CPSC 3660 - Spring 2018

Introduction to Database Systems

Course Project Summary

Nora White, Matthew Rose, Ryan Lockett

User requirements

Business overview and objectives

Westside Auto Inc. is a local used-car dealership located in West Lethbridge. They have built up enough business that they now want to move their paper tracking system to an online database that can be used on site, or remotely. The website must be able to track purchases made by buyers and allow salespeople to sell vehicles to customers with warranties.

Project overview

The client

The Westside Auto Inc. website will be based on the specifications provided in the overview document given to us in our CPSC 3660 class. We have no further contact with the client and will be using assumptions made to guide us through the development process.

The problem

Westside Auto Inc. has requested to move their paper tracking system to an online source. The project scope will include developing the website to handle the original forms that were filled out by hand, as well as the added task of creating predefined data that can be selected from to aid in the purchase and sale of vehicles, as well as maintain consistency throughout the data.

Project scope

1. Purchase vehicles through buyers
   Buyers are able to log the vehicles that they have purchased for the dealership and any repairs that they assume are needed for the vehicle.

2. Sell vehicles to customers
   Salespeople are able to enter customer data, choose a vehicle to sell to the given customer, create warranties for that vehicle, and provide information on the total cost and financing of the vehicle.

3. Be able to use this system online
   Buyers need to be able to access the system remotely, so this project needs to be created for web access.

System requirements

The first stage that we progressed through was the requirements gathering stage. As we read through the project overview document, we gathered data on what we understood and what we didn't understand about what the company does and how the data interacts with other data. We made assumptions based on data that we didn't understand, and came up with use cases based on the system goals that we found in the project overview document given to us.

System goals

The following are the system goals that the new website for Westside Auto Inc. will need to handle:

1. Create new employees (buyers and salespeople)
2. Create new items that can be chosen for warranty
3. Purchase a vehicle (or multiple vehicles)
    a. All vehicles can have multiple repairs that are recorded at the point of purchase
4. View and edit vehicles and repairs
5. Sell a vehicle that exists in the database to a new customer with warranties
6. View and edit customers
    a. See vehicles that they have purchased
    b. See their payments (and add payments)
    c. See their employment history (and add employment history)

Assumptions

1. A buyer can purchase more than 1 vehicle at a time (from one seller)
    a. Reason: The buyer can find many low priced cars at the same auction.
2. A vehicle can have multiple repairs or none
    a. Reason: multiple things can be repaired on the same car, as well as a car may not need repairs to be made.
3. A repair will never be repeated exactly the same as in previous jobs.
    a. Reason: the estimated repair cost and actual cost will never be repeated exactly.
4. The commission that a salesperson makes is sales dependent and varies sale to sale.
    a. Reason: Sometimes an employee lists a lower commission in order to help sell the car faster.
5. A person will never be a customer unless they have at least one sale.
    a. Reason: There would be no need to hold data on a customer who hasn't purchased a car or any warranty plan.
6. A customer can only buy one vehicle at a time.
    a. Reason: Each form of payment is dependent on the single car they purchased, they would need to create a new sale form in order to purchase a new vehicle.

7. A customer does not need to have an employment history to buy a vehicle.
    a. Reason: Any customer can buy a vehicle.
8. A customer will always have at least one payment on file because of the requirement to have a down payment.
    a. Reason: A customer must have a way of payment whether it by all up front or as a monthly finance that will accumulate interest.
9. Commission is added on top of the sale price, so total cost is calculated at listing price + commission + warranties.
    a. Reason: Because commission can change from sale to sale and it provides an incentive for the salesperson to sell more warranties.
10. Finance amount is calculated based off of total due minus the down payment.
    a. Reason: In order to make the sale, the company needs a guarantee that the person will pay back the vehicle. The purpose of financing is to have that guarantee. Financing can have interest but in this case, it does not.

