Phase 1 Report

CS 6400-Fall 2021

Team 081

Data Types

USER

Attribute	Data type	Nullable
Username	String	Not null
Password	String	Not null
First_name	String	Not null
Last_name	String	Not null

CUSTOMER

Attribute	Data type	Nullable
Email	String	Null
Phone	string	Not null
Street_address	String	Not null
City	String	Not null
State	String	Not null
Postal_code	String	Not null

INDIVIDUAL CUSTOMER

Attribute	Data type	Nullable
First_name	String	Not null
Last_name	String	Not null
Drivers_lices_nr	String	Not null

BUSINESS

Attribute	Data type	Nullable
TIN	Integer	Not null
Business_name	String	Not null
Primary_name	String	Not null
Primary_title	String	Not null

SALE

Attribute	Data type	Nullable
Sold_price	Float	Not null
Sold_date	Datetime	Not null

VEHICLE

Attribute	Data Type	Nullable
Year	Integer	Not Null
Model_name	String	Not Null
Invoice_price	Float	Not Null
VIN	String	Not Null
Color	List<Strings>	Not Null
Description	String	Null

MANUFACTURER

Attribute	Data Type	Nullable
Manufacturer_name	String	Not Null

CAR

Attribute	Data Type	Nullable
Door_count	Int	Not Null
List_price	Float	Not Null
Invoice_price	Float	Not Null
Inventory_date	Datetime	Not Null

CONVERTIBLE

Attribute	Data Type	Nullable
Roof_type	String	Not Null
Back_seat_count	Int	Not Null

TRUCK

Attribute	Data Type	Nullable
Cargo_capacity	Int	Not Null
Cargo_cover_type	String	Null
Axle_count	Int	Not Null

VAN/MINIVAN

Attribute	Data Type	Nullable
Has_driver_back_door	Boolean	Not Null

SUV

Attribute	Data Type	Nullable
Drivetrain_type	String	Not Null
Cupholder_count	Int	Not Null

REPAIR

Attribute	Data type	Nullable
Labor_charges	Double	Null
Description	String	Not null
Completion_date	Datetime	Null
Odometer_reading	Int	Not null
Start date	Datetime	Not null

PART

Attribute	Data type	Nullable
Quantity	Int	Not null
Price	Double	Not null
Vendor_name	String	Not null
Part_number	String	Not null

Business Logic Constraints

Ronald Around (Owner)

Only Ronald Around (Owner) can enter sold prices that are less than or equal to 95% of the invoice price.

Only Ronald Around (Owner) can update the labor charges on a repair to a value less than their previous value.

Salesperson and Sales

If a sold price is less than or equal to 95% of the invoice price, the system will reject the sale.

Vehicles are bought by customers via a salesperson.

The purchase date should be tracked to determine when a vehicle leaves inventory.

The list price (List_price) is calculated as 125% of the invoice price, however, customers can negotiate and receive a lower price (Sale_price).

Vehicles

The model name and model year are entered by the user in free form.

Model years cannot exceed the current year plus one.

The year entered must include century digits.

A description field can be free form of multiple pieces of additional information

The invoice price must include dollars and cents. On the client-side, two separate boxes (dollars and cents) may enforce both values to be entered.

Service Writers and Repairs

Only one repair can be open (have a null completion date) at a time.

Any updates on labor charges cannot be less than their previous values unless updated by the owner.

Once created, only labor charges and parts can be changed, or the repair marked complete.

Task Decomposition and Abstract Code

Abstract Code Legend

Form - bold, underline

Task - bold

Button - bold, italics

ENTITY - bold, all caps

Attribute - italics

$variableNames - camelcase, preceded by “$”

Abstract Code

Main Landing

Task Decomposition

Lock Types: Read only.
No. of Locks: Single.
Enabling Conditions: None(without login) or Triggered By successful login.
Frequency: High.
Consistency (ACID): Not critical**.** Order is not critical.
Subtasks: Decomposition not needed. Mother task is not needed.

Abstract Code

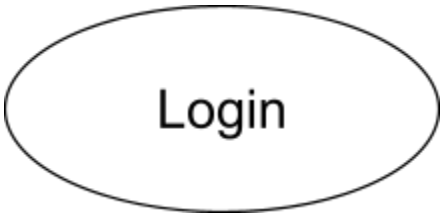
- User lands on the **Main Landing** form.
- Run ***Number of Total Vehicles Available*** task and show the value on the page.
- Show search fields for *Vehicle_type* , *Manufacturer_name*, *Model_year* , *Color* **Dropdowns, *Search Vehicles*** button**, *List_price*** search tab**, *Keyword*** search tab , and ***Log In*** link.

