Maha Lakshmi Annavarapu
Deepa Muralidhar

Professor Dr. Sunderraman
Advanced Database Design and the Web 1/31/2021

Online QBE processor for SQL

This project is a web programming project which contains three pieces. The Front End, the Middleware and the Back End Database.

MahaLakshmi and Deepa worked on the entire project together. Using Zoom video chat's screen share and remote control facility we both had the entire project running on both computers.

To facilitate the workload sharing, Mahalakshmi will write about the front end and Deepa will write about the back end including the middleware pieces.

The files we programmed are:

Index.html, querytables,py, qbe.py, pp.py

Database used: Classrooms (Provided to us by Professor Sunderraman)

Front End

Outline:

- HTML, Java Script, JSON – To build the Front end we used Html , Java Script an JSON
- Front end exists in the folder named "index".

  Working:

  We opted for Visual studio code as a working platform for the front end. We used Header tags, labels for the input intake, buttons, tables for the front end to give it a shape.

  i.<div id> : It is the tag we used for the table to display.

  ii. Document.getElementById("login"). onclick : In the script tag we used this function for the database to take the credentials of the user and verify them to give the access .

  iii. if-then loop : We used to if-then make sure the user does not give null inputs. If a null input was given, the program pops up with a error alert saying to enter a "Please correct the cred" and also for in valid query saying "please correct the query".

  iv. select: It was used to create a drop-down list to collect the information from the user input. The id attribute is needed to associate the drop-down list with a label.

Maha Lakshmi Annavarapu
Deepa Muralidhar

Professor Dr. Sunderraman
Advanced Database Design and the Web 1/31/2021

AJAX

- Fetch API: We used fetch function to send information with JavaScript .
- .then: To do the next operation after the resource is fetched, we wrote .then call.

  ```
  fetch('http://127.0.0.1:5000/graphql?'
    .then(resp => {
          console.log(resp);
  ```

- POST: We used POST method to form-data inside the body of the URL request.
- JSON: To convert the javascript object into a string we used JSON.stringify().

  ```
  body: JSON.stringify(gqlObj),
   headers: {
   'Content-Type': 'application/json',
  },
  ```

  myJSON is now a string , and ready to b sent to a server.

- For loop : Loop over tables

  ```
  for(var index =0; index< tables.length; index++)
  ```

- Var inputs : To read textbox elements

  ```
  var inputs = table.getElementsByTagName('input');
  ```

- ColumnsValue+ : columnsValue.push(colsValue)
- Row+ : To add the row for the column asked .

  ```
  Row+="</tr><tr>";
  resp.data.qbe.queryData.forEach(data => {
  data.onerow.forEach(colms => {
  Row+='<td style="border:1px solid">'+colms+'</td>'; });
  ```

- Reset Skeletons: A Reset Skeletons button sets all the dropdown to 0

These were some of the important functions and methods learnt and used in the javascript / html piece of the QBE project.

Maha Lakshmi Annavarapu
Deepa Muralidhar

Professor Dr. Sunderraman
Advanced Database Design and the Web 1/31/2021

Credentials:

