COL362/632: Introduction to DBMS

Assignment 2

Due: 3 March 2023 11:59PM (**hard deadline, will not be extended**)

Revision History

Revision	Date	Author(s)	Description
1.0	18.02.2023	TAs	Initial Version.
2.0	26.02.2023	TAs	First Revision for Clarifications
3.0	-1.03.2023	TAs	Second Revision for Clarifications

General Instructions

1. Please complete the assignment *in pairs*. No collaborations with other entities – person(s)[other than your partner], AI agents, websites, discussion fora etc.– is allowed.
2. You are expected to use PostgreSQL 8.4.22 for this assignment.
3. There are a total of 5 queries in this assignment. Answer all queries.
4. **Marking scheme:** Both parts of the problem statement will be evaluated independently, with all queries having equal weight.
5. Command to run .sql files: \i /path/to/file.sql
6. You will submit a zip file: $< entrynumber1\_entrynumber2 >.zip$. The zip file will contain 1 pdf document and 2 .sql files namely $< entrynumber1\_entrynumber2 >.pdf$, *standard_queries.sql* and *optimized_queries.sql*. Both partners have to submit the assignment with the same file names.
7. Demo will be taken as part of the evaluation. You would be required to generate specified diagrams in Picasso, explain your observations, and explain how you optimized the queries.

Also, note that all the documents that are linked in this file are accessible only within IITD using your OneDrive account.

Plagiarism Rules

If caught, you will receive 0 for the assignment, and a **penalty** of −10% in the course.

1 Picasso

In this assignment, we will be using Picasso database query optimizer visualizer software. Picasso is a software developed by the Database Systems Lab of IISc Bangalore that helps make database query optimization easier to understand. It helps by visualizing the process of optimizing a database query. This visual representation makes it easier to understand what's happening during the optimization process and to identify areas where the query could be improved to run faster.
Picasso can be downloaded from here.
The links for software documentation, tutorials and sample Picasso diagrams can be found here.
We have provided steps for picasso installation and pre-requisite setups here.
For windows users, during installation of java, move up the path of jdk 8 added in the environment variables(move it to top)

2 Dataset

We will be using a large graph dataset containing information about a social network such as Posts, Comments, Tags etc. The data has been slightly modified for the purpose of this assignment to make it compatible with the Picasso tool.

2.1 Description

The database is comprised of the following main tables:

Table Name	**Description**
Organisation	Information about organisation name, location etc
Place	Information about place name, place type etc
Tag	Information about tags used in social media posts and the tag classes they belong to
TagClass	Information about tagclass
Post	Information about post which are messages posted on the social network
Comment	Information about comments which are made in response to a post
Person	biographical information about a person such as name, city location etc
Forum	Information about forums/groups where messages are posted

These following tables are tables which represent edge relationships between the above tables:

Table Name	Description
Comment_hasTag_Tag	Information about tags in comments
Post_hasTag_Tag	Information about tags in posts
Forum_hasMember_Person	Information about members of a forum
Forum_hasTag_Tag	Information about tags in a forum
Person_hasInterest_Tag	Information about which tags a person is interested in
Person_likes_Comment	Information about the comments a person liked
Person_likes_Post	Information about the posts a person liked
Person_studyAt_University	Information about the university a person studied in
Person_workAt_Company	Information about the company a person works at
Person_knows_Person	information about whether a person knows another person

Detailed description of the dataset with tables and their attributes can be found here (link accessible through IITD OneDrive).

2.2 Schema

Schema diagram for the dataset can be found here (link accessible through IITD OneDrive).

2.3 Instructions about data loading

1. Download the cleaned-up data from this link. It is a zip file that contains CSV files for each of the tables described above. It also contains two sql files to be used for data loading.

2. In order to load this data into you database, you need to first define the structure for these data tables. Run the file database_structure.sql using \i /path/to/create_table.sql

3. Once the table structures are defined, you can load the data to your database by running \i /path/to/import_database.sql. **Note that you first need to change import_database.sql by modifying the path of csv files for the corresponding tables**.