- When the user logs in,in addition to the above , on the **Main Landing** form
  - If the logged in user is **MANAGER**,also show ***View/Generate reports*** link, filter by ***sold/unsold/all vehicles*** dropdown, ***Search Vehicle by VIN*** search tab.
  - if the logged in user is **INVENTORY CLERK**, also show ***Add vehicle*** button/link to **Add Vehicle** form, ***Search Vehicle by VIN*** search tab.
  - if the logged in user is a **SALESPERSON**, also show ***Search Vehicle by VIN*** search tab.
  - if the logged in user is a **SERVICE WRITER**, also show ***Search Vehicle by VIN*** search tab, link/button to the **Repair** form.
  - if the logged in user is the **OWNER**, run all tasks as above and show all the tabs and links viewable to each individual logged in users.

LOGIN

Task Decomposition

Lock Types: Read-only.
No. of Locks: Single.
Enabling Conditions: None
Frequency: High
Consistency (ACID): Not critical**.** Order is not critical.
Subtasks: Decomposition not needed. Mother task is not needed.



Abstract Code

- The user enters email and password in the corresponding fields
- If data validation is correct for both fields user and password then:
    - When the ***Login*** button is clicked:
    - If $userName record is found but $password does not match
        - Go to **Login** form showing an error message
    - If $userName and $password are correct
        - Get the user role and persist information along with the session
        - Check role of the current user and go to **Main Landing** form showing fields according to the role
- Else if both user and password are incorrect, display the **Login** form with a message indicating the error.

Get Number Of Total Vehicles Available

Task Decomposition

Lock Types: Read-only on **VEHICLE, SALE** tables.
No. of Locks: 2
Enabling Conditions: None.
Frequency: High.
Consistency (ACID): is critical(properly reflect the correct number of vehicles for purchase only)
Subtasks: No decomposition needed.

Abstract Code

- When a user lands on the **Main Landing** form, run the **Get Number Of Total Vehicles Available** task**.**
- **Get Number Of Total Vehicles Available-** Jumps to the **Get Total Vehicles Available** task**.**
    - Run query on **SALE** table and **VEHICLE** table select count of vehicle.* using *VIN* as a join where the sold price is not null.
    - Display count.

https://gatech.instructure.com/courses/196970/assignments

Get Options for Search Dropdowns

Task Decomposition

Lock Types: Read-only on the **VEHICLE** table.
No. of Locks: Single.
Enabling Conditions: None.
Frequency: High.
Consistency (ACID): not critical.
Subtasks: No decomposition needed.

Abstract Code

- When a user lands on the **Main Landing** form, run the **Get Options for Search Dropdowns** task**.**
    - **Main Landing**. Populate *Vehicle type* , *Manufacture*r, Y*ear, Color* ***dropdowns*****.**
    - if no button is clicked, do nothing. Otherwise, users may select different options from the dropdowns and may enter the list price and/or keyword.
    - User hits ***Search Vehicles*** button,
        - Jump to the **Search Vehicles** task.

Search Vehicle by VIN

Task Decomposition

Lock Types: Read-only on the **VEHICLE** table.
No. of Locks: Single.
Enabling Conditions: User clicks on ***Search Vehicle by VIN*** tab.
Frequency: Medium.
Consistency (ACID): Critical.
Subtasks: No decomposition needed.

Abstract Code

- When the ***Search Vehicle by VIN*** search tab is clicked:
  - User clicks on the ***Search Vehicle by VIN*** and enters *VIN*.
  - If data validation is successful, then:
    - When enter button is clicked:
      - If a record is found, go to the **Vehicle Details** form.
    - Else:
      - Go back to **Main Landing** form, with the error message “Sorry, no such Vehicle found!”.

Search Vehicles

Task Decomposition

Lock Types: Read-only on the **VEHICLE** and **SALE** tables.
No. of Locks: 2
Enabling Conditions: User clicks on ***Search Vehicles*** button.
Frequency: High.
Consistency (ACID): Critical, vehicle sale status must be most up to date.
Subtasks: No decomposition needed.

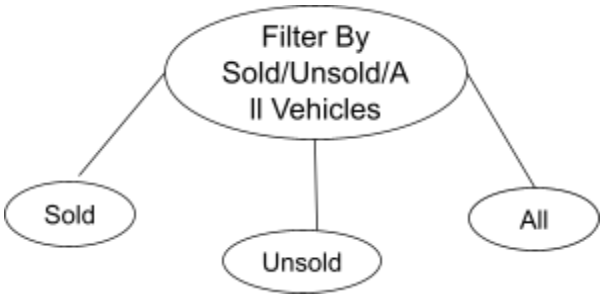
Abstract Code

- When the ***Search Vehicles*** button is clicked-Jump to **Search Vehicles** task.
- Run the **Search Vehicles** task: query for information on the **VEHICLE, SALE** tables where sold price is null using *VIN* as join.
- select *VIN*, *Year*, *Model_name*, *Manufacturer_name*, *Colors*, *List_price*, *Description*, *List_price* from **VEHICLE** corresponding to chosen criteria by user. Additionally the type of the vehicle will be used for filtering.
    - if vehicle has single color:
        - In ascending order with respect to *VIN*, display all vehicles that match the selected criteria from ***dropdowns*** (the *Color*, *Vehicle_type*, *Year*, *Model_name*, *Manufacturer_name*, List_price, keyword and *VIN*) If a keyword was entered and matched the description, indicate this with a checkbox.If the user selects one individual result, jump to **Vehicle Details** form.
    - Else if a vehicle has multiple colors:
        - In ascending order with respect to *VIN*, display all Vehicles that match the selected criteria entered (the *Color*, Vehicle_type,*Year*,*Model_name* , *Manufacturer_name*, *List_price*, keyword and *VIN*) and a single row with all colors listed.If a keyword was entered and matched the description, indicate this with a checkbox. If the user selects one individual result, jump to **Vehicle Details** form.
    - Else:
        - If no record is found matching the selected criteria ,go back to **Main Landing** form and display an error message: “Sorry, it looks like we don’t have that in stock!”

