CS166 Project Report
Saad Umar
SID: 862156083
Partner: Jeevan Sandhu

Demo Link:

https://drive.google.com/file/d/1PQNY4DBx0hXAXcj4fJYMEK93UYK_f8We/view?usp=sharing

** Please Look at the contribution section below for more details

Introduction and approach:

In this project, we are building a database for a mechanic shop. Using this database, we will be able to track the information about the mechanic shops customers, the cars coming in and leaving the shop, the mechanics, and which mechanic is working on which car, the details about the car ownership, service request which includes information such as the customer's id, the vin of the car, the odometer reading, the complaint, etc. The database will also include billing information.

Our approach to set up this database was to start with the ER diagram. The ER model describes interrelated things of interest and how each entity is related to another and if they have any relationships. After setting up the ER design, we set up the relational Schema Design using the ER model. The final step was to adding onto the relation schema design, our main database program. To set up our program, we used java with the queries in SQL being embedded into the java program. In this phase, our approach was to first get the SQL down and test it out manually to see if the code gives the expected results and then we added that query into the java program, where we had the rest of the interface.

Functioning and Application:

Note: The only change in the given code I made was in the createPostgreDB.sh file and the startPostgreSQL.sh file.

Function1 (AddCustomer): In this function, we add a new Customer into the database. We provide an interface that takes as input the information of a new customer (i.e. first, last name, phone, address). We ask the user to insert all the above required information related to the customer and check if they do not follow the constraints from the relational schema design, then we give an error message. Finally, we take in the user input and if it obeys all the constraints, it is added into the customers table.

Function2 (Add Mechanic): In this function, we add a new mechanic to the database. We are trying to provide an interface that takes in the mechanic information. The information includes the mechanic's first and last name, their id, the mechanic's years of experience. We then insert the user input into the Mechanic table.

Function3 (AddCar): This function allows us to add a new car into the database. We provide an interface that takes as input the information of the car such as its VIN, make, model, year). We check if the user input does not follow the constraints from the relational schema design, then we give an error message. Finally, we take in the user input and if it obeys all the constraints, it is added into the Cars table.

Function 4(InsertServiceRequest): This is the function to insert a service request. We ask the user to input the customer's details and then use that input to check if there are any matching results in the table. If there are any matching results, we print them, otherwise, we display a message asking the customer if they would like to add a new customer. If they answer yes, then

we call the addCustomer function, otherwise, we return back to the menu. If there are matching results, we further ask the user to input the customer id, and if that matches we ask them to choose option 1 or option 0. Option 0 is to choose an existing car while option 1 is to choose of adding a new car. Similarly, if option 1 is selected, we call the AddCar function. Furthermore, if the customer wants to add a new car. We ask the user to input the vin number, make, model, and year make of the car to enter the new service request. Additionally, we also ask them to enter the date, odometer reading, and complaint. We then write our query and execute it to insert this service request into the service request table.

Function 5(CloseServiceRequest): This function was suppose to complete an existing service request. Given a service request number and an employee id, the client application should have verified the given information and create a closing request record. Additionally, This function is to close a currently open service request. The user inputs the rid (request id) and mid (mechanic id) of a currently open request. This is used for searching the open request table and finding the request trying to be closed. The user is then asked to input the date the request was closed, any comments on the request, and the final bill for the service. This will then populate the ClosedRequest table. Since there is no status in the ServiceRequest table, the ClosedRequest table will need to be checked to see if a service request has been closed. Closed requests will still show in the ServiceRequest table.

Function 6: This function asks for listing customers with Bill less than 100. So in the query, we select the customer's first and last name from the customer table, the bill, and complaint from the closed_request table and the date from the service_request table. Then in the where clause, we put our conditions that the bill should be less than 100 and join our "Closed_request" and "Service_request" tables using the "rid" attribute. We also join our "Customer" and "Service_request" table using the (customer)"id" attribute.

Function 7(ListCustomersWithMoreThan20Cars): I wrote a query which lists the customers full name, last name and the total number of cars they own given they have more than 20 cars.

Function 8(ListCarsBefore1995With50,000 Miles: This function asks for listing cars before 1995 and has less than 50,000 miles on them. So in the query, we select the make, model, and year of the car from the "Car" table. Additionally, we also choose the odometer reading from the "Service_Request" table. In the Where clause, we join the "Service_request" and "Car" table using their common attribute which is the "Vin", and then we put our conditions that the Service_Request.odometer should have a reading less than 50,000 and the car make year should be less than 1995.

Function 9(ListKcarsWithTheMostServices): In this function, we ask the user to input the value of k. Our query asks to list the make, model, and number of service requests for the first K cars with the highest number of service orders.

Function 10(ListCustomersInDescendingOrderOfTheirTotalBill): This function asks for listing customers in descending order of their total bill. So in the query, we select the customers first name, last name, their total bill from the Customer table and we select the customer id from the service request table and we also choose the SUM(bill) from the closed_request table. Then in the where clause we ensured the rid of the "Closed_request" table is equal to the rid of the "Service_request" table and Grouped them by their customer id and ordered them in Descending order.

Saad Umar
SID: 862156083
Partner: Jeevan Sandhu

Optimization:

Query Optimization is the process of changing a query in a way that it gives the same result but in a more efficient way. Query optimization is important to reduce the system resources needed to fulfill a query, and also to provide the query result to the user much quicker. Some optimization techniques involve hashing, and trees. I could have optimized my queries by creating an Index. For example, I could have created an index on the customer table by using a B++Tree on the first name and idea of the customer. Additionally, I could have also created an index on the car table using a BTree on the “make” and “vin” attributes. Lastly, I could have created BTree on the “rid” in the service_request table.

Contribution

Initially, we had planned that Jeevan would be doing all the odd-numbered functions while I’ll be doing the even-numbered functions. However, due to some miscommunication, and time-delays, I ended up doing all the functions except 5. But Jeevan also ended up working on functions 1,2, 3, and 5. Hence, I created a small demo video which includes more functionality of our program which I wasn’t able to demo during lab hours. Demo link is given below:

Demo Link:

https://drive.google.com/file/d/1PQNY4DBx0hXAXcj4fJYMEK93UYK_f8We/view?usp=sharing