Marcos Miranda
Isaiah Suizo

CS166 Final Project Report

Assumptions:

During the project we had multiple assumptions, some of which included the acceptable data fields for the tables in our queries. Here is a list of assumptions we made on the data of each table.

In Customer:
ID's consisting solely of integers
First names characters only letters
Last name characters only letters
Phone numbers - Acceptable format denoted by ###-###-####
Address characters only letters

In Mechanic
ID consisting solely of first name
First name characters only letters
Last name characters only letters
Experience is an integer

In Car
VIN consisting of integers and letters
Make characters only from letters
Model characters only from letters
Year consisting solely of integers

In Owns
Ownership id consisting of only integers
Customer id consisting of only integers
Car vin consisting of only integers

In Service_Request
Rid consists of only integers
Customer id consists of only integers
Car_vin consisting of both integers and letters

Contributions and Responsibilities:
We split the work up based on the functions that we were given

Marcos: Add mechanic, insert service request, list customers with bill less than 100, list cars before 1995 with 50000 miles, list customers in descending order of their total bill

Isaiah: Add customer, add car, close service request, list customers with more than 20 cars, list k cars with the most services

Edge case handling was taken on by both of us.

Verification Helper Functions Added:

public static String readBinaryChoice()
- used to read user input for (y/n) questions
- only accepts 'y' or 'n', not case sensitive (capitals work too)
public static int readChoice(int maxVal)
- used to have user select over a range of numbers from 1 - maxVal
- numbers outside range not accepted
public static int readUserInteger()
- used to verify that the user entered a positive integer
- only positive integers accepted
public static String readName()
- used to verify names entered by users (first/last names)
- only accepts either single word names, or composite names separated with '-'
public static String readPhoneNum()
- used to read phone number from user
- only accepts numbers in form ###-###-####
public static String readUserString(String stringType,int maxSize)
- used to accept user input for various strings
- maxSize provided from database table size limits (prevents errors on queries with invalid string sizes)
public static int readYEAR_Domain()
- used to read a year from the user following database domain constraint
- only years satisfying database domain are accepted
public static int readYEARS_Domain()
- used to read # years (of experience) from user
- only #years satisfying database domain are accepted
public static String readDate()
- used to read dates from user
- only accept form of ##/##/####
public static boolean isInt(String userString)
-used to verify that the string passed is an integer

Core Functions Behavior:

1. public static void AddCustomer(MechanicShop esql)
    a. Prompts user for first name, last name, phone number, and address of a new customer.
    b. Id for new customer is fetched from cid_sequence.
    c. New customer is inserted into Customer table in database using respective data values.
2. public static void AddMechanic(MechanicShop esql)
    a. Prompts user for first name, last name, and years of experience for a new mechanic.
    b. Id for new mechanic is fetched from mid_sequence.
    c. New mechanic is inserted into Mechanic table using respective data values.
3. public static void AddCar(MechanicShop esql)
    a. Prompts user for the last name of a customer who will be the owner of the new car.
        i. Function public static int getcIdFromLName(MechanicShop esql) is called to search for user-given last name.
            1. If no user exists with the last name, prompt user for a different last name.
            2. This function will return the id of the customer who owns the new car for use in Owns

    b. Prompts user for the VIN, make, model and year of a new car.
    c. New car is inserted into Car table using respective data values.
    d. New row is added to Owns using respective data values.
4. public static void InsertServiceRequest(MechanicShop esql)
    a. Prompts user to indicate whether to add request for an existing customer or a new one
    b. If existing customer chosen
        i. Prompt user for the last name of customer.
        ii. If several customers with given last name exist, show list of customers and prompt user to select one.
            1. Save id for selected customer.
        iii. If one customer with given last name exists, confirm with user on selection of that customer
            1. If confirmation denied, return to step (a)
            2. If confirmation confirmed, save id for selected customer
        iv. If no customers with given last name exists, return to step (a)
        v. Use customer id to fetch cars owned by that customer, and prompt user to select a car.
        vi. Save vin from selection.
    c. If user chooses to add request for new customer
        i. Call function public static String AddCustomer_ReturnID(MechanicShop esql) to add the customer and save the returned customer id
        ii. Call function public static String AddCar_ReturnVIN(MechanicShop esql) to add a car and save the returned vin
    d. Prompt user for date, odometer reading, and complaint for the service request.
    e. Insert service request into Service_Request using respective data values.
5. public static void CloseServiceRequest(MechanicShop esql)
    a. Prompts user to enter in a valid employee ID for verification and a valid request ID in the Service Request table. User is repeatedly prompted to enter a valid input at each attribute until input is acceptable
    b. User is then prompted to enter a valid close date, bill, and comment to be inserted into the Closed_Request table.

Ways to Improve our DBMS:

1. Using a B+ tree data structure to allow for more convenient querying through more efficient insertion, deletion, and searching.
2. Using a cascading delete method to ensure that all corresponding records from parent to child tables are deleted.