Filter By Sold/Unsold/All Vehicles

Task Decomposition

Lock Types: Read only on the **VEHICLE** and **SALE** tables.
No. of Locks: 2
Enabling Conditions: Manager/Owner clicks on ***Filter by sold/unsold/all*** dropdown.
Frequency: High.
Consistency (ACID): Critical.
Subtasks: Decomposition needed.



Abstract Code

- When user clicks on ***Filter by sold/unsold/all***
  - Populate ***Filter by sold/unsold/all*** dropdown. A dropdown opens, user selects either sold or unsold or All vehicles.
  - if the user clicks on **unsold**:
    - Run a query on the **SALE**, **VEHICLE** tables, use *VIN* as a join and see where the sold price is null.
      - Display all vehicles as per the query.
  - if the user clicks on **sold**.
    - Run a query on the **SALE**,**MANUFACTURER**, **VEHICLE** tables.Use VIN as a join and see where the sold price is not null,
      - Display all vehicles as per the query.
  - Else:
    - Run a query on **VEHICLE,MANUFACTURER** Table and display all Vehicles, if **all vehicles** selected from dropdown.

Add Vehicle

Task Decomposition

Lock Types: Write-only on Vehicle Table
No. of Locks: 1, insert new row into the **VEHICLE** table.
Enabling Conditions: Triggered when successfully logging in and adding a new vehicle.
Frequency: Low
Consistency (ACID): Not critical, all vehicles can be accessed as a new one is added. Other vehicles may be updated while a new one is added.
Subtasks: No mother task needed, adding a new row to a single table. No decomposition is needed.

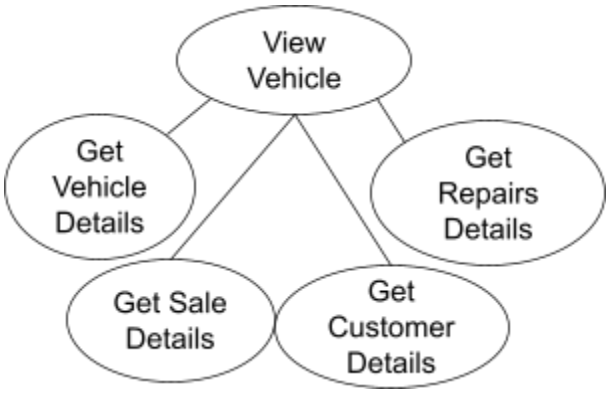
Abstract Code

- Show **Main Landing** form**.**
- Upon:
    - Click ***Add Vehicle*** button **-** Jump to the **New Vehicle** form**.**
- **INVENTORY CLERK** inputs all prompts for a complete new vehicle description**.**
- **INVENTORY CLERK** name is also recorded
- Upon:
    - Click ***Submit New Vehicle*** button - perform the following:
        - check that all fields are complete
            - if input fields are empty, return “missing field” alert, show which fields are missing by outlining them in red, and allow the clerk to fill in and submit again.
        - for all inputs, check that types match expected schema
            - If schema doesn’t match, return “wrong schema” alert, outline which fields have wrong schema. Allow the clerk to correct and submit again.
        - if all checks pass, submit the record to the database and perform **Add Vehicle** task

View Vehicle

Task Decomposition

Lock Types: Read-only **VEHICLE**
No. of Locks: Several different schema constructs are needed.
Enabling Conditions: When a vehicle is clicked
Frequency: High, accessed by both customers, and employees
Consistency (ACID): critical, if a vehicle is sold, it must be reflected.
Subtasks: Mother task needed. Decomposition needed.Subtasks depending on the user role.