Use cases (forms)

The main forms are:

1. Create new buyer
2. Create new salesperson
3. Create new warranty item
4. Create new purchase
5. Create new sale

Secondary forms are:

1. Update customer information
2. Add new payment history
3. Add new employment history
4. Update vehicle information
5. Update repair information

Requirements models

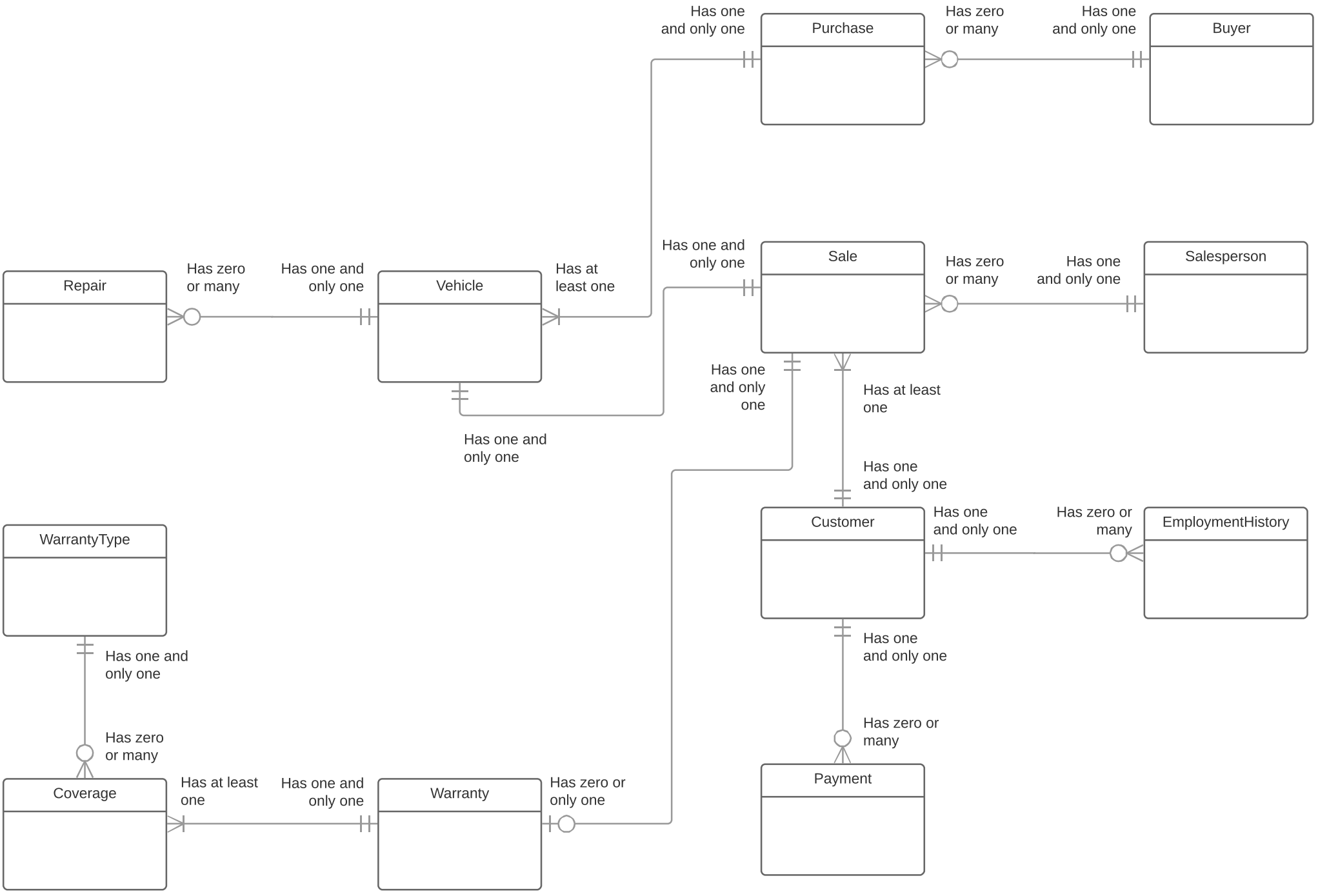
Using the data that we gathered in the requirements gathering stage, we compiled rough ideas of what our relations would be, and what attributes they would need. We then completed the conceptual ERD based on our findings.

After completing the conceptual ERD, we moved on to the logical ERD, and began to add to our relations. Through this process we learned where tables were and were not needed, and where redundant data would be created.

