CSC 540 - DATABASE MANAGEMENT CONCEPTS & SYSTEMS

FALL 2018 PROJECT 1 - DATABASE APPLICATION DESIGN & IMPLEMENTATION

CARS - CAR Repair and Service management system
Project Description

Introduction

The goal of the project is to design a relational database application for supporting a chain of car repair and service centers by a company. The project should be carried **out in teams of four (4)** unless prior permission is obtained from the instructor**,** and team peer assessments will be part of the overall grade for each student. The project is expected to be executed in stages with interim deliverables submitted, as described in this document.

The description has several details that you should pay attention to. First, it describes the data and application requirements. It then gives information about deliverables and tentative deadlines (deadlines may change slightly) and "getting started" guidelines.

Note: You should necessarily assume that this is an imperfect description and will be subject to updates. Therefore, it is important that you read through the description in the coming days and ask questions to clarify any missing or ambiguous statements.

Project Specification

Background and Overview

Acme Car Service and Repair Centers is a chain of car service centers owned by Acme corporation spread across the United States. They service and repair only cars that are manufactured by Honda, Nissan, and Toyota.

Your task is to design a database system, CARS, that allows Acme corporation and its customers to keep a record of and manage the following information for all its repair centers. We have suggested the possible attributes for each entity. You may need additional attributes beyond the ones mentioned in this document.

Service Center details

Every state in the United States has three service centers, each center independently managed by a manager, a receptionist and at least five mechanics, and operates 5 days a week (M-F) from 8 AM to 7 PM. Each service center is identified by *a unique ID, a name, an address, and a telephone number*. The employees at these locations fit into one of the three roles: *manager, receptionist, and mechanic*. For each employee, we store their *9-digit employee ID, name,*

address, email address, phone number, the service center he works at, their role at the service center. An employee can work at only one service center. Additionally, we want to store payroll details about the wages paid to the employees (start date, wages, frequency, etc.). The location manager and receptionist are paid a monthly salary while the mechanic is paid hourly based solely on the number of hours that he works servicing/repairing a vehicle (a maximum of 11 hours a day). Assume that a paycheck is generated on the 1st and 15th each month.

Customer details

A customer in our system is identified using a *unique customer id*. Additionally, the system records the following information about the customer: *name, email address (unique), address, phone number, and the vehicles he gets serviced/repaired* at service centers owned by Acme. For each car, the system records the *license plate number* (which uniquely identifies the car), *the type of the car (make, model and year), date of purchase, the last recorded mileage, the type of most recent service/repair, and the date of most recent service/repair*. For sake of simplicity, we shall assume that car ownership does not change and also assume that the license plate number of the car always remains the same.

Inventory details

For each service center, we want to keep track of its parts inventory. For each part in the inventory, that is uniquely identified by a *partid*, we want to record its *name, current quantity, unit price, a threshold for the minimum quantity* that must be present in the inventory and *a minimum order threshold* specifying the minimum quantity that must be ordered for the part whenever an order is placed (see next). If the quantity for a part falls below the given threshold for any reason, the system must automatically create an order (see next) for that part to maintain the quantity above its threshold. A part can be supplied by a distributor (uniquely identified by *distributor Id*) and each distributor has a delivery window for each part e.g. 4 days, 7 days etc.

The part ordering process follows these steps:

1. Calculate the desired quantity to be ordered. One way to do this is to calculate the value *max(required quantity, minimum order threshold)*.
2. Check if any of the other Acme centers has the part in sufficient quantity such that ordering the part from that center does not cause its inventory to fall below the minimum threshold.
3. If one or more such centers are found, an order should be placed to obtain the part from the center with the largest available quantity for that part.
4. If no such center is found, the part should be ordered from its corresponding distributor.

Placing an order creates an order record with at least the following details: *order id, date, part id, quantity, center/distributor id where the part is coming from, the center where the part is going to, order status (complete/pending)*. A separate order is created for each part that needs to be ordered. An order status always starts as “pending” and is changed to “complete” when the order is received by an employee.

To make it simpler, we will assume that parts are always available at the distributor and not bother with keeping the inventory for a distributor.

Car and Service details

Services offered by Acme are either *maintenance* or *repair.* For each type of car that is serviced by Acme service centers, there is a predetermined *maintenance service* schedule based on the number of miles traveled since the last service. Three types of services are provided for each car type: Service A, Service B, and Service C. The service type is determined by the given number of miles on the vehicle e.g. Service A-10K miles, Service B-25K miles, etc. In general, service schedules rotate, first Service A, then B, then C then back to A and so on. Each service includes a number of basic services e.g. oil and filter change, and each basic service is associated with a set of parts in specific quantities. Each basic service is associated with an estimate on hours to complete and a labor charge rate in dollars per hour. There are basically two labor charge rates used by Acme, a *low* charge rate and a *high* charge rate. It is anticipated that some services like changing tires would have a cheaper charge rate than work involving an engine. Acme also provides labor as a complimentary service. That is, any service that is provided for the first time will be charged only for the parts. If it is provided again then the customer will be charged for both parts as well as labor. Also, certain basic services come with a warranty.