Abstract Code

- Show **Main Landing** form**.**
- Upon:
  - Click **Vehicle** link **-** return **View Vehicle** form**.**
- Perform **View Vehicle** task and run the following steps/query to populate the form:
  - Check user role.
  - If the role is anonymous, sales person, service writer, :
    - select *VIN*, *Vehicle_type*, *Year*, *Model_name*, *Manufacturer_name*, *Color*, *List_price*, *Description* from **VEHICLE** and **MANUFACTURER**.
  - If the role is Inventory Clerk :
    - select *VIN*, *Vehicle_type*, *Year*, *Model_name*, *Manufacturer_name*, *Color*, *List_price*, *Description, Invoice_price* from **VEHICLE** and **MANUFACTURER**.
  - If the role is Managers or Roland Around:
    - select *VIN*, *Vehicle_type*, *Year*, *Model_name*, *Manufacturer_name*, *Color*, *List_price*, *Description*, *Invoice_price*, *Inventory_date*, from **VEHICLE** and **MANUFACTURER**.
    - If the vehicle has been sold:
      - select *Sale_date*, *Sale_price* from **SALE**
      - select all but *Drivers_lices_nr* and *TIN* from **CUSTOMER** where *Drivers_lices_nr* /*TIN* matches each customer for each vehicle sold.
    - If the vehicle has repairs
      - select the customer *Name* (*First_name* and *Last_name* for individuals, or *Buisness_name* for companies), the *Service_writer* who entered the repair, and the repair's *Start_date*, *Completion_date*, *Labor_charges*, sum(part *Price* * *Quantity*) as *Parts_cost*, and *Total_cost*.

Lookup Customer

Task Decomposition

Lock-types- Read only for User.
Number of Locks- Single.
Enabling Condition- User's Login and Trigger by sale or repair.
Frequency- Medium.
Consistency- not critical.
Subtasks- Mother task is not needed. No Decomposition needed.

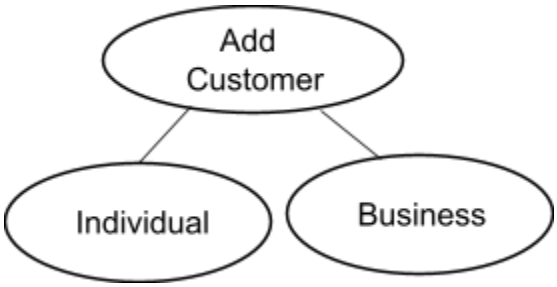
Abstract Code

- User enters *drivers_lices_nr or tax TIN* in the input field.
- If data validation is correct, then:
- Click ***Enter*** button:
    - If *drivers_lices_nr* or *TIN* is found
        - Go to the **CUSTOMER** page.
    - Else
        - Go to **Add Customer** task
- Else input field is invalid, display input field again with message "Try again".

Add Customer

Task Decomposition

Lock-types- Write only for User.
Number of Locks- 2, Individual and Business schema constructs are needed.
Enabling Condition- when no look up customer is found.
Frequency- Medium.
Consistency- not critical.
Subtasks- Task is decomposed into two tasks. Mother task is required to coordinate sub tasks.



Abstract Code

- User clicks ***Add Customer*** button.
- User enters phone*, address* and *email* in the input field.
- If **CUSTOMER** is Individual, then
    - User clicks on ***Individual*** button
    - User enters *First name, Last name* and *Drives_lices_nr* in the input field
- Else
    - User clicks on ***Business*** button
    - User enters *Business name, TIN, Primary Contact Name* and *Primary Contact Title* in the input fields.
- User click ***Add*** button

Enter Sale

Task Decomposition

Lock-types: Write-Only on Sale Table
Number of Locks: 2, Vehicle and Customer
Enabling Condition: Triggered By successful search of a vehicle on the **Main Landing** form and the following click on the Sale option.
Frequency: High
Consistency: not critical, the order is not critical
Subtasks: Mother task is not needed. No Decomposition needed

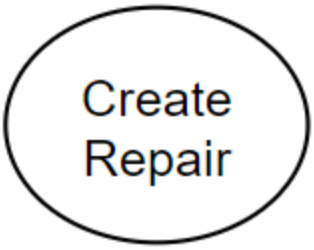
Abstract Code

- User bring vehicle information from **Search Vehicle** Form
- User lookup for Customer using search
    - Run the **Lookup Customer** task using *driver's license or tax ID*
        - If customer exists
            - Show customer information
        - If the customer does not exist
            - Add customer using **Add Customer** task
- The user enters the *Sale Price* value
- The user enters *Sale Date*
- When the User clicks ***Enter Sale*** button
    - Get $userName from the information of the current user using the system
    - Check role of the current user
    - Price Sale is validated
        - If $priceSale is not present or is not numeric
            - display error
        - If the role of the current user is Sales Person then
            - If the $soldprice is less than or equal to 95% of the Invoice price
                - Show error message
        - Else if the role is Owner then
            - Allow inserting any numeric value
    - The sale Date is validated
        - If $saleDate date is not present
            - display error
        - If $saleDate is greater than the current date
            - display error
    - Insert VIN, Sale Price, Sale Date, Username into **SALE** Table
- Jump to **Main Landing** form

Create Repair

Task Decomposition

Lock Types: Write-only.
No. of Locks: 1, only Repair schema construct involved.
Enabling Condition: *VIN* and *DLN or TIN* set.
Frequency: Medium.
Consistency (ACID): Not critical.
Subtasks: No subtasks are required and no mother task is required.



