Final Project DBS2

Athanasios Papanikolaou και Foivos Tzavellos

Database Systems 2
Department of Electrical & Computer Engineering
University of Thessaly, Volos
`{atpapanikolaou,ftzavellos}@e-ce.uth.gr`

Abstract MySQL is a very famous tool among all SQL languages. It is simple, user-friendly and efficient. The same goes for mySQL Workbench. However, for someone new to the field or a complete stranger to these sort of languages, using mySQL and its Workbench is a difficult task. A faster and extremely easier way would be a system that needs different kind of inputs and creates the proper coding for the language to work alone.

1 Introduction

In our project we aimed to create a system that creates the perfect mySQL queries after asking its user for the proper input. The goal was to give the user the complete freedom of his actions and the ability to use the program with minimum SQL knowledge. The program is running on python 3 in jupyter notebook, a well known language and program to run, easily accessible and on a browser. The program is filled with all basic SQL actions and all combinations can be achieved through the correct input in a form of a chatbot that asks questions and executes answers.

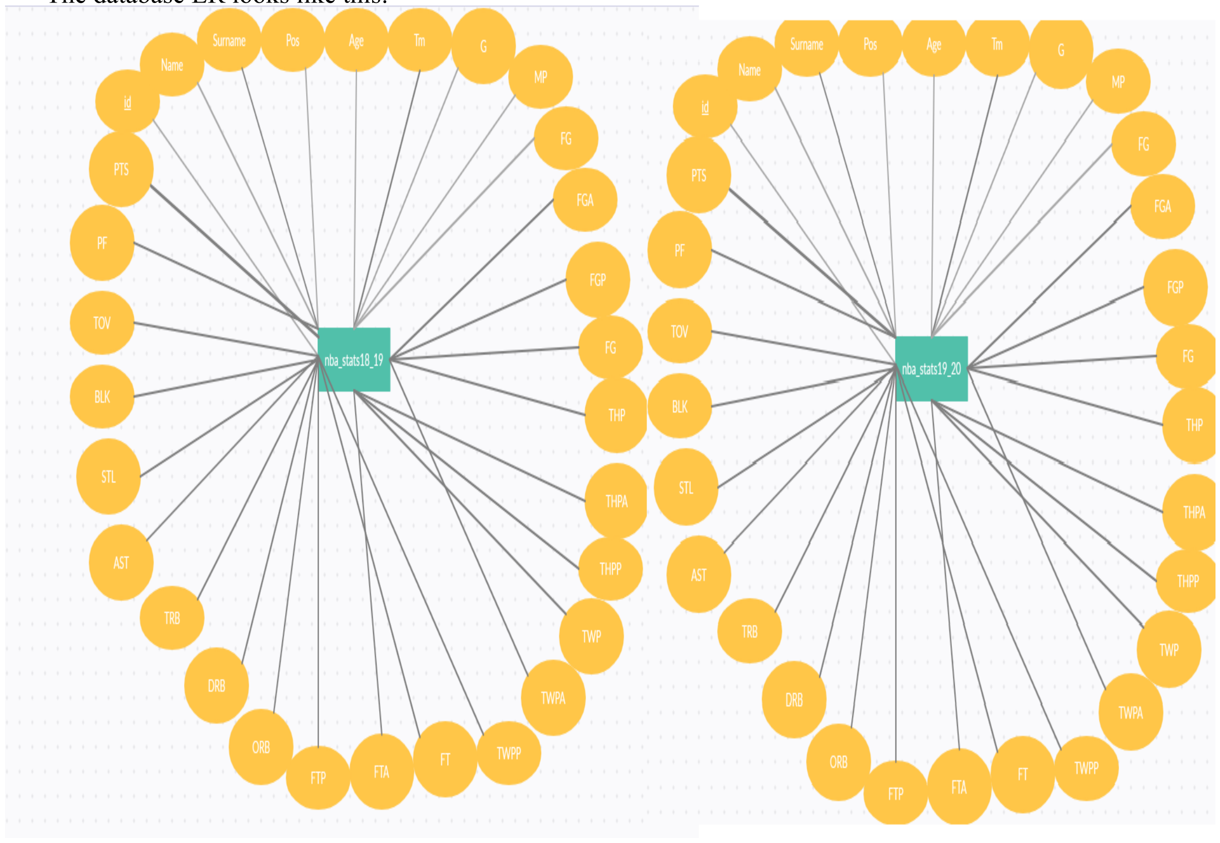
2 The basis of the program

The program is separated into two different notebooks that are connected to each other. The first one consists of all the functions that the second one is calling when needed. These functions are for connecting to the workbench, connecting to a database or creating and connecting to a new one, showing existing tables in the connected database as well as the famous SQL actions of creating and dropping a table, creating, updating and dropping a view, inserting into / updating / deleting a row from a table as well as the complicated select action and one for exiting the whole program.