Each service type structure is inclusive, i.e., Service B provides everything that Service A does plus some additional services, and Service C provides everything that Service B does plus some additional services.

Repair services include a specific problem reported by the customer, a diagnostic service and fee, and then the actual repair and fees (parts and labor). The diagnostic service produces a report which includes the identified faults and list of recommended repair services. If the repair includes a warrantied service, the cost of the repair is done at no charge. However, there is still a diagnostic fee included.

The system should maintain records of all service appointments for each customer and the invoices associated with each service.

The daily schedule at Acme is broken into 30 minutes time slots from 8 AM to 7 PM. A customer may request an appointment for either a repair or a maintenance service. Each request always includes the car type, model, license plate number and current mileage on vehicle and preference for a mechanic (optional). In response to a maintenance request, the two earliest, “possible” time slots are offered to the customer. By “possible” we mean, the time slots whose duration are sufficient for the estimated time for the required service. The required service is determined by the current mileage of car given by customer and information about the last service. A repair service request is done similarly but also includes the problem that led to the request. Scheduling of appointments tries to ensure that no more than half the day is allocated to maintenance requests in order to allow enough daily openings for repairs which tend to be more urgent. (half the day does not have to be exact. You can consider it a window of between 50 to up to 59% when it is necessary and the rest of the day is open.) The customer then selects one of the two appointments offered (we can assume that one of the two choices is

always suitable). A customer may also later request a rescheduling of an existing appointment. (You should be careful of double bookings for the same time slots given that there is a period between offering a time slot and the customer confirming it).

Besides the availability of a time slot, the scheduling needs to check the availability of parts needed for the service. All parts needed should be either available within the local inventory of the shop, the local inventory of one of the other Acme shops, or the supplier's inventory with a supplier delivery window that is at least 1 day prior to appointment date. It is important that the system not allow overcommitment of parts or double counting. For example, the fact that a part exists in an inventory should not be used to confirm multiple services that will use the same part. In other words, confirming an appointment should capture the fact that though a part is in the inventory it has been committed for a service and will be used by a particular date. When an appointment is rescheduled, the commitment dates for parts will change. In summary, the appointment scheduler should only offer those slots to a customer that meet both the inventory and time slot constraints.

At the beginning of every day, three inventory updating tasks are run: (i.) one task is run to update the counts of parts to be used that day, basically adjusted (decrementing them) to reflect the fact the parts will be removed and actually used that day; (ii) update the status of any pending orders whose items have arrived to "complete" and update their counts (iii.) for any pending orders whose orders have not arrived but is past the delivery window for the distributor, a notification is generated. To simplify the notification system, we do not try to integrate email. Rather, you can just create a notification table that contains a tuple for each notification that needs to be sent. The structure of the table is up to you but should include a date column and you should provide a menu prompt to print notifications for any given date.

These updating tasks will be run via a menu prompt.

Details about the car types, their service schedules and parts required for each service will be provided along with the sample data and an application flow document soon. The application flow document will provide a set of menu options that you should organize your application around.

Role and Access Control Requirements

Employees of Acme and its customers all have unique ids for logging into the system. Customers will use their email addresses as their login id while managers and receptionists will use their 9-digit employee id for accessing the system. Please refer to the application flow document for a detailed description of the access control requirements for each role viz. customer, receptionist, and manager.

Application Requirements

In general, your application should support only role-authorized actions. This will largely be accomplished through the combination of menus that present only appropriate actions for the given role and context, and advanced features like Views, Procedures or Triggers and

Authorization. Additional specification about menu format is provided in the application flow document. This will allow a consistent menu format that will make grading the projects easier. It is expected that your application should handle errors elegantly and not reset on every simple error. For e.g. the application should prompt for another input in case the user input was invalid. Providing user-friendly messages to users when actions are invalid will also be expected.

We expect a command line interface for the application and do not expect a graphical user interface. Developing the latter is strictly optional and no extra credit will be provided for doing so.

Application Flow

This part of description gives a general idea of what the application should be like, which menu items are available for a particular role, the kind of queries to run/actions to perform for a particular menu item, and the kind of output to expect. The application entry point should be an account creation screen (for customers only) or login, if an account already exists, for customers and employees. After customers log in they should be given options like schedule appointments, view invoices etc. A more complete application flow description is available as a separate document.