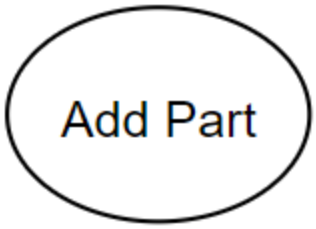
Abstract Code

- User enters the *Description* and *Odometer_reading* in the input fields
- User clicks the ***Create Repair*** button
- *Odometer_reading* is validated to be numerical
    - If *Odometer_reading* is not numerical
        - Display error message
- *Start_date* set to the current date
- *VIN*, *Description*, *Start_date*, and *Odometer_reading* inserted into the **REPAIR** table
    - Insert *VIN, Start_date,* and *DLN* or *TIN* into the appropriate table

Add Part

Task Decomposition

Lock Types: Write-only.
No. of Locks: 1, only Part schema construct involved.
Enabling Condition: None.
Frequency: Medium.
Consistency (ACID): Not critical.
Subtasks: No subtasks are required and no mother task is required.



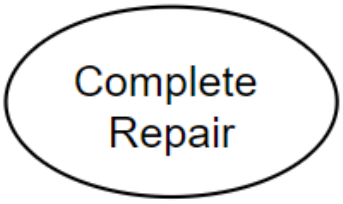
Abstract Code

- User enters the *Part_name*, *Vendor_name*, *Price,* and *Quantity* in the input fields
- User clicks the ***Add Part*** button
- *Price* is validated to be numerical
    - If *Price* is not numerical
        - Display error message
- *Quantity* is validated to be an integer
    - If *Quantity* is not an integer
        - Display error message
- *Part_name* is validated to be present
    - If *Part_name* is not present
        - Display error message
- *Vender_name* is validated to be present
    - If *Vender_name* is not present
        - Display error message
- *VIN*, *Start_date*, *Part_name, Vender_name*, *Price*, and *Quantity* inserted into the **PART** table
- Jump to **Show Repair** task

Complete Repair

Task Decomposition

Lock Types: Write-only.
No. of Locks: 1, only Repair schema construct involved.
Enabling Conditions: Completion date not set.
Frequency: Medium.
Consistency (ACID): Needs to be consistent with **View Repair** to avoid showing a closed repair.
Subtasks: No subtasks are required and no mother task is required.



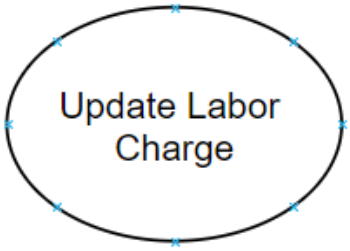
Abstract Code:

- User clicks the ***Complete Repair*** button
- $completionDate set to the current date
- *Completion_date* is updated to $completionDate on the **REPAIR** table
- Jump to **Main Landing** form

Update Labor Charge

Task Decomposition

Lock Types: Write-only.
No. of Locks: 1, only Repair schema construct involved.
Enabling Conditions: Completion date not set.
Frequency: Medium.
Consistency (ACID): Not critical.
Subtasks: No subtasks are required and no mother task is required.



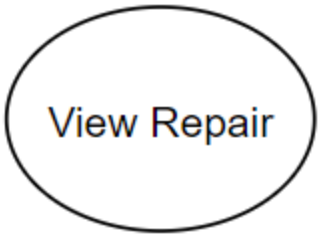
Abstract Code

- User enters the *Labor_charges* in the input fields
- User clicks the **Update *Labor Charge*** button
- *Labor_charges* is validated to be numerical
    - If *Labor_charges* is not numerical
        - Display error message
- If user is not owner
    - *Labor_charges* is validated to greater than the current value
        - If *Labor_charges* is less than the current value
            - Display error message
- *Labor_charges* is updated on the **REPAIR** table

View Repair

Task Decomposition

Lock Types: Read-only.
No. of Locks: 3, Repair, Part, and Customer schema constructs involved.
Enabling Conditions: User entered valid VIN
Frequency: Medium.
Consistency (ACID): Needs to be consistent with **Complete Repair** to avoid showing a closed repair.
Subtasks: No subtasks are required and no mother task is required.