2.4 Instructions about the schema

1. For the purpose of this assignment, when we use the term "Message", it refers to both posts as well as comments, i.e. both the tables have to be used. In other words, Post and Comments are sub-classes of Message.

2. Comments are either posted in response to a Post with some parentpostid or in response to a comment with some parentcommentid

3. Note City, Country and Continent are sub-classes of the table Place. They can be inferred from the 'type' attribute in the Place table. The attribute 'partofplaceid' in the Place table establishes a dependency between Continent, Country and City through the 'id' of its parent place.

   Note continents have 'null' value as partofplaceid.

   You can understand this parent-child place dependency with the help of the following example.

Id	Name	PartofPlaceId	Type
153	Agra	0	City
0	India	1454	Country
1454	Asia		Continent

4. Similarly, School and University are sub-classes of Organisation which can be inferred from the 'type' attribute in the table 'Organisation'.

5. Not for a TagClass there is a parent child dependency relation with the column 'subclassoftagclassid'.

   Note that Thing which is a tagClass has 'null' value as subclassoftagclassid. You can understand this parent-child place dependency with the help of the following example.

Id	Name	SubClassofTagClassId
219	Senator	95
95	Politician	211
211	Person	239
239	Agent	0
0	Thing	

6. Note that the edge relationship in Person_knows_Person table is symmetric in nature. For a tuple (Person A, Person B) in that table, it implies Person A knows Person B as well as Person B knows Person A. In this table, you can assume that the constraint "person1id < person2id" (in terms of PostgreSQL comparison) always holds true.

7. A person is said to be a 'friend' of another if they know each other.

2.5 Instructions for Picasso

1. While creating a new DBConnection in Picasso, enter "Public" in the schema field.

2. Note that your query templates in picasso must satisfy the following conditions given in the documentation in the link here

3. Note that postgres version of picasso doesn't satisfy string/varchar as :varies predicate.

4. Note that the query template in picasso supports tablename and their attribute names in lowercase only.

5. Generating Execution Cost Diagram can take considerable amount of time.

6. Pay attention to the resolution parameter and choose appropriate values.

7. For some syntax valid in PostgreSQL, Picasso gives parsing errors. You can't use left joins in query templates in Picasso. In case of multiple tables joined the table containing the attribute which is being varied should appear in the from clause up to the first join or else you could write the queries in Picasso by using multiple tables in from clause using "," separation and joining in where clause.

8. Due to some errors in the Picasso code, you cannot write the ":varies" thing at the very end of the query. (ex: "select id from comment where length :varies;" will give an error).

9. Demo for Picasso diagram interpretation can be found here

2.6 Instructions for Queries

1. Do not output duplicates for any query.
2. Output format mentioned with each query contains the exact column names of expected
3. For each query some input variables are given with some example values for you to test. While submitting you should not hard-code these example values, instead use the variable name itself in the query in the following manner:

   Let's say the variable name is "var1" and the query requires limiting the output number of rows to var1. Then you have to write the query as (note the ":" before variable name): **Select column_name from table limit :var1;**

   For testing if you have used the variable name properly in the query you can run the following command in psql before running your query: **\set variable-name variable-value**
4. \set does not work in Picasso, so while generating plan diagrams use sample values for input variables which are not varied in Picasso.
5. All the variables names input or otherwise will be treated as case-sensitive in submissions. **If the input variable is X, query submissions with lower case character x will get a 0**.
6. All the input variables will be non-null.
7. **Solutions of the queries can be found here**

3 Problem Statement

For each of the queries given in the next section:

3.1 Part 1

1. Write syntactically correct and working SQL queries which give the correct output in *standard_queries.sql*. The format of the file should be as follows. One line should identify the query number (note the two hyphens before and after the query number), followed by the actual, syntactically correct SQL query. Leave a blank line after each query. If you plan to skip a query follow the instructions as it is and leave a blank line instead of SQL query.
   --1--
   SQL QUERY - 1
   --2--
   SQL QUERY - 2
   --3--
   SQL QUERY - 3

2. Generate Exact Plan Diagram, Compiled Cost Diagram & Cardinality Diagram for all the queries and Execution Cost Diagram for any **one** query, using Picasso. For Diagram generation, use given columns in the ":varies" parameter (i.e. range predicate). For variables in queries mentioned to be varied in Picasso, you need to use ":varies" on columns which are restricted by these variables.

