German International University in Cairo
Informatics and Computer Science
Dr. Eng. Amal Yassien
Ahmed Ramadan
Nourhan Ahmed

Database Programming, Winter 2025
Project Description 2

Release Date: 16.11.2025
Deadline: 09.12.2025 @ 23:59
Teams: 2 or 3 members

Project Objective

Imagine waiting for 30 minutes for your purchase on Amazon to be complete, or imagine waiting for 1 hour for your post to be shared on Facebook, or your story to be uploaded on Instagram! Database operations are needed for all these daily life actions. Consequently, database engines always aim to optimize their query execution performance to avoid delays so that users can run their application smoothly. Database administrators resort to optimizing the most frequently performed queries on their engine by creating indices. However, they think very wisely about the necessity of the adding index and the type of index to be used, as an index can add an access overhead. Therefore, you will work in teams of 2 or 3 students in this project is to help a database administrator optimize a set of queries. What is different from project 1 is that now the database administrator would make informed decision based on the how the server performs while executing the query by inspecting the server physical plan and changing server configurations accordingly. Within project 1, the database administrator job was to optimize the queries given the default server configuration settings. Keep in mind that changing server configuration setting repeatedly on live applications can severly impede the server performance. Thus, it is nice through project 1 and project 2 to identify when is that measure is necessary to make. For each given query, you should

- Identify the attribute to create the index on in order to enhance the query's performance
- Determine the most expedient type of index to be created
- List the execution/physical plan of each query along with planing and execution timings before and after creating the index.

Needed Software

The following software is needed for this project:

- pgAdmin
- PostgreSQL server

Note: the vm used in the project 1 would be sufficient for implementing the project.

Note: You can also install postgres server natively on the machine.

Note: if you choose to use the vm or the native postgres version, you need to use that setup for the whole project, i.e. for all 23 queries.

[What you will do in the project] Query Physical Plan Optimization Steps

In order to be able to monitor the performance of your queries' physical plan before and after the index, you need to follow the below steps carefully:

a) Create a new database in pgAdmin

b) Right-click on the database you created in the left panel in pgAdmin, and open the query tool

c) **(Virtualbox Users)** Open cms from the VM using firefox (required) and download Project 1 Files.zip and extract it. **(Linux - Mac - Windows Users using native postgres server)** Open cms from your browser and download Project 2 Files.zip and extract it.

d) In Tilix/Terminal/SQL Shell, write the below command to load the database table and data into the database you just created in step (a).

   ```
   $ psql -f [full-path-to-sql-file] [databasename]
   ```

   replace [full-path-to-sql-file] with the full path to database.sql file (within the extracted Project 2 Files.zip) and [databasename] with the name of the database you created in step (a)

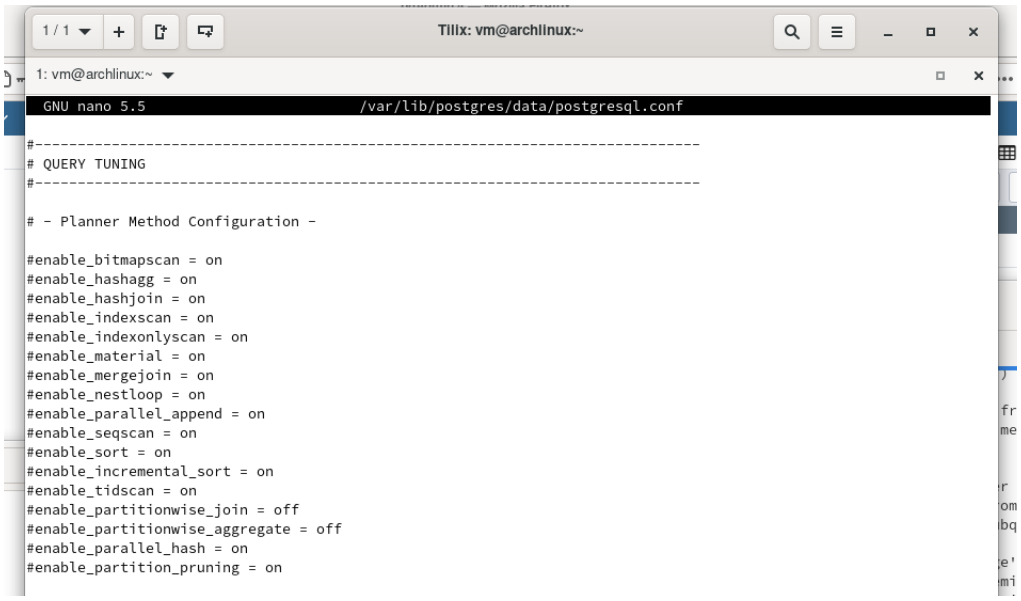
e) For each query provided in Pro2.txt file, do the following:

