Databases and the Web – QBE to SQL Processor

Assignment-1

Submitted by: Abhinav Peddi, Sathwik Chowda

Overview:

Query by Example (QBE) is a database query language for relational databases. It is the first graphical query language, using visual tables where the user would enter commands, example elements and conditions. Many graphical front-ends for databases use the ideas from QBE today. Originally limited only for the purpose of retrieving data, QBE was later extended to allow other operations, such as inserts, deletes and updates, as well as creation of temporary tables.

This project comprises of a web app which acts as a QBE to SQL syntax processor and executes the QBE commands from the visual structure by converting them into a SQL query.

Front End: *Technologies used: HTML, JQUERY*

- Files in the Project: Index.html, retrieval.js
- Functions in retrieval.js
  - **retrieveTables() :** Used to connect to Backend through graphQL by passing username, password and DB name and retrieve the tables in the DB.
  - **getSkeleton() :** Used to send the selected tables through graphQL url to backend and retrieve the structures (Attributes) of the selected tables.
  - **resetSkeleton() :** Used to reset all the tables under the QBE Interface and the query results.
  - **getResults() :** Used to send the selected tables and the entered values of the attributes through the graphQL url to backend and retrieve the tables from the executed SQL query
- Index.html is the main html file for the project that gives access to the home page.

Back End: *Technologies used: Flask, Python, MYSQL, Graphene*

- Files in the Project: app.py, QBECON.py
- Functions and classes in QBECON.py
  - There are several data types (classes) defined in the file so as to parse the data from the front end and then fit them according to the defined structure while sending back.
  - Functions in Queries section of graphQL: resolve_alltables(), resolve_tableAttributes, resolveQres().
  - Most used datatypes: Dictionaries and Lists.

QBE working statements:

- P.
- P._variable (*Any value in place of _variable)
- _variable
- P.AO(x) (*Any number in place of ‘x’ numbered starting with 1)
- ‘constant’ (*constant must be written in ‘quotes’)

Advance Features:

- Ordering the tables results:
    - Table results can be ordered on any column(integral) of the table either in Ascending or descending.
    - P.AO(number_priority) for Ascending, P.DO(number_priority) for Descending
- Nested Ordering:
    - One can order a column based on another column in the first.
    - Ex: P.AO(1) under age and P.AO(2) [*it can be DO as well] under percentage of a table. Then the table will be ordered by age first and the by percentage.

Contribution:

- **BY** Abhinav Peddi : Worked both on front end JQuery functions as well developed the tables schema and Query functions in the backend.
- **BY** Sathwik Chowda : Worked mostly on the front end to develop the GUI of the web application and also worked on the JS scripts. Later collectively worked on the backend.