URL: http://127.0.0.1:5000/

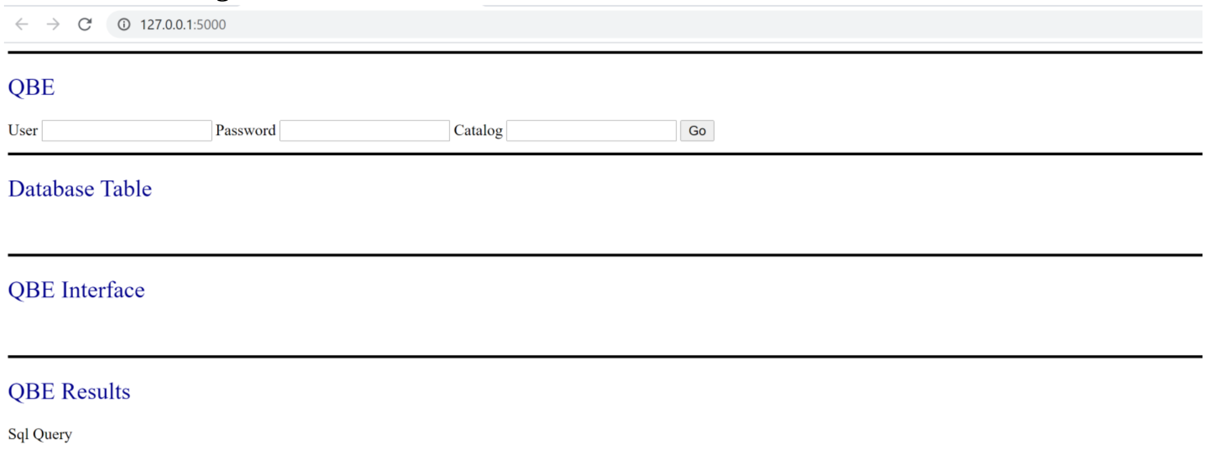
Name: root

Password: mysqlcommunity

Catalog/database: raj

Working Screen Shots:

Phase 1a: Giving credentials to the server


127.0.0.1:5000

QBE

User Password Catalog Go

Database Table

QBE Interface

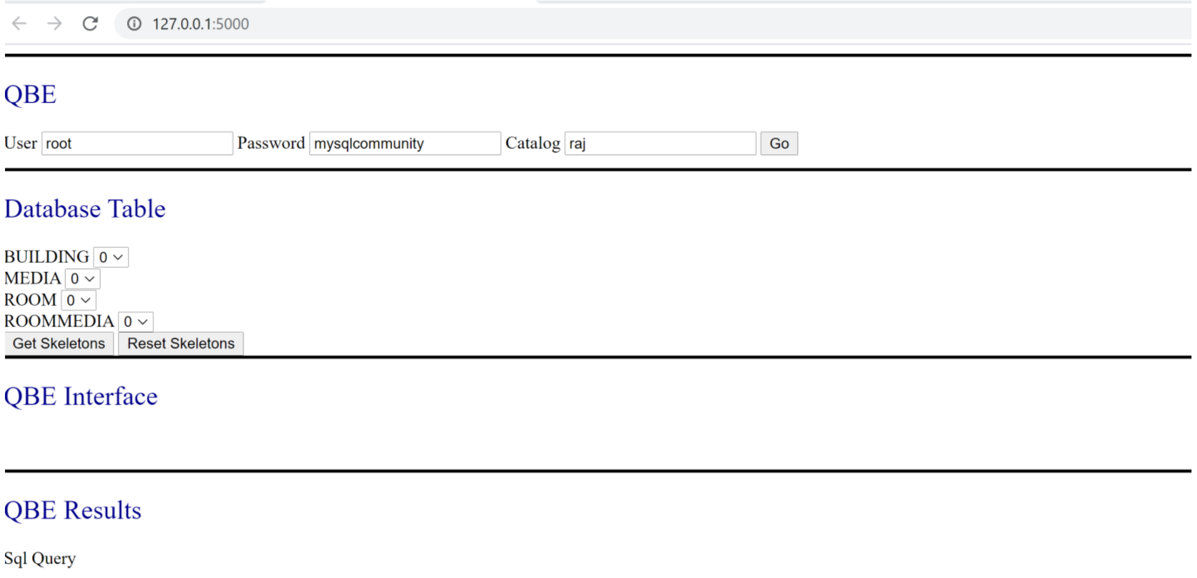
QBE Results

Sql Query

Maha Lakshmi Annavarapu
Deepa Muralidhar

Professor Dr. Sunderraman
Advanced Database Design and the Web 1/31/2021

Phase 1b: Gave credentials, got verified and the Database Table has displayed to take input.



