T-202-GAG1: Exercise 11

Readings

Ramakrishnan & Gehrke: Chapter 8, 9, 12.1-12.3 and 21

The following links to Postgres documentation will also be useful:
https://www.postgresql.org/docs/9.5/static/sql-createrole.html
https://www.postgresql.org/docs/9.5/static/sql-grant.html
https://www.postgresql.org/docs/9.5/static/sql-revoke.html

Preparation

For this assignment, it is important to first listen to Module 09 and of value to have listened to the guest lecture on security in module 11. This should not take much more than 90min. of your time.

The Assignment

For this assignment, you can work groups but it is not necessary. Do as much of the following as you can in the span of 90 minutes.

This assignment is created with psql (command prompt) usage in mind. If you use some other tool you need to try and avoid outputting results when filling the database, as otherwise your results will be skewed. Also, do not commit after each insert, as that will skew your results too. The FILL.sql script avoids both those pitfalls.

If your computer has known issues with FILL scripts, or problems using psql, please arrange to work with someone with a computer that does not have such issues.

Part I: Create the Databases

First you need to download and extract the DB.sql file from Canvas in a folder with a known path.

Note: You need to modify the path in the 2_FILL.sql file accordingly.

Start by creating two databases *E11_nokey* and *E11_key*. Then, using the \c command to connect to the *E11_nokey* DB and use the \i command to run the **1_CREATE_NOKEY.sql**, which will create the database schema from P2 but without any PRIMARY KEY and FOREIGN KEY constraints.
Then populate the schema with the commands in **2_FILL.sql**. Write down how long it takes, so that you can compare the insertions with and without the overhead of index maintenance.

Next you should use the \c command to connect to the *E11_key* database and there you run the \i command on the **1_CREATE_KEY.sql** that will create the same database structure but this time with PRIMARY and FOREIGN KEY constraints.
Then repeat the steps to populate the database by running the **2_FILL.sql** again, making sure you write down how long it takes. You task is to write a report with

your measurements, as well as a explanation for why there is a difference in the measurements adding the data to *E11_nokey* and *E11_key*.

Part II: Security, Roles and Privileges

In this part your task is to create a role called ***e11_readonly*** with the password e11forever. This new user should be able to login to your Postgres server, access both the *e11_key* and *e11_nokey* databases but only be able to read from the data tables in both.

Use the \o e11part2.txt to redirect the psql command line output to file before running commands to show that you can read the data tables but not alter nor write to them (you do this by running some well-chosen SELECT and INSERT commands.
When done, run the \o command without any parameter to get the output back to the console screen.

Part III: Performance of Basic Queries

As ***e11_readonly,*** run the 3_Q1.sql file on both databases and write down how long it takes to execute each of the two queries, along with how many buffers were read.

- What could explain the difference in execution time and buffers read between Query I and Query II in each of the two databases?

Make sure to save to file the output of the ***explain plan*** output for both queries, for each to the two databases as you will need to include them in your report.

Part IV: Advanced Queries and Query Tuning

If you still have time, try running both variants of the **4_Qfile2.sql** and **5_Qfile3.sql**. As you man notice, the nested sub-select version of the queries run for a very long time. So, you may want to stop them (ctrl-c) and switch to the old P2 database we used in Project 2 (with much less data).

Then try creating an index to speed up the execution.

- Which attribute(s) should be included in the key and in which order, for best results. You can try some combinations
- How long does the execution take using the index?
- Do you see the index used in the query plans?
- Why are the queries written as a join so much faster than the nested subqueries? Can we see the difference in the query execution plan?

Deliverables

Submit a PDF file containing the outcomes of your experiments (including hardware information) and your thoughts about the results of the experiments (in particular the answers to the questions in bullets).

In the (short) report, section headings for each part are useful, as are tables.

Additional Exercises

Exercises: 8.3, 8.7, 8.9, 8.11.