Query Evaluation Plans

This assignment can be done in groups of two people. (One person should submit the actual assignment, with file name as rollno1_rollno2.pdf (or whatever extension). The other one should submit an empty file called submitted_by_rollno2.)

Late submission penalty: We will accept late submissions till 10 PM, with 5% penalty for each hour (rounded upwards to the nearest integer hour) that it is late.

1. Load the university schema (from this file) and populate it with a large dataset from this file. If you need to drop tables, use this file.
    1. **Do not use pgadmin3 for loading data as it may hang.**
    2. open terminal and start psql with appropriate port number and database name (refer 1st lab on PostgreSQL setup)
    3. From within psql, run the command \i <filename>
        1. e.g.just type
           \i DDL.sql OR
           \i largeRelationsInsertFile.sql
           Run the command from the same directory where you saved the file.
    4. You will see "INSERT 0 1" being printed for every insert. It will take some time but will terminate successfully. You may see an unchanging screen even though the system is continuously printing the above line. Hit enter to see if prints are ongoing.
2. Run each of the following queries to find the time taken, and use the explain feature to find the plan used for each of the queries. By studying the plans, try to figure out why each query either ran fast or ran slowly. No need to submit anything for this part of the assignment, just see what plans are generated).
    1. select * from takes natural join student;
    2. select * from takes natural join student where ID = '1234';
    3. select ID, count(*) from takes group by ID;
    4. select * from student, instructor where student.id = instructor.id and student.id = '1234'
    5. select * from student, instructor where upper(student.id) = upper(instructor.id) and student.id = '1234'
    6. select * from student where exists(select * from instructor where instructor.name = student.name)
    7. select * from student where not exists(select * from instructor where instructor.name = student.name)
    8. select * from student where tot_cred = (select sum(credits) from course where course.dept_name = student.dept_name)

1. This part requires submissions; **for each case, submit a query showing the required plans. You can submit either a text file, or a a .odt/.docx/.pdf file containing both the query and the plan**.
   NOTE: PostgreSQL uses a technique called **bitmap-index-scan**, which is used for secondary index access to multiple tuples. It scans the secondary index, and creates a bitmap with 1 bit per page in the file, and sets the page bit to 1 if the index shows a relevant tuple is in that page. After finishing the index scan the bitmap is complete, and pages are now accessed in physical order, skipping pages whose bit is 0. For each retrieved page, all tuples are scanned, and the original selection condition has to be rechecked, since the page may contain tuples that did not satisfy the original condition (as few as 1 tuple may satisfy the condition). Bitmap index scans are useful only when multiple tuples are retrieved using a secondary index.
    1. Create a selection query whose chosen plan uses a bit-map index scan. You can create indices on appropriate relation attributes to create such a case.
    2. Create a query where PostgreSQL chooses a (plain) index nested loops join.
    3. Create a query where PostgreSQL chooses a merge join (hint: use an order by clause)
    4. Add a LIMIT 10 ROWS clause at the end of the previous query, and see what algorithm method is used. (LIMIT *n* ensures only *n* rows are output.) Explain what happened, if the join algorithm changes; if the plan does not change, create a different query where the join algorithm changes as a result of adding the LIMIT clause.
    5. Create a nested subquery with an exists clause where the subquery uses the relation takes, and see what plan is chosen by PostgreSQL. Explain what is happening.
    6. As above, but with a not exists clause.