127.0.0.1:5000

QBE

User root Password mysqlcommunity Catalog raj Go

Database Table

BUILDING 0
MEDIA 0
ROOM 0
ROOMMEDIA 0

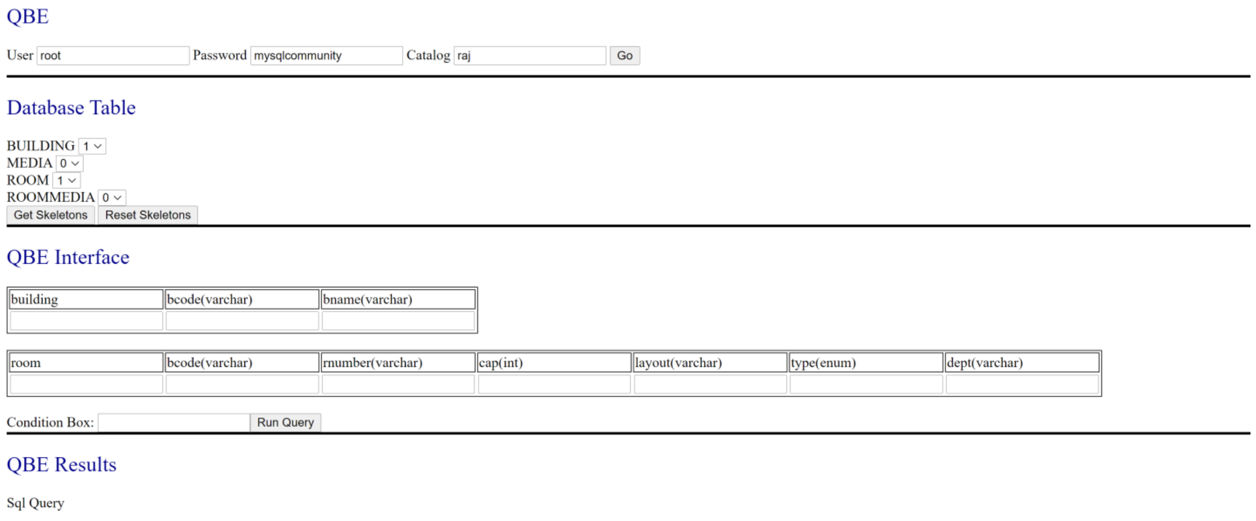
Get Skeletons Reset Skeletons

QBE Interface

QBE Results

Sql Query

Phase 2a: Gave input in the drop down list and clicked on "Get Skeletons" button. Got the QBE interface.



QBE

User root Password mysqlcommunity Catalog raj Go

Database Table

BUILDING 1
MEDIA 0
ROOM 1
ROOMMEDIA 0

Get Skeletons Reset Skeletons

QBE Interface

building	bcode(varchar)	bname(varchar)

room	bcode(varchar)	rnumber(varchar)	cap(int)	layout(varchar)	type(enum)	dept(varchar)

