mechanics need databases too

Prepared by Group 14: Heather Fillerup- Software Developer, Chris Nelson- Software Developer

CS 340-400: Spr 2020

Project- Step 3 Draft

April 29, 2020

## Upgrades to the Draft version

* Removed Parts table since we only have to implement one M:M relationship
* Removed cost, hours and rate attributes to focus on the tasks and mechanics for the repair order
* Changed the name of the status table to work\_tasks for better clarification
* Changed the name of the repairs table to repair\_orders for better clarification
* Changed repairs\_statuses relationship table to work\_orders and added it as a composite table for better clarification
* Made work\_orders a composite table with attributes moved from work\_tasks (mechanic\_id, start\_date and end\_date). This was to satisfy the requirement that when we delete our M:M task and repair orders relationship record, we cannot delete any record from the repair\_orders or work\_tasks tables.
* Updated the customers participation with cars, a customer can have 0 or more cars, this will allow a customer to be added to the database without requiring a car\_id
* Updated the cars participation with repair orders, a car can have 0 or more repairs, this will allow a car to be added to the database without requiring a repair order
* Changed mechanics relationship with work\_tasks (statuses). Mechanics has a M:M relationship with both repair\_orders and work\_tasks.
* Added parts\_neeeded attribute to repair\_order, this will allow us to use logic to only add an Order Parts work\_order to a repair\_order when true
* Added pair programming to programming assignments because we want to work on everything together if possible.
* Changed year attribute to model\_year and model to model\_name, since year is an SQL keyword
* Removed address fields from customers and description field from cars, it will be concatenated from model\_year, make and model\_name for simplification

## Project Outline

Mahinui auto shop, a single location, has seen record business in the last decade, repairing 50 or more cars on any given day. With more customers coming in by the day, keeping track of records has become a nightmare. The owner, Brad, has finally decided to upgrade his repair order workflow from pen and paper being passed between his 10 mechanics to a website database. Brad is looking to create a system for his mechanics to track the tasks involved with a car's repair, from diagnosis to customer pick up, and be able to view a display of the progress on the homepage. The website will allow users to:

1. Search for car
   1. If car is not found
      1. Search for customer to add to car
         1. If customer is not found
            1. Add customer
      2. Add car
2. Add repair order to a car
3. Add work orders to repair orders
   1. First work order automatically added is the diagnosis task
      1. Followed by customer approval, order parts, repair, test drive and finally contact customer
   2. Add Mechanic to work order
4. Complete current work order and move on to next work order if needed
   1. Order parts task is only added if the repair order needs parts
   2. Provides an option to delete repair order's current work order or delete the entire repair order
5. View on the website homepage the following display of all of the cars currently being repaired at the shop and the current task being performed

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **EXAMPLE DISPLAY** | | | | | | |
| **Repair Order** | **Customer Name** | **Car Description** | **Current Task** | **Start Date** | **End Date** | **Mechanic** |
| 1 | Jason Bateman | 2015 Honda Accord | Diagnosis | 03/1/2020 |  | Johnny |
| 2 | Charlize Theron | 2014 Toyota Civic | Customer Approval | 3/2/2020 |  | Ben |
| 3 | Ryan Reynolds | 2011 Honda Ridgeline | Order Parts | 3/3/2020 |  | Cameron |
| 4 | Scarlet Johansen | 2009 Toyota Front Runner | Repair | 3/4/2020 |  | Peter |
| 5 | Jeff Bridges | 2014 Fiat 500 | Test Drive | 3/5/2020 |  | Frank |
| 6 | Tyler Perry | 2017 Kia Soul | Contact Customer | 3/6/2020 |  | Brian |
| 7 | Brandon Fraser | 2019 Chevrolet Corvette | Diagnosis | 3/6/2020 |  | Ben |

## Programming Implementation and Assignments

For this project we will be implementing pair programming when possible. The goal is for the code in this project to be done together and we will switch back and forth between who is actively programming and who is giving feedback and checking for errors. This will allow us to learn from each other and help ensure that the code is of good quality.

## Database Outline

**customers**: records details about the customers who own the cars being repaired (Heather)

* id: INT, AUTO\_INCREMENT, UNIQUE, NOT NULL, PK
* f\_name: VARCHAR, NOT NULL
* l\_name: VARCHAR, NOT NULL
* contact\_no: VARCHAR, NOT NULL
* email\_address: VARCHAR, NOT NULL
* relationship: a 1:M relationship between customers and cars is implemented with customer\_id as a FK inside of cars, where a customer can have 0 to many cars, and a car can only have one customer.