3 Data used

In our environment we used two datasets of NBA players statistics. The first table NBA stats 18-19 is the stats for the 2018-2019 season of the NBA and the second table NBA stats 19-20 is the stats for the 2019-2020 season of the NBA and after preprocessing them and importing them in the mySQL Workbench (as explained in the appropriate and detailed instructions manual) we started experimenting on them.

The database ER looks like this:



This database was just a testing ground to run experiments on. The chatbot is created in order to access any database and affect any kind of table and query created by the user.

4 The functions notebook

All functions work in the same way as a basis. They have an incomplete query that the chatbot in the other notebook fills with the user input and after that it executes it and creates the final result. Apart from that it prints the final created query for user inspection and possibly returns, if needed, the rows selected to be printed out.

The functions are in a different Jupyter Notebook than the chatbot in order to facilitate the user by preventing him from seeing extra information that he does not need. The only thing the user is interacting with is the chatbot and the workings of the chatbot are hidden from his view.

5 The chatbot notebook

This chatbot allows the user to log in and connect to the mySQL server using his/her credentials, to create a database or connect to an existing one and perform the actions of Show Tables, Create Table, Drop Table, Create View, Update View, Drop View, Insert, Update, Delete, Select as well as Exit the program and end the connection with the workbench, the functions used in the other notebook.

After choosing an action he can input the needed values and call the corresponding function with this input as parameter in order to create the query and execute it.

As mentioned in the Functions notebook, every function used has a query that the user fills with his input in this chatbot and the query is finally executed. The procedure in the chatbot follows questions about the values the user wants to use in each action, whether it is about connecting to the database or using one of the functions to affect a table, giving total freedom and control of the final query/result.

In each action and input in general there are fail-safes that prevent the program from shutting down due to an error (error handling) in order for the bot to continue running and giving the user the chance to repeat the procedure correctly this time.

Every error brings the user back to the original question in order to try again, using a loop to achieve that.

Every action is simple and similar to almost all the others. Questions are asked for input and after proper processing of the input, error handling and calling the function, the query is finally formed and executed. If the user is correct with his input, with correct use of the commas and the quotations for strings as well as correct SQL syntax there can be no misunderstanding of the procedure and no errors.

However, the only complicated part in terms of coding and chatbot questions is the Select action as well as the Where conditions both in Update and in Delete actions.

In these cases, apart from the regular input, there is a chance the user wants to create nested Select queries inside the Where conditions or inside the Select, From and Having conditions of the Select action, by using the special nested Select function which has the same code as the select action in the chatbot notebook with the only difference that it returns the created query without executing it. In this case the user MUST input the condition he wants but instead of typing all the nested select queries he should only type ”()” (without the quotations). After moving to the next step, he will be asked to fill every nested select query one by one in the previous condition and only when he finishes them all will he move to the next condition.

```
E.g.: Select * From table Where column > () AND column2 <
    () Group By column3 = value
```

In this scenario he will be asked the name of the table, the select condition, the where condition and after the system finds the ”()” he will be asked to fill those nested select queries first one by one and possibly any other double nested select queries in them and THEN move on to the next condition, in our case Group By.

Every condition that does not allow nested select queries does not make this extra step of looking for ”()” but only the ones mentioned before. In case no ”()” is found in the input, the procedure continues normally like any other action/condition.

6 Results and Discussion

In retrospect, our program works in every kind of tested scenario and combination of queries. It is made in a way to be used by anyone from beginner to expert and with the user’s local Workbench and databases. The only thing the program is lacking is more complicated functions such as Join and Union. An experienced user could avoid using them or in some way put them inside the queries we are already creating but in the end of the day there are no specific questions asked to fill for these actions in the chatbot.

7 Conclusion and Future Works

The program is very useful and efficient but there are a number of ways it could improve and evolve. Our first thought is that the program could be created as a web application instead of running in Jupyter Notebook, providing a more friendly user interface with the addition of clickable actions and the added benefit of having everything (database and chatbot) in one place. This project can be easily done in the future and we hope that users will find it helpful for their SQL experimentations.