Condition Box: Run Query

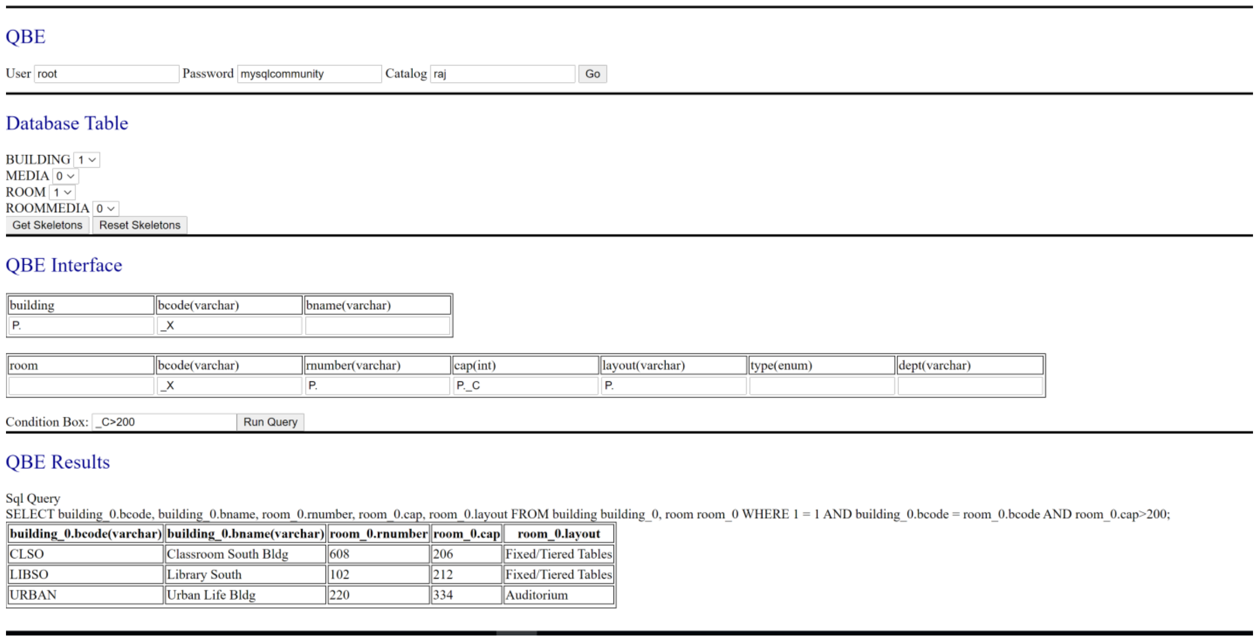
QBE Results

Sql Query

Maha Lakshmi Annavarapu
Deepa Muralidhar

Professor Dr. Sunderraman
Advanced Database Design and the Web 1/31/2021

Phase 3a: Gave input in the QBE interface including the condition box and clicked on the "Run Query" button and QBE results has come with the answer required.



QBE

User root Password mysqlcommunity Catalog raj

Database Table

BUILDING 1
MEDIA 0
ROOM 1
ROOMMEDIA 0

QBE Interface

building	bcode(varchar)	bname(varchar)
P.	_X	

room	bcode(varchar)	rnumber(varchar)	cap(int)	layout(varchar)	type(enum)	dept(varchar)
	_X	P.	P._C	P.		