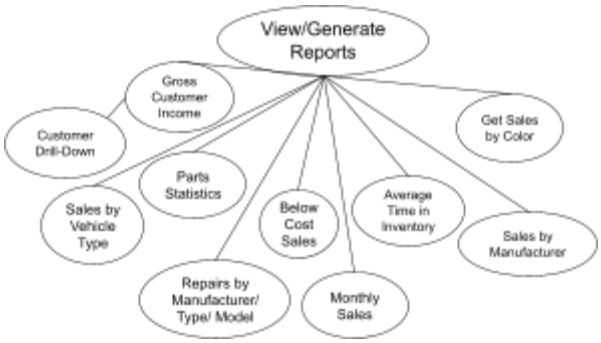
Abstract Code

- Select *Labor_charges*, *Description*, *Odometer_reading*, *Start_date* from **REPAIR** table where the *VIN* matches the *VIN* entered and *Completion_date* is null
- If no results are returned
    - Display **CUSTOMER** input panel
    - If **CUSTOMER** is added or selected
        - Display *Description*, and *Odometer_reading* input panels and ***Create Repair*** button
- Else if results are returned
    - Get the **CUSTOMER** for the **REPAIR** using the *Start_date* and *VIN* number of the **REPAIR** returned
        - If **CUSTOMER** is an **INDIVIDUAL PERSON**
            - Get customer *First_name* and *Last_name*
            - Display *First_name* and *Last_name*
        - If **CUSTOMER** is a **BUSINESS**
            - Get customer *Business_name*
            - Display *Business_name*
    - Get the *Quantity*, *Vendor_name*, *Part_name*, and *Price* for the list of **PART** that match the *Start_date* and *VIN* number of the **REPAIR** returned
        - For each **PART**
            - Display *Quantity*, *Vendor_name*, *Part_name*, and *Price*
    - Display **PART** input dialogs and ***Add Part*** button
    - Display *Labor_charges* in labor charge input and enable for editing
    - Display ***Update Labor Charges*** and ***Complete Repair*** buttons
    - Display *Description*, *Odometer_reading*, and *Start_date*

View/Generate Reports

Task Decomposition

Lock Types: Read-only **CUSTOMER**, **REPAIR**, **PART**, **VEHICLE**, **SALE**
No. of Locks: Several different schema constructs are needed.
Enabling Conditions: Log in, then request a report.
Frequency: Low, (ex: monthly reports) only by owner and manager.
Consistency (ACID): is not critical, read only, concerned with historical data.
Subtasks: Each individual report is considered a subtask. Each subtask can be done independent of others. Ex: generate a single report.



Abstract Code

- When selecting a certain report, a drop down menu opens, select the required report type.
- Upon:
- Clicking **Sales by Color**- Jump to **Sales by Color** task.
    - Perform Query of the **SALE** and **VEHICLE** records:
        - Filter only vehicles that have been sold.
        - Group by *Color*:
        - Count rows in each group/color where *Sale_date* < current date -30 (for the last month's sales)
            - rename col *Previous_30_days*
        - Count rows in each group/color where *Sale_date* < current date -365 (for the last year's sales)
            - rename col *Previous_year*
        - Count rows in each group/color for all time (for the last month's sales)
            - rename col *All_time*
        - For rows with more than one color:
            - relabel the list of colors as *Multiple* and add their results together
        - For vehicle colors that are not part of the report, the colors will be added to the report and their corresponding values set to 0.

- Clicking **Sales by Type**- Jump to **Sales by Type** task.
    - User (Manager/Roland) selects **Sales by Type** from the drop down **Generate/View reports** menu from **View Reports** form.
    - Run query on the **SALE, VEHICLE** tables
    - Using **VIN** as a join and Checking where sold price is not null
        - For each vehicle type (subtype):

        - Count all sale where *Sale_date* < current date -30 (for the last month's sales)
            - Display vehicle type and count for each type and if no row with sold price not null ,display 0.
        - Count all sale where *Sale_date* < current date -365 (for the last year's sales)
            - Display vehicle type and count for each and if no sale in the previous 365 days ,display 0.
        - Count all sales (for the last available sale date)
            - Display vehicle type and count for each and if no sale from the last available sale date ,display 0.

- Clicking **Sales by Manufacturer**- Jump **to Sales by Manufacturer** task.
    - User (Manager/Roland) selects **Sales by Manufacturer** from the drop down **Generate/View reports** menu from **View Reports** form.
    - Perform query on **SALE,VEHICLE** tables
    - Using *VIN* as a join and checking *Sale_price* is not null to get vehicles that have been sold.
    - for each *Manufacturer_name*:
        - count all sale where *Sale_date* < current date -30 (for the last month's sales), display *Manufacturer_name* and count
        - count all sale where *Sale_date* < current date -365 (for the last year's sales),display *Manufacturer_name* and count
        - count all sale (for the last available sale date) ,display manufacturer_name and count
- Clicking **Gross Customer Income**- Jump to **Gross Customer Income** task.
    - Perform query of the **SALE**, **VEHICLE,CUSTOMER** and **REPAIR** tables, order by gross income descending and last sale/repair data descending then get the top 15:
        - First, join **CUSTOMER**, **SALE**, and **REPAIR** tables on **CUSTOMER** key attributes.
            - Select *Name* and *Business_name, Sales_price, Sale_date, Start_date*, *Completion_date*, *Total_cost.*
            - Each customer will either have a sale, a repair, or both.
        - Group by **CUSTOMER** key attributes and for each customer/group:
            - include only repairs in progress (WHERE *Completion_date* = Null and *Start_date* is not Null)
            - Get first sale/repair
                - order ascending by *Sale_date* select first row *Sale_date* field
                - order ascending by *Start_date* and select first row *Start_date* field
            - Get most recent sale/repair
                - order descending by *Sale_date* select first row *Sale_date* field
                - order descending by *Start_date* and select first row *Start_date* field
            - number of sales/repairs:
                - Count rows of resulting sales