   1. Run the following command to get the physical plan of the query execution

      ```
      explain analyze [individual-query]
      ```

      replace [individual-query] with the current query you are working on

   2. Take a screenshot of the output and report on the planning time, execution time and the physical plan adopted (Plan Before Index). If the plan is a large one, then you can save it in a txt file from pgAdmin console.
   3. Determine the parts of the plan that had the highest cost, slowest runtime, and largest number of rows for the physical plan presented in the output. Using these parameters (but not only that), you can identify operations hogging up the query performance which can lead to identifying candidate attributes to create an index on.
   4. Determine the most expedient type of index to be created to optimize this query. You can rely on (1) parameters extracted in the last step, (2) the brainstorming and reasoning exercise we follow in labs, and/or (3) any course material or online resource. Also, you can use any index type supported by postgres.
   5. Create your chosen index in the query tool of pgAdmin. You can (but don't have to) create more than one index per query, if you see that this is necessary, but this needs to be mentioned in the report.
   6. Run the explain analyze command again (on the same individual query you are currently working on ;))
   7. Take a screenshot of the output and report on the planning time, execution time and the physical plan adopted (Plan After Index)
   8. Determine the parts of the plan that had the highest cost, slowest runtime, and largest number of rows for the physical plan presented in the output
   9. Compare the two physical plans (before and after index ones) and provide a reason for the observed behavior.
   10. In case the index you created was not used by postgres, then modify the postgres config file by turning off the technique used by the planner, restart postgres, and rerun the explain analyze command. For more, open the postgres config file and go the query tuning section. It should look similar to the below.



```
GNU nano 5.5                    /var/lib/postgres/data/postgresql.conf

#------------------------------------------------------------------------------
# QUERY TUNING
#------------------------------------------------------------------------------

# - Planner Method Configuration -

#enable_bitmapscan = on
#enable_hashagg = on
#enable_hashjoin = on
#enable_indexscan = on
#enable_indexonlyscan = on
#enable_material = on
#enable_mergejoin = on
#enable_nestloop = on
#enable_parallel_append = on
#enable_seqscan = on
#enable_sort = on
#enable_incremental_sort = on
#enable_tidscan = on
#enable_partitionwise_join = off
#enable_partitionwise_aggregate = off
#enable_parallel_hash = on
#enable_partition_pruning = on
```

10-1 For convenience, you can turn off that technique via the query tool of pgAdmin. It should be something like the below.

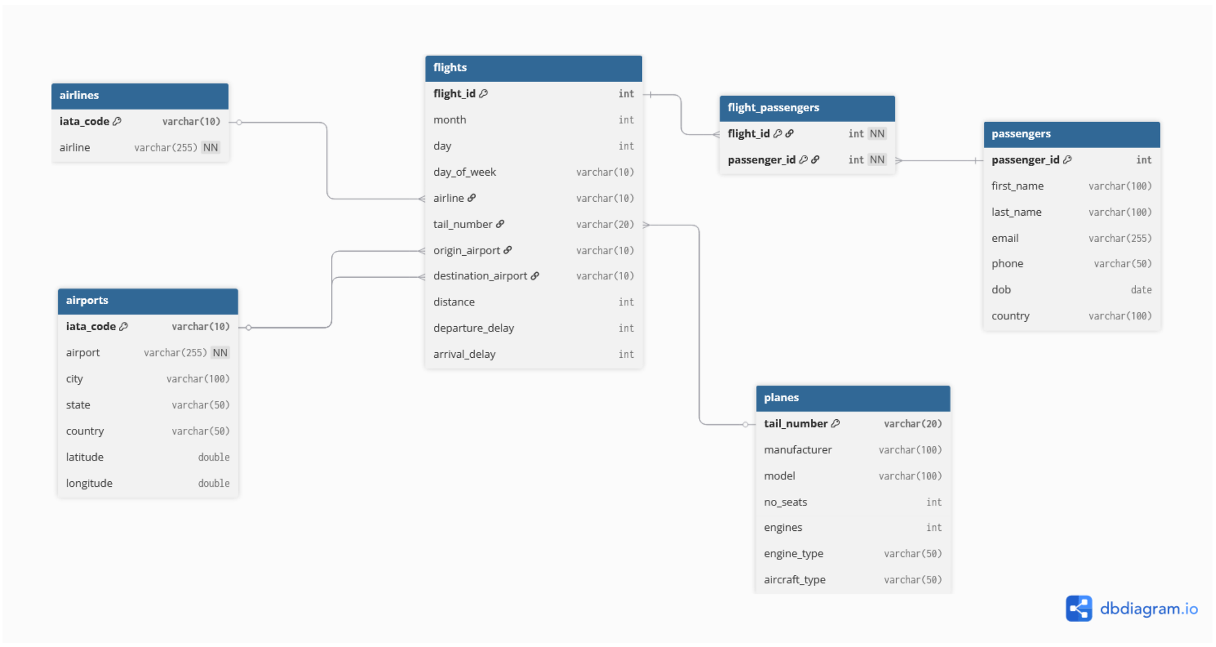
```
set [technique_name] = off;
```

replace [technique_name] with the name of the technique you want to disable. For example, set enable_seqscan = off;

11. Drop the index you created from the database
12. Undo any changes performed on the config file in step 10 or turn on the technique that you turned off in step 10-1.

