Mark as done

Opened: Tuesday, 3 September 2024, 12:00 AM
Due: Wednesday, 11 September 2024, 11:59 PM

This assignment is an individual assignment.

You can submit either a text file, or a .odt/.docx/.pdf file. For each case, submit
* a query
* the chosen plan got by using **explain analyze**
* your reasoning behind constructing the query (briefly)
* any other information that is asked for in that part, such as an explanation for what you observed.

Before you start on this assignment, set up the large database and try out the sample queries as explained on Moodle.

1. Create a query where PostgreSQL uses bitmap index scan on relation takes. Explain why PostgreSQL may have chosen this plan.
2. Create an aggregation query where PostgreSQL uses sort-based aggregation. Explain why PostgreSQL may have chosen this plan
3. Create a query where PostgreSQL chooses a (plain) index nested loops join
   (NOTE: PostgreSQL uses nested loops join even for indexed nested loops join. The nested loops operator has 2 children. The first child is
   the outer input, and it may have an index scan or anything else, that is irrelevant. The second child must have an index scan or bitmap index
   scan, using an attribute from the first child.)
4. Create an index as below, and see the time taken to create the index:
   create index i1 on takes(id, semester, year);
   Similarly see how long it takes to drop the above index using:
   drop index i1;
5. Create a table takes2 with the same schema as takes but no primary keys or foreign keys. Find how long it takes the execute the query
   insert into takes2 select * from takes
   Also, find the query plan for the above insert statement.
6. Next drop the table takes2 (and its rows, as a result), and create it again, but this time with a primary key. Run the insert again and measure
   how long it takes to run. Give its query plan, and explain why the time taken is different this time
7. PostgreSQL does not create indexes on foreign key columns by default. See the effect of having/not having such an index by measuring the execution time of a deletion query. To do so execute the following:
   1. **begin;**
      1. The above is important since you want to roll back the deletion later, to avoid reloading the database!
   2. Now run **explain analyze delete from course where course_id = '400';** and find the time it takes. Record this time.
      Note that PostgreSQL does not explain how it checks for foreign key constraint violation.
   3. Finally run **rollback;** to restore the database state
   4. Next add indices on all foreign keys; for our purpose these will do: (1) all foreign keys that reference course_id directly and (b) on foreign keys that reference these relations (transitively!). For your convenience, these are listed below.
      1. section (course_id), prereq (course_id), prereq(prereq_id)
      2. teaches(course_id, sec_id, semester, year)
      3. takes(course_id, sec_id, semester, year)
   5. And now measure the time it takes for the same delete as above, using the same 3 steps you used earlier.
   6. Submit the timings along with an explanation of the results you observe.
8. Consider the following nested subquery with an exists clause:
   *select count(*) from course c where exists (select * from takes t where t.course_id = c.course_id)*
   What is the plan is chosen by PostgreSQL. Report the plan and actual execution costs. Explain what is happening.

9. As above, but with replace the *exists* clause in the query by a *not exists* clause.
10. As above, but with the query
    select count(*) from course c where (select count(*) from takes t where t.course_id = c.course_id) < 4

Add submission

Submission status

Submission status	No submissions have been made yet
Grading status	Not graded
Time remaining	4 days 2 hours remaining
Last modified	-
Submission comments	▶ Comments (0)

Jump to...

‹ Previous Activity

✉ Contact site support

You are logged in as Soumik Dutta (Log out)

Data retention summary

Get the mobile app

Powered by Edwiser RemUI