Exercises with POSTGRESQL

The goal of this lab is to go over the last two lessons on schemas, views, and on query evaluation.

1 Connecting to the server

To connect to the server you will need to have your identifiers provided at the beginning of the lab. Once you have your identifier you will connect using the `psql` command in the terminal. The general syntax of the command is `psql -h` *hostname* `-p` *port* `-U` *user database*

In this lab we will use the following parameters:

- *hostname* is teach.jachiet.com
- *port* is 5555
- *user* is the one given (of the form tpXXX where the X are numbers)
- *database* will be either `unicode` or your own username.

Once you hit enter you should be asked for a password, type the one given at the beginning of the lab. Once you are successfully connected, I invite you to read `https://tomcam.github.io/postgres/` to learn how to manipulate the psql shell. Note that your user is configured in a way where it can only connect once, you cannot open two connection with the same user!

2 Query evaluation

2.1 Exploring the unicode dataset

In POSTGRESQL adding `EXPLAIN` at the beginning of a command tells POSTGRESQL to show the query plan instead of actually running the query. For this section we will use the `unicode` table in the `unicode` database of the server.

Exercise 1. What is the schema of the table?

Exercise 2. What are the indexes present?

Exercise 3. Design queries such that each of the operators below appears in the query plan of one of your queries. For that you will need to look at the data (number of records, which columns actually contain null, etc.).

- `SeqScan`
- `Index Scan`
- `Index Only Scan`
- `Bitmap Index Scan`
- `Bitmap Heap Scan`
- `BitmapOr`
- `BitmapAnd`
- `Filter`
- `Nested Loop`
- `Hash Join`

- `Merge Join`

Note that some of these operators might be complex to trigger. You need to think about how postgres optimizes to trigger these operations (e..g. `BitmapAnd` or the `Merge Join`).

3 Schema and views

For this part you should switch to the database which is the same as your username.

We consider the following scenario, you are a developer for a game. In the game, there are several kinds of objects that the players can collect (e.g. magical flowers, gems, etc.). Players can also sell or buy objects from a marketplace that works in the following way: players declare some of the objects in their possession as "buyable" with a price for each buyable object and then any player can buy items marked as buyable. When a player buys an item, the possession of the item is transferred to the buyer while the money corresponding to the item is transferred from the buyer to the seller. Note that, as long as the item is not sold, it still belongs to the player. Your goal is to create a database for this marketplace.

The database schema you will develop needs to support players, the money each player has, the various types of objects present in the game and how many objects of each kind each player has.Your database need to be able to support the following kind of operations:

- Create a new type of object or add new players,
- Change the name of a type of objects or the name of a player,
- Attribute an object of a given type to a player,
- Increase or decrease the amount of money a player has,
- Retrieve the list of all the items that a player has,
- Compute the current balance of a player,
- Allow a player to mark one of their item as buyable with a given price,
- Allow a player to buy the cheapest item of a given type from the marketplace.

Ideally your schema should enforce the following constraints:

- All the objects that the players have correspond to some known type,
- Each object is possessed by exactly one player,
- Any single object cannot be marked twice as buyable, nor it can have two different prices (but two different objects of the same kind can have different prices),
- Any player has a positive amount of money.

Exercise 4. Draw the entity-relationship for the scenario.

Exercise 5. Write SQL queries to create the schema.

Exercise 6. Insert a few players, types of objects and objects.

Exercise 7. Write the SQL queries corresponding to each operation that need to be supported. Note that, depending on your schema choice, one operation might correspond to several queries (e.g. one SELECT and then two updates depending on the result of this select).

Exercise 8. If you use several queries for one operation, what happens when several operations are run in parallel? Can you use transactions to deal with that problem?