   Note: For Picasso variables that are to be varied, if it is not present in any relation you need to vary the underlying column that is being constrained by the Picasso varies variable. In case there are multiple columns/same column at multiple places which are constrained by the same Picasso varies variable, then it is up to you to decide which columns to vary such that you produce Picasso diagrams which have two Picasso Selectivity Predicates. Refer here

3. In the pdf, provide the query templates, all the query plans(plan trees) generated by Picasso and the Exact Plan Diagram, Compiled Cost Diagram, Cardinality Diagram for all the queries and Execution Cost Diagram for any **one** query.

4. Comment on your observations for both plan and cost diagrams.

3.2 Part 2

Optimize the query. Evaluation will be done based on the correctness and runtime of the query(excluding preamble and cleanup runtime)[1].

Submit these optimized queries in *optimized_queries.sql*. The format of the file should be as follows. One line should identify the purpose and query number, followed by the actual, syntactically correct SQL code. Leave a blank line after each SQL code. Any changes in the database created in the preamble section of the query **must** be reverted back in the cleanup section, failure to do so would lead to **penalty of 15% for the assignment**.
--P1--

[1]The configuration of machine that will be used for evaluation is 8 CPUs, Intel(R) Xeon(R) Gold 5318Y CPU @ 2.10GHz, 7.77 GB RAM, 80 GB HDD, Ubuntu 20.04 Server amd64

PREAMBLE - 1(OPTIONAL DEFINITIONS)
--Q1--
SQL QUERY - 1
--C1--
CLEANUP EVERYTHING YOU CREATED FOR THIS QUERY
--P2--
PREAMBLE - 2(OPTIONAL DEFINITIONS)
If you are not writing any preamble, query or cleanup section, follow the instructions for the comments as it is and leave a blank line for the SQL code portion. You may take the help of the Picasso tool to understand the different query plans and execution costs of your solution.

4 Queries

1. For a given list of tags, for each person p_1 interested in some of those tags(p_1 need not be interested in all the tags in the list), recommend new friends(p_2) who do not know p_1. Both p_1 and p_2 should be interested in at least K common tags from the given list of tags(*Condition1*)

   An additional condition is that among all messages posted by mutual friends of p_1 and p_2, both p_1 and p_2 should have liked at least X number of common messages(*Condition2*). To elaborate *Condition2* , let list of messages posted by mutual friends of p_1 and p_2 is:

   Message1: liked by p_1, liked by p_2.

   Message2: liked by p_1, liked by p_2.

   Message3: neither liked by p_1 nor by p_2.

   Message4: liked by p_1.

   Consider only those post messages which have a creation date before *lastdate*(excluding *lastdate*) and comment messages with a length of more than *commentlength*.

   Let X be 3. In this case p_1, p_2 don't satisfy *Condition2*(Only Message 1 and Message 2 are liked by both p_1, p_2). If X is 1 or 2 then *Condition2* is satisfied.

   Your SQL query should handle all values of input variables. K will be smaller than the total tags in the list.

   Only have person1sid, person2sid in output such that person1sid < person2sid.

   You can use the following as sample input variables for testing:
   \set K 2
   \set X 10
   \set taglist '(\'Frank_Sinatra\', \'William_Shakespeare\', \'Elizabeth_II\',
   \'Adolf_Hitler\', \'George_W._Bush\')'
   \set commentlength 100
   \set lastdate '\'2011-07-19\"

   - Input Variable: taglist, X, K, commentlength, lastdate
   - Picasso Varies Variable: commentlength, lastdate

- Output Format: person1sid (person p_1's id) , person2sid (person p_2's id), mutualFriendCount
- Order by: 1. p_1's id (ascending order), 2.mutualFriendCount (descending order), 3. p_2's id (ascending order)

2. For a given *country_name*, count all the distinct triples of Persons such that person p_1 is a friend of person p_2; person p_2 is a friend of person p_3; person p_3 is a friend of person p_1 and all of them were born in the given country and studied in the same university and their birthdays fall in the same month(year could be different). For all persons, the date of account creation should be after *startdate* and before *enddate*.(not including *startdate* and *enddate*). You can ignore the entries with NULL UniversityID

   Distinct triples mean that given a triple t_1, there is no other triple t_2 such that t_1 and t_2 have the same set of elements.