        - count rows of resulting repairs
    - Gross income:
        - Sum all vehicle sales
        - Sum all *Total_costs* for repairs
        - Add the sales and total repair costs together for each customer to get the total revenue from the customer

- Final fields in result: *Name* (full) or *Business_name*, date of first sale/repair start, date of most recent sale/repair start, number of sales/repairs, gross income (sales and total repairs costs combined)

- Clicking **Customer Name** on the Gross Customer Income report- Jump to **Customer Drill-Down** report.
    - Customer-Drill-Down Query:
        - Vehicle Sales part of the report will follow query:
            - Join **CUSTOMER, SALE PEOPLE , VEHICLE**
            - select *Name, Business_name*, *TIN*, Drivers_license_nr,
            - Select *VIN*, *Year*, *Manufacturer_name*, *Model_name*
            - select all where customers name == clicked Customer Name.
            - include fields: *Sale_date*, *Sale_price*, VIN, *Year*, *Manufacturer_name*, *Model_name*, and salesperson *Name*
            - order by Sale_date descending and by *VIN* ascending
        - Repairs part of the report will follow query:
            - Select *VIN, Start_date, Completion_date*, *Odometer_reading*, *Total_costs*, *Labor_costs*, Service Writer *Name*, Total_parts_cost = (Total_costs - Labor_charges)
            - include fields for each repair: *Start_date*, *Completion_date* (null if repair not finished), *VIN*, *Odometer_reading*, parts *Costs*, *Labor_charges*, *Total_cost*, service writer *Name*.
            - To fulfill this condition “incomplete repairs listed before complete ones with same sorting sort first by end date then start date. “order by *Completion_date* descending, *Start_date* descending, and *VIN* ascending.

- Clicking **Average Time in Inventory**- Jump to **Average Time in Inventory** task.

    - User (Manager/Roland) selects **Average Time in Inventory** from the drop down **Generate/View reports** menu.
    - Upon:
    - Clicking **Average Time in Inventory**- jump to **Average Time in Inventory** task.
    - Perform query on **SALE** and **VEHICLE** table for information about the average time a vehicle remains in inventory.
    - Perform left join on **VEHICLE** (left table) and **SALE** (right table) by VIN
    - Select columns: *Subtype (Vehicle_type), Inventory_date,* and *Sale_date* .
    - Calculate Time_in_Inventory by subtracting *Inventory_date* from *Sale_date*

- Group by Subtype and perform mean function on Time_in_Inventory to get Average_Time_in_Inventory.
- Replace Null Value in average time as “N/A”
    - Display the result

- Clicking **Parts Statistics**- Jump to **Parts Statistics** task.
    - User (Manager/Roland) selects **Parts Statistics** from the drop down **Generate/View reports** menu.
    - Upon:
    - Clicking Part Statistics- jump to Parts Statistics task.
    - Perform query on **PART** table for information about the parts.
    - Select Columns: *Vendor_name, Quantity and Price*
    - On **PART** table, group by *Vendor_name*
        - Calculate the Total_Number_of Parts by adding the Quantity
        - Calculate the Total_Dollar_Amount by multiplying the *Quantity* and the *Price*
        - Display the result

- Clicking **Repairs by Manufacturer/Type/Model-** Jump to **Repairs by Manufacturer/Type/Model** task.
- Clicking **Below Cost Sales**- Jump to **Below Cost Sales** task.
- Clicking **Monthly Sales**- Jump to **Monthly Sales** task.

Below Cost Sales

Task Decomposition

Lock Types: Read-only.
No of Locks: 4, Sale, Vehicle, Customer, and User schema constructs.
Enabling Conditions: User has selected Below Cost Sales report.
Frequency: Low.
Consistency (ACID): Not critical.
Subtasks: No subtasks are required and no mother task is required.

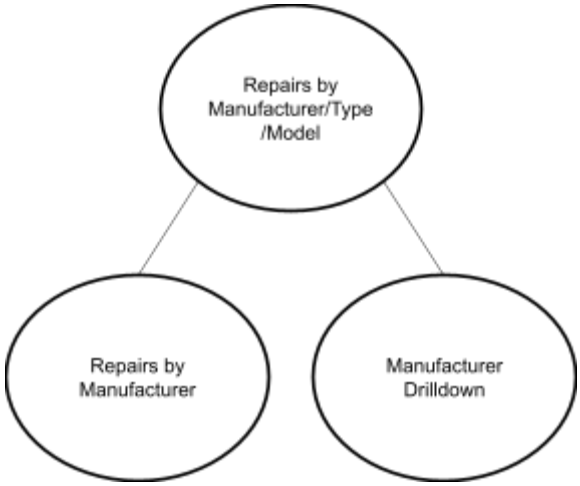
Abstract Code

- For each **SALE**
    - Get **VEHICLE** associated with the **SALE** by *VIN*
        - If the *Sale_price* is less than the *Invoice_price* of the **VEHICLE**
            - Get the **CUSTOMER** associated with each Sale
                - If **CUSTOMER** is an **INDIVIDUAL PERSON**
                    - Set $customerName equal to the concatenation of *First_name* and *Last_name*
                - If **CUSTOMER** is a **BUSINESS**
                    - Set $customerName equal to the *Business_name*
            - Get the **USER** associated with each **SALE**
            - Set $soldInvoiceRatio to *Sale_price* divided by *Invoice_price* times 100
            - Display *Invoice_price*, *Sale_price*, $soldInvoiceRatio, $customerName, and *Name*
            - If $soldInvoiceRatio is less than 95
                - Set background to red
- Order by *Sale_date* and $soldInvoiceRatio descending

