**An Online QBE Processor for MySQL**

**Team members –**

1. Name - Sonal Chavan

E-mail – [schavan6@student.gsu.edu](mailto:schavan6@student.gsu.edu)

Panther id – 002581382

1. Name – Shreejaa Talla

E-mail – [stalla1@student.gsu.edu](mailto:stalla1@student.gsu.edu)

Panther id – 002612614

**How to run the project –**

1. Extract the submitted .zip file
2. Run command ‘python app.py’ in home folder
3. Open index.html file in frontend folder in a browser

* As CDNs are used for Bootstrap, a working Internet connection is a must.

**Project Details-**

As the assignment requires, this application works in 3 steps –

1. User enters credentials of a working MySQL database. If connection is successful, user will be presented with a list of tables from that database along with a number input per table (range 0-2).
2. User enters the values in inputs next to tables they want to use in their query. This number will denote how many skeletons of that table will be shown. Upon clicking ‘Get Skeletons’, corresponding table skeletons in appropriate number will be shown.
3. User enters appropriate expressions for forming a QBE query. On clicking ‘Get Results’, corresponding SQL query and its results will be displayed.

**Extra features other than SPJ queries –**

1. AVG operator in SQL is supported by our app. User can enter P.AVG under column names if they want to see average of a column in results. AVG keyword is also supported in condition box. If user enters ‘P.\_C’ under a column, they can enter AVG.\_C in condition box to compute average of that column in final SQL query.
2. In inputs under the columns, user can enter <, >, <=, >=, != along with constants. E.g. to fetch rows with age column’s value greater than 20, he can enter >20 under the age column.
3. User can order the query results by one or more columns. E.g. to order results by age and salary, he will enter P.AO(1) under age column and P.AO(2) under salary column. This is equivalent to ‘ORDER BY AGE ASC, SALARY ASC’. P.DO(1) is an example of descending order.

**Acceptable input values in a skeleton-**

1. P.
2. P.\_X
3. \_X
4. String constant enclosed by single quotes or a number e.g. 20, ‘CLSO’
5. Constants with comparison operator e.g. <20.
6. Ordering of columns. E.g. P.AO(1), P.DO(2)
7. P.AVG to compute average of that column’s value

**Acceptable input values in condition box –**

1. Comparison operators - >,<,>=,<=,=,!=. They have to entered corresponding to variable names. E.g. \_X>200 or \_Y<\_X.
2. AVG keyword along with variable name. E.g. AVG.\_C

**Technologies used –** Python Flask, HTML, CSS, JavaScript, jQuery, Bootstrap.

**Important Files –**

1. qbe.py – Contains all the grapheme query resolvers for queries to get tables from database, get column details to build skeletons and convert QBE query to SQL and return results.

Important APIs in it –

* resolve\_qberesult(self, info, qbeconditions, conditionBox)

‘qbeconditions’ here is a list of all expressions user has entered in skeleton along with the corresponding column name, table name and table cardinality (i.e. if multiple skeletons of a same table are present, the index of skeleton under which this expression was entered.)

Output of this API is list of result rows retuned from MySQL and corresponding SQL query that was run.

* resolve\_columns(self, info,tablename):

For a given table name, this API will return details of its columns. Used to construct skeletons at the UI.

* resolve\_tables(self, info,username,password,database):

For the first part of the application, this API accepts username, password, database name. It stores them as global variables for use in subsequent queries. And it returns list of tables in that database.

1. index.html – Contains basic skeleton for holding the data. Three project parts have separate div’s dedicated to them. They will be made visible only when program is in the corresponding stage.
2. Main.js – contains all the JavaScript functions that handle the button clicks on index.html, make appropriate Ajax calls, construct dynamic table and display them.

UI validations handled are as following –

* User must select ‘1’ for at least one table name before clicking ‘Get Skeleton’.
* ‘P.’ is the only acceptable value under a table name.
* Condition box must contain a comparison operator or an AVG condition.
* At least one of the columns must contain a P. in its expression.

1. **Contributions by team members –**
2. Sonal Chavan

* Algorithm and implementation of QBE to SQL conversion.
* Graphene queries that handle first two parts of the project (fetch table list from login credentials, fetch column details of tables for displaying skeletons)
* Embedded Bootstrap into the UI to improve appearance.
* UI interactions for part three of the project
* Two extra features – order by and use of comparison operators ‘<,>,<=.>=’ in the columns.

1. Shreejaa Talla

* Extra feature – average (AVG query: backend implementation and UI conditions)
* Initial HTML skeleton
* UI elements and interactions for first two parts of the project
* Testing on multiple DBs
* Testing on multiple browsers