**cars:** records details about the car being repaired (Chris)

* id: INT, AUTO\_INCREMENT, UNIQUE, NOT NULL, PK
* customer\_id: INT, NOT NULL FK
* license\_plate: VARCHAR, NOT NULL
* make: VARCHAR, NOT NULL
* model\_name: VARCHAR, NOT NULL
* model\_year: YEAR, NOT NULL
* relationship: a 1:M relationship between cars and repair\_orders is implemented with car\_id as a FK inside of repair\_orders, were a car can have 0 or more repair orders and a repair order can have only one car ; a 1:M relationship between customers and cars is implemented with customer\_id as a FK inside of cars, where a car requires one and only one customer and a customer can have 0 or more cars

**repair\_orders:** records details about the repair order being done on a car (Heather and Chris)

* id: INT, AUTO\_INCREMENT, UNIQUE, NOT NULL, PK
* car\_id: INT, NOT NULL, FK
* date\_received: DATE
* date\_completed: DATE
* parts\_needed: TINYINT, NOT NULL, DEFAULT 0
* current\_status: INT, NOT NULL, DEFAULT 1
* relationship: a M:M relationship between repair\_orders and work\_tasks and a M:M relationship between repair\_orders and mechanics are both implemented with a composite table work\_orders; a 1:M relationship between cars and repair\_orders is implemented with car\_id as a FK inside of repair\_orders, where a repair order can have only 1 car, but a car can have 0 or more repairs

**work\_tasks:** records the types of tasks that can be added to repair orders, these tasks are associated to repair orders through work orders (Heather)

* id: INT, AUTO\_INCREMENT, UNIQUE, NOT NULL, PK
* name: VARCHAR, NOT NULL (diagnosis, approval, parts, repair, test, contact)
* relationship: a M:M relationship between repair\_orders and work\_tasks and a M:M relationship between mechanics and work\_tasks are both implemented with a composite table work\_orders

**work\_orders**: composite table that records the tasks that have been added to the repair\_orders and also tracks the mechanic responsible for the work order (Heather and Chris)

* repair\_order\_id, NOT NULL PK
* order\_task\_id, NOT NULL PK
* mechanic\_id: INT, FK
* start\_date: DATE
* end\_date: DATE

**mechanics:** records details of the mechanic responsible for the work orders (Chris)

* id: INT, AUTO\_INCREMENT, UNIQUE, NOT NULL, PK
* f\_name: VARCHAR, NOT NULL
* l\_name: VARCHAR, NOT NULL
* relationship: a 1:M relationship between mechanics and work order is implemented with mechanic\_id as a FK inside of work\_orders, where a mechanic can have 0 or more work\_orders but a work order can only have one mechanic; a M:M relationship between repair\_orders and mechanics and a M:M relationship between mechanics and work\_tasks are both implemented with a composite table work\_orders;

## Actions based on the feedback from previous steps

* Added more details to the project overview and better defined what the website will do and what the tracking display will look like.
* We are keeping our tables/entities lowercased, this makes it easier to code and view, especially in keeping with the format that the SQL keywords will be uppercase, having lowercase tables and columns makes it easier to read. Also we won't have to remember what is uppercase and what isn't.
* Kelley was confused by what we were trying to accomplish. A car comes into the shop for a repair, however a repair goes through various stages\tasks which are assigned to different mechanics and can take varying amounts of time to complete. A repair can have multiple mechanics working on it, e.g. one assigned to diagnose, a different one assigned to get customer approval, etc. This helped us realize that we needed to update our repairs table to repair\_orders, which is the main tracking mechanism for the overall reason the car is in the shop, then we can add work orders (tracked in the composite table work\_orders) that records the various tasks being done to the cars, the mechanic assigned to each task and start and end dates of the task.
* Fixed FK arrow pointing to wrong PK in schema
* Sibai mentioned tracking days needed to repair the car, we could calculate this based on the time between the repair\_orders' start date and end date attributes. However, this would not provide meaningful data because we are not tracking the exact specific type of repair, we are only tracking the mechanic general tasks workflow. E.g. brake repair job is two hours, transmission replacement could be two weeks, 169 hours would be the average.

## Schema

**customers**(

id,

f\_name,

l\_name,

contact\_no,

email\_address,

)

**cars**(

id,

customer\_id,

license\_plate,

make,

model\_name,

model\_year)

**repair\_orders**(

id,

car\_id,

date\_received,

date\_completed,

current\_status)

**work\_orders** (

repair\_order\_id,

work\_task\_id,

mechanic\_id

start\_date,

end\_date)

**work\_tasks**(

id,

name)

**mechanics**(

id,

f\_name,

l\_name,

rate)