Repairs by Manufacturer/Type/Model

Task Decomposition

Lock Types: Read-only.
No. of Locks: 3, Vehicle, Part, and Repair schema constructs.
Enabling Condition: User has selected Repairs by Manufacturer/Type/Model report.
Frequency: Low.
Consistency (ACID): Not critical.
Subtasks: Two subtasks are required, but no mother task is required.



Abstract Code

- Query **VEHICLE** and **REPAIR** table
    - Left join **REPAIR** and **PART** tables
    - Group by *VIN* and *Start_date*
        - Sum *Quantity* times *Price* as $partsCost
    - Left join **VEHICLE** and **REPAIR** tables
    - Join **VEHICLE** and **MANUFACTURER** tables
    - Group by *Manufacturer_name*
        - Count *Start_date* as $countRepairs
        - Sum *Labor_charges* as $allLaborCosts
        - Sum $partsCost as $allPartsCosts
        - Sum *Labor_charges* and $partsCost as $totalRepairCosts
    - Order by *Manufacturer_name* ascending
- For each result row
    - Display *Manufacturer_name*, $countRepairs, $allPartsCosts, $allLaborCosts, and $totalRepairCosts
- If user clicks on row (manufacturer drilldown)
    - Query **VEHICLE** and **REPAIR** table
        - Left join **REPAIR** and **PART** tables
        - Group by *VIN* and *Start_date*
            - Sum *Quantity* times *Price* as $partsCost
        - Join **VEHICLE** and **REPAIR** tables
        - Join **VEHICLE** and **MANUFACTURER** tables
        - Where *Manufacturer_name* equals the name on the row selected by the user
        - Group by *Vehicle_type*
            - Count *Start_date* as $countRepairs
            - Sum *Labor_charges* as $allLaborCosts
            - Sum $partsCost as $allPartsCosts
            - Sum *Labor_charges* and $partsCost as $totalRepairCosts

        - Order by $countRepairs descending
    - For each result row
        - Display *Vehicle_type*, $countRepairs, $allPartsCosts, $allLaborCosts, and $totalRepairCosts
        - Query **VEHICLE** and **REPAIR** table
            - Left join **REPAIR** and **PART** tables
            - Group by *VIN* and *Start_date*
                - Sum *Quantity* times *Price* as $partsCost
            - Join **VEHICLE** and **REPAIR** tables
            - Join **VEHICLE** and **MANUFACTURER** tables
            - Where *Manufacturer_name* equals the name on the row selected by the user and *Vehicle_type* equals name on the result row
            - Group by *Model_name*
                - Count *Start_date* as $countRepairs
                - Sum *Labor_charges* as $allLaborCosts
                - Sum $partsCost as $allPartsCosts
                - Sum *Labor_charges* and $partsCost as $totalRepairCosts
            - Order by $countRepairs descending
        - For each result row
            - Display *Model_name*, $countRepairs, $allPartsCosts, $allLaborCosts, and $totalRepairCosts

Monthly Sales

Task Decomposition

Lock Types: Read-only.
No. of Locks: 3, Sale, Vehicle, and User schema constructs.
Enabling Conditions: User selected report.
Frequency: High.
Consistency (ACID): Not critical.
Subtasks: Top SalesPerson.

Abstract Code

- Query the table **SALE** to get *Sale_Date*, and from the *Sale_Date* get the month and year of the sale, get also *Sale_Price* and *Username* from the table.
- Join to table **VEHICLE** using *VIN* to get the *Invoice_price* of the **VEHICLE**
- Subtract *Invoice_price* from *Sale_price* to get the $totalNetIncome
- Divide *Invoice_price* by *Sale_price* to get the $ratio and display as a percentage
    - COUNT(VIN) as $numberVehicleSold,
    - SUM(Sale_Price) as $totalSales,
    - SUM($totalNetIncome)
- Group by Year and Month
- Validate values in the form
    - If $ratio is greater than or equal to 125% show the row with a green background
    - If $ratio is less than or equal to 110% display the row with a yellow background
- Order the query by year and month descending, most recent year and month first
- When User clicks on a year monthly link display drill-down of the top-performing sales person
    - Create the drill-down Top Sales of each year month result as the top-performing salesperson
        - Using the same query add the *Username* to the SELECT statement and join to the table **USER** to get the User's name as *First_name* and *Last_ name*
        - Group by year, month, Name of the sales person
        - Order by $numberVehicleSold, $totalSales DESC