Condition Box: _C>200

QBE Results

Sql Query

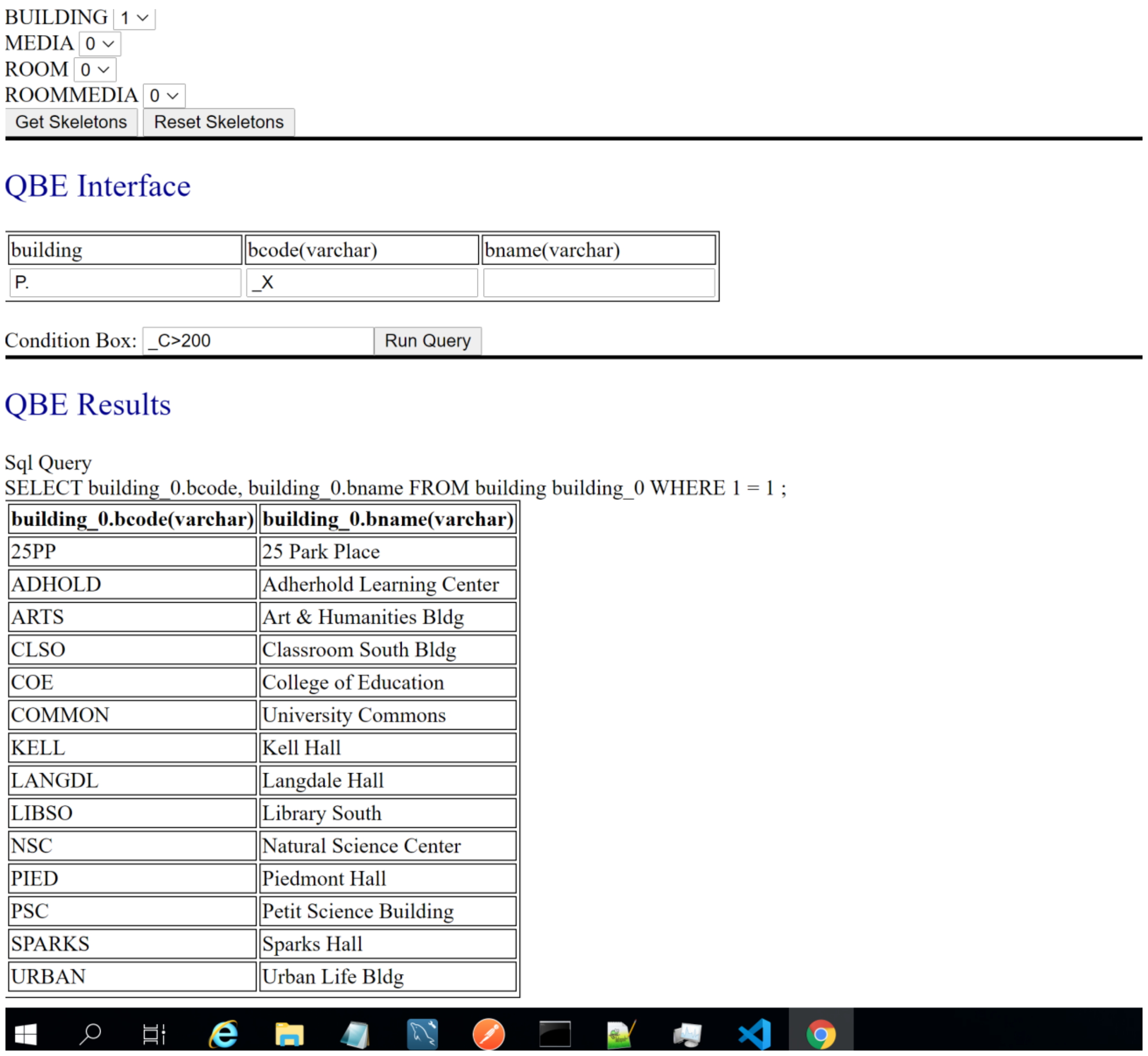
```
SELECT building_0.bcode, building_0.bname, room_0.rnumber, room_0.cap, room_0.layout FROM building building_0, room room_0 WHERE 1 = 1 AND building_0.bcode = room_0.bcode AND room_0.cap>200;
```

building_0.bcode(varchar)	building_0.bname(varchar)	room_0.rnumber	room_0.cap	room_0.layout
CLSO	Classroom South Bldg	608	206	Fixed/Tiered Tables
LIBSO	Library South	102	212	Fixed/Tiered Tables
URBAN	Urban Life Bldg	220	334	Auditorium

Maha Lakshmi Annavarapu
Deepa Muralidhar

Professor Dr. Sunderraman
Advanced Database Design and the Web 1/31/2021

Phase 3b: Trying different query with different combination.



BUILDING 1
MEDIA 0
ROOM 0
ROOMMEDIA 0

QBE Interface

building	bcode(varchar)	bname(varchar)
P.	_X	

Condition Box: _C>200

QBE Results

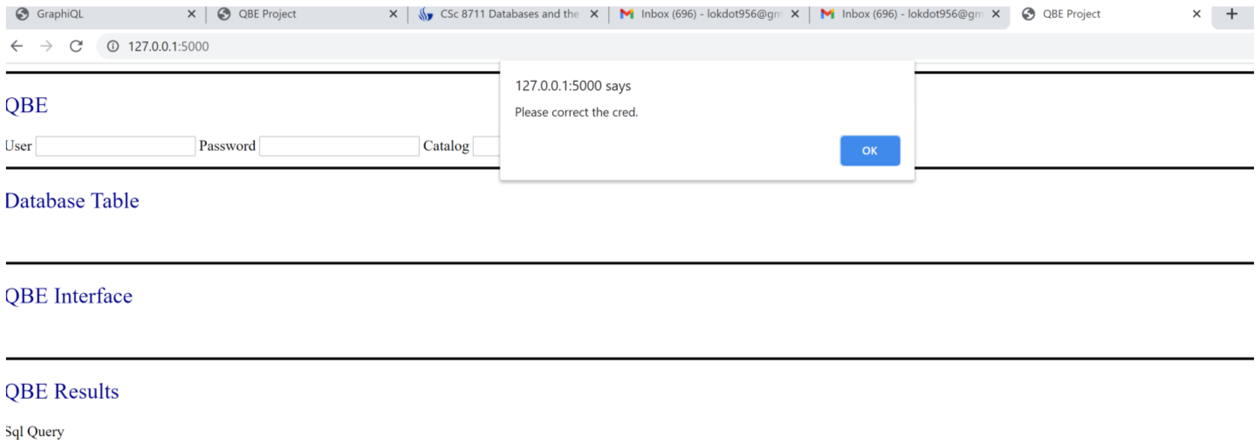
Sql Query
```
SELECT building_0.bcode, building_0.bname FROM building building_0 WHERE 1 = 1 ;
```

