c` which has been introduced in the lecture. This C program measures the time needed to access the same chunk of memory (8 MB) over and over until the total amount of memory used equals 4 GB.
Now, lower the size of the chunk of memory (see macro `SCANSIZE`). Each time you do, determine the time your program needs to complete and write it down.

   Explain how you can use these experiments to determine the size of your Level 1, Level 2 (and Level 3, if any) data caches of your CPU. Can you determine the cache level sizes on your system? What cache level sizes do your measurements suggest for your particular CPU?

[1] https://www.postgresql.org/docs/12/functions-conditional.html
[2] https://www.postgresql.org/docs/current/static/catalog-pg-class.html