HOMEWORK 4 (Overview)

When working with databases performance and convenience are key. In order to achieve this we may want to use views, stored procedures and indexing, which is the focus of this assignment.

REQUIRED TASKS (P)

1. If you received feedback to change something in homework 3, or if you simply want to make adjustments of your own accord, implement these changes, clearly state what has been modified, and explain the reason behind each change.

Views:

A view is essentially a stored query. The data from the view does not have to be stored, but the actual query code is saved. This means that the view can be referenced in other queries as a relation instead of having to rewrite complex query code.

2. Views are useful to improve a database, but there are different types of views to consider. Describe the difference between a virtual view and a materialized view.
3. Give an example of a view that could be used in your *social media platform* database.
4. For each statement below, answer whether it is true or false. Motivations are not needed.
   a) A materialized view could contain data that is not up to date.
   b) It takes much longer to create a virtual view than a materialized view.
   c) A query that uses a virtual view always runs much faster than the same query using a materialized view.
   d) A query that uses a materialized view usually runs faster than the same query using a virtual view.

   Important note: The only language you should base your answer on is PostgreSQL.

Stored Procedures:

A stored procedure is a set of stored SQL statements that are stored in the database and can be executed whenever. Stored procedures allow you to include business logic in the database, making code reusable and reducing redundancy. They can accept input parameters, perform insert/update statements, and return results. Something which is super useful when connecting the database to other sources or when doing scheduled updates of data.

5. Create a stored procedure named *investment_return* that takes three parameters: *initial_investment*, *yearly_return* and *number_of_years*. The procedure should return the size of the investment after *number_of_years*. For example, if the *initial_investment* is 100, the *yearly_return* is 0.1 and the *number_of_years* is 10, the procedure should return 259.
6. For the two statements below, answer whether it is true or false. Motivations are not needed.
    a. Unlike functions, stored procedures cannot be called from a SELECT statement.
    b. Stored procedures cannot execute other stored procedures.

    Important note: The only language you should base your answer on is PostgreSQL.

Indexing:

Indexes are used to improve the performance of queries by allowing the database to locate rows more quickly. Just like an index in a book can help the reader quickly look up the correct chapter – indexes allow the database to find rows faster than scanning the entire table sequentially.

7. In your own words, explain what an index is in a database and how it can improve query performance.
8. For each statement below, answer whether it is true or false. Motivations are not needed.
    a. An index is a special case of a materialized view.
    b. PostgreSQL automatically creates an index on primary key columns when a table is created.
    c. Indexes can be created on expressions, not just on table columns.

    Important note: The only language you should base your answer on is PostgreSQL.

Grading Criteria:

- Is the feedback for homework 3, if such was given, implemented?
- Are all changes in task 1, if such were made, implemented correctly and explained?
- Are at least 7 of the True / False answers correct?
- Is the example from question 3 applicable to your actual database?
- Can the stored procedure from question 5 be run in an actual database?
- Is the explanation of indexes from question 7 in-line with the description from the course book?

HOMEWORK 4 P+ (More constraints and views)

Important requirement: To be able to pass the P+ assignment, the P assignment has to pass.

TASKS (P+):

1. In PostgreSQL, CHECK statements can not contain any subqueries. Explain at least one benefit and one disadvantage of this according to the course literature.
2. Santa has gotten tired of deciphering and converting the below RA-expression to SQL every time Santa wants to run the query. Your tasks are to:
    a. Explain what the RA does.
    b. Convert the relational algebra-expression C into an SQL view. Do not forget to motivate the steps you have taken.

Schemas:

Elf(Elf_Name:string, Favourite_Candy:string)

Gift(Elf_Name:string, Wrappingpaper_Type:string, Wrappingpaper_Length:integer, Content_Cost:integer)

Wrappingpaper(Wrappingpaper_Type:string, Cost_Per_Meter:integer)

RA-expression:

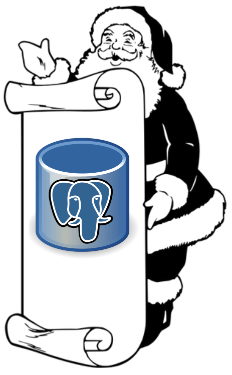
$$A := \pi_{Elf\_Name,\, Wrappingpaper\_Length \times Cost\_Per\_Meter + Content\_Cost \rightarrow cost\_per\_gift}(Gift \bowtie Wrappingpaper)$$

$$B := \gamma_{Elf\_Name,\, SUM(cost\_per\_gift) \rightarrow total\_cost}(A)$$

$$C := \pi_{Favourite\_Candy,\, average}(\gamma_{Favourite\_Candy,\, AVG(total\_cost) \rightarrow average}(B \bowtie Elf))$$

Grading Criteria:

- Is an explanation of at least 1 benefit and 1 disadvantage of PostgreSQL not allowing subqueries in CHECK constraints given?
- Is the relational algebra expression explained clearly and correctly?
- Is the answer a PostgreSQL statement that converts it to a view?
- Does the SQL view match the relational algebra expression?



NOTE: 1) Homework **MUST** be handed in via Canvas **only** and in the correct assignment folder, in one file representing the entire homework. **2)** Your names, group number must be included in the document. **3)** Take into consideration this **exercise is accumulative** and builds upon your solutions. **4) All homeworks are based solely on the course case study.**