building_0.bcode(varchar)	building_0.bname(varchar)
25PP	25 Park Place
ADHOLD	Adherhold Learning Center
ARTS	Art & Humanities Bldg
CLSO	Classroom South Bldg
COE	College of Education
COMMON	University Commons
KELL	Kell Hall
LANGDL	Langdale Hall
LIBSO	Library South
NSC	Natural Science Center
PIED	Piedmont Hall
PSC	Petit Science Building
SPARKS	Sparks Hall
URBAN	Urban Life Bldg

Maha Lakshmi Annavarapu
Deepa Muralidhar

Professor Dr. Sunderraman
Advanced Database Design and the Web 1/31/2021

Error alert:


GraphiQL
QBE Project
CSc 8711 Databases and the
Inbox (696) - lokdot956@gm
Inbox (696) - lokdot956@gm
QBE Project
127.0.0.1:5000
OK

127.0.0.1:5000 says
Please correct the cred.

QBE

User Password Catalog

Database Table

QBE Interface

QBE Results

Sql Query

Back End:

The backend database was mySQL Workbench. We used the classrooms database that was provided to us to make sure that all our programs ran successfully. We used python 3.9 (64 bit) to write our programs. We installed flask, graphene and CORS packages and imported them in our main python program (querytables.py). These constitute the middleware.

We used incremental design and development process as was recommended to us in during our lectures. There were three phases to our project.

Phase I: Our first step was to get GraphiQL to work. (This let us become independent of the design on the front end.) The steps involved to make this work was :

1. To get the listener running (app.py – this was provided to us by the Professor.) We modified this to read the querytables.py program and point to the schema created within queryTables.py. We also added a program statement that pointed the listener to look for index.html
2. QueryTables.py is the main program that has the queries, the class definitions and the three resolve functions. (tablenames, tablestructures and qbe). The queries class has three queries that calls the objects of the three classes.
    a. Tablenames does the authentication of the user id and password. This returns the list of table names in the database.

3. Once the listener was running we ran GraphiQL and the queries on the interface to verify that the functions in the python program was running.
4. Then we tested it with the front end html / java script program to make sure that the list of tables was displayed correctly.

Phase II: The next step after the database authentication was to get the table structure displayed.

5. Tablestructures accepts 2 lists – one for the table names and one for the count which tells us how many of each table needs to be drawn on the screen. It returns a list that has the list of tables and within each entry is a list of all columns in the list. Note that the authentication had to be done each time.

Once verified that this function worked, we identified the query needed by testing it through GraphiQL . This query was then programmed into the front end using JavaScript and we tested that the table structure for the tables selected (1, 2, 0) was correctly displayed.

Phase III: The final step was the QBE to SQL translation.

6. qbe resolve function takes in qbe commands as input and the column names were there are qbe commands and returns the list of tables populated with the data.
    a. This function also calls qbe translator program that translates qbe to SQL and then sends it to the database.
    b. It returns the data in form of a list which is then displayed at the front end.
7. There is an additional function called the pp.py which is third party program we used to make the sql queries print in a nice format.

ErrorChecking:
There is error checking built into the program. It checks for invalid database name, password, etc. There is a also a check for invalid QBE commands such as a missing ".", misplaced "." and unbalanced quote.