Sample Queries

Queries on your database will be helpful for assessing the quality of database design. However, it isn't possible to leave that to only demo day. Consequently, you will need to implement some queries as part of your project. The list of sample queries will be given shortly. However, here are some of the required queries that must be supported by your application.

Required Queries

The following shows examples of query classes and queries that should be supported by your application:

Retrieval Queries

1. Find a customer who did not get Service A for a particular car on time
2. Find a customer who has not get any of his cars serviced
3. Find the most frequently used part at the current location
4. Find the make/model of the car that is repaired most often
5. Find the distributor with the maximum quantity of part Air Filter with minimum delivery window
6. Find the mechanic with the most number of labor hours at each service center of Acme corporation

Reporting Queries

1. Generate an invoice for the most recent service for a customer
2. Generate an itemized invoice for the most recent service for a customer (540 only)
3. For each location, show the average number of cars serviced per day
4. For each mechanic, show the average number of hours worked per week
5. For each service, show the average number of miles that a car is driven before receiving that service. Do this across all service centers. (For 540 - Also do this per service center)
6. Display the current inventory levels at the current location
7. Find the number of repair requests in past one month which includes a warrantied service

Insert and Update Queries

1. Create a new user account for the receptionist, and mechanics
2. Update profile information (name, address, phone number, and password) for logged in user
3. Place orders for new parts
4. Schedule a new service

Project Deliverables

Milestone 0 - Questions on the Forum

Post any questions for clarification of the project description on the forum.

Milestone 1 - Report, Due October 8th

For the first milestone, you should:

1. Fill in the form for deciding team members as soon as possible.
2. An ER-Diagram along with a list of Entity and Relationship Types that you identify in the project description. For each relationship type, you should state the arity of the relationship e.g. if it is binary, ternary, etc. Relationships should include any hierarchical relationships (subtypes) that you identify. There is no need for verbose text, just a categorized listing of these is fine.
3. Relational Model: A list of tables, 2 - 3 sentences description of each including what constraints (including referential constraints) it encodes, a listing of functional dependencies, a discussion of normal form choices faced and justification for the decision made.
4. For this report, you should list any application constraints that you identify in the description. Also, include a list of functional dependencies that are present.

5. A statement acknowledging that you have asked all questions you need to clarify any ambiguities in the description.

Milestone 2 - Report and Application, Due Oct 28th

For this milestone, your team will need to submit a complete report which includes:

1. A revised and more complete ER Diagram than that from deliverable 1 and appropriate descriptions.
2. Implementation of E-R model in relational schemas as well as key and participation constraints etc.

Milestone 3 - Final Report & Demo, Nov. 14th

Each team will have to book a timeslot to demo the application to the Professor or Teaching Assistants. The details about the demo will be conveyed later. The following need to be submitted.

1. Constraints: A description of constraints that were not implemented as part of table definitions and why and how they were implemented in the final design. In particular, a separate subsection here should highlight constraints that couldn't be implemented in the database at all and had to be implemented in application code. Note that a key part of assessing your design is how well you used the DBMS to implement constraints v/s implementing in application code.

   Two SQL files: First one should contain SQL for triggers, tables, constraints, procedure. The second file should contain queries for populating the tables with the sample data. Sample data will be provided closer to demo date.
2. Executable file (e.g. - Executable JAR file) and source Java Code.
3. README.txt - This should contain the names of the members and any additional instructions that you think will be necessary to compile and execute your code.
4. Peer review. A link will be provided to submit the review when the milestone will be due.

Grading

Deliverable	Milestone	Percentage	Deadline
Report	Milestone 1	5	October 8, 2018
ER Diagram	Milestone 1	10	October 8, 2018

Relational Model + Constraint	Milestone 1	10	October 8, 2018
Revised ER Diagram + Report	Milestone 2	25	October 28, 2018
Constraints Description + Executable + SQL files + Readme	Milestone 3	50	November 14, 2018
Peer Review		% of Average of peer review grades	November 14, 2018

Getting Started

I would recommend using an IDE(like Eclipse or IntelliJ) for developing the project and some form of version control like GitHub for sharing project amongst team members. (You can create private repositories on NCSU Github). Using Oracle's SQL-Developer would also help you in writing long queries, triggers and procedures because of advanced features like debugging and static analysis of query.

The following link will help you to connect to the database using the JDBC Driver: Creating a connection using JDBC. Students are encouraged to read more about proper handling of connection and JDBC Best Practices.

Note that points will be deducted for **improper handling** of connection in the java application.