Provided Material and Database Schema

The database that you will use in this project has the schema shown below.



The following files are provided along with your project description:

database.sql contains the SQL commands that create the tables with the schema shown in the figure along with the SQL commands that will load/insert rows in the tables created

Pro2.txt contains 23 queries to be optimized individually. In other words, you should work with each query separately, as shown in the **Setup** section

[What should I Submit?] Deliverable

You are required to submit the (1) **create index commands** used to optimize each query (expected 23 commands) and it should be clear which query every create index command aims to optimize, and (2) a **pdf file** that contains the following:

a) For each of the 23 queries, list which attribute you will create an index on and which index type you will use along with justification

b) List the the physical plans of each query planing and execution time should be visible. If you changed anything in the config file for a query or disabled a certain technique through pgAdmin, then list the command(s)/techniques disabled concerning the query tuning used in the config file for each query (expected 46 plans). If you saved the plans in txt file, then include txt files containing all queries physical plans along with the submission and mention in the report the txt file name that I should inspect for each query.

c) List the parts of the two plans (before and after the index) that had the highest cost, slowest runtime, and largest number of rows for the physical plan presented in the output.

d) Compare the physical plans before and after adding the index for every query

e) Provide a justification for the observed behavior (before plan vs. plan after). Keep in mind to make justification tailored toward each query.

[How will my submission be graded?] Grading

The following scheme will be followed in the grading for every query (23 queries):

- Listing which attribute you will create an index on and which index type you will use along with justification (2 points)
- Providing justification for the index effect on the query performance will be worth (8 points) divided as follows:
  a) List the physical plan of each query (before and after the index, expected 46 plans) along with their planing and execution times (expected 46 planning and execution timing values, i.e. 92 time values in total ;)). If you changed anything in the config file for a query or disabled a certain technique through pgAdmin, then list the command(s)/technqiues disabled concerning the query tuning used in the config file (2 points)
  b) List the parts within the two plans (before and after the index) that had the highest cost, slowest runtime, and largest number of rows for the physical plan presented in the output (2 points)
  c) Compare the physical plans before and after adding the index for every query and provide a tailored-to-query-content-operation justification for the observed behavior (4 points)

Evaluation

An evaluation will be conducted for each team after the project deadline. A sheet will be posted later. During this evaluation, all team members have to be present and show their contribution and their knowledge of every aspect in the project development. Team members who don't show for the evaluation will get a ZERO grade. Team members who show minimal contribution or knowledge of the project implementation will get a deduction in their project grade.

Project Tips and Tricks: Part 2 ;)

In this section, some guidelines and hints are listed below with the aim to guide you in performing the project well.

a) Study the database schema closely

b) Look at the table columns and the values within the table columns

c) Rely on the query optimization part and logical planning content provided in the tutorial to predict the plan of the database engine, so that you can identify the correct attributes to place indices on.

d) Rely on discussions we had within the practical part in the tutorial and related course material including but not limited to lecture slides, textbook, postgres documentation, and your search online in formulating your justification for the observed behavior.

e) Look at every query as if it is the only query that user apply to the database.

f) Don't forget to drop the indices created to optimize previous queries. Also don't forget to undo any changes that you made to the config file (e.g. if you disabled seqscan in the previous query, enable it again) or turning the setting on again from the query tool.

g) Study the plan presented before the index and consider the actions that the planner would perform. Ask yourself, would an index make things better?

h) Also try to disable the operations that the planner often relies upon from the config files and observe the output.

i) Try to execute the to-be-optimized query first to see its output on pgAdmin.

j) You can easily identify common properties regarding the tables in the database by selecting the count of rows in each table. Similarly, you can use aggregate functions and distinct command to know min, max, and number of unqiue values within a given column.

k) Lastly **smile**, we still love you guys and still want you to have the maximum benefit from the project. We wish you happy optimizing and planning days ;)

[When is the deadline?] Submission

In this project, you will work in teams of 3 or 2 student(s) to fulfill the requirements. We expect one submission per team. Please submit your project through this Google form by latest Tuesday December 9th 2025 at 23:59.

Kindly note the following:

- Team members should work equally during this project
- Cheating cases will get a zero grade for all parties involved. i.e. Teams that submit similar projects will get zero (the ones who will copy and the ones who gave their codes)
- Late submissions will get a zero grade
- It is your responsibility to ensure that your submission deliverable along with your data (name, id) are submitted correctly before the deadline.
- Make sure to upload your submission on your google drive or any other online platform and provide a **shareable** link to it
- Team members that don't show to the evaluation will get a ZERO grade for their project
- Team members who show minimal contribution or knowledge of the project during the evaluation will get a deduction