   Your SQL query should handle all values of the input variable. *startdate* would not occur after *enddate*.

   You can use the following as sample input variables for testing:
   \set startdate '\'2010-06-01\'"
   \set enddate '\'2012-07-01\'"
   \set country_name '\'China\'"

   In Picasso, you need to vary the creation date of accounts of the persons, based on how you write the query you can decide to vary the dates for any number of persons, but do explain in the report how you varied them in Picasso. Note that you need to use "varies" in Picasso over two columns simultaneously.

   - Input Variable: country_name, startdate, enddate
   - Picasso Varies Variable: startdate, enddate
   - Output Format: count

3. Find all tagClasses having tags such that message count in duration *begindate* to *middate* (both included) grows at least 5 times as compared to message count in duration *middate* to *enddate* (both included) i.e. (no of messages created in the first interval) >= 5*(no of messages created in the second interval). Consider only those tags which have been used at least once in both date periods. With the tagClass, also give the count of tags which satisfy the above condition in that tagClass.

   You don't have to consider the parent-child relationship of tag classes. Just get the tag class from the tag itself and don't consider parent tag classes of that tag class.

   You can use the following as sample input variables for testing:
   \set begindate '\'2010-02-03\'"
   \set middate '\'2010-12-03\'"
   \set enddate '\'2011-05-03\'"

   - Input variable: begindate, middate, enddate
   - Picasso Varies Variable: begindate, enddate

- Output Format: tagclassname, count
- Order by: count (descending order), tagclassname (ascending order)

4. Find all the messages with at least X number of reply Comments. Find the top 10 most popular tags in these messages.

   As an example, consider post p_1 has 2 reply comments c_1 and c_2. Say, c_1 has 2 reply comments and c_2 has 3 reply comments. Then the message p_1 has 2 reply comments , message c_1 has 2 reply comments and message c_2 has 3 reply comments. In other words, you don't need to consider replies to replies for a post message.

   The most popular tag means the count of messages that has this tag is maximum(in the case of multiple tags with the same count of messages, break ties according to the given below order by constraints). Note that for the count of messages consider only messages with at least X number of reply comments only.

   You can use the following as sample input variables for testing:
   \set X 4

   - Input variable: X
   - Picasso Varies Variable: comment's length, creation date of post message
   - Output Format: tagname, count
   - Order by: count (descending order), tagname (ascending order)

5. Given a *country_name* and a *tagclass*, find all the forums that are moderated by people living in that country and contain **at least one post message** attached to a tag belonging to the given tagclass. For each forum, output the most popular tag and the count of post messages in that forum that are attached to that tag.

   The most popular tag means the post message count for that tag should be maximum in that forum(there can be multiple most popular tags in which case output all of them).In other words, the tag which has been referenced by the maximum post messages posted in a particular forum is the most popular tag for that forum.

   For finding the most popular tag, use the Post_hastag_tag table to count the number of post messages associated with a particular tag for that forum. The table forum_hastag_tag can't be used since it only contains tag information but doesn't have any information about the number of post messages it is associated to.

   Note for the purpose of this query, only the messages in the post table (i.e. post messages) need to be considered.

   You can use the following as sample input variables for testing:
   \set country_name '\'India\"
   \set tagclass '\'TennisPlayer\"

   - Input variable: *country_name*, *tagclass*
   - Picasso Varies Variable: account creation date of the forum moderator, forum creation date

- Output Format: forumid, forumtitle, mostpopulartagname, count
- Order by: count (descending order), forumid (ascending order), forumtitle (ascending order), mostpopulartagname (ascending order)