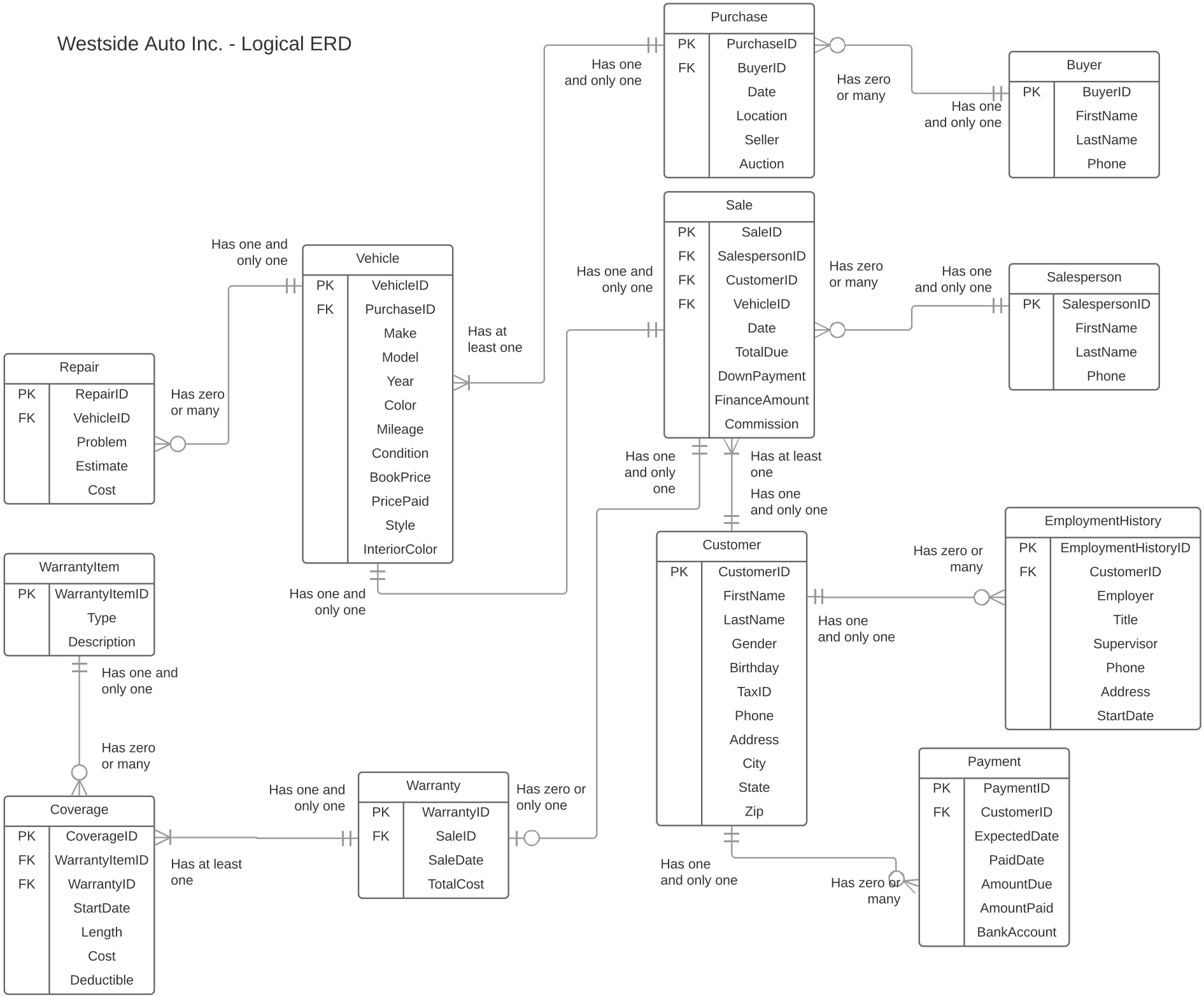
Immediately after developing the logical ERD, we added data types and defined primary and foreign keys to our attributes, pushing us into the physical ERD.

We then used the physical ERD as a guideline for what kind of interactions we would need with the data. We understood which pages we needed based on what forms were already in existence on paper, but we also needed a way to get the data into those forms before hand. We came up with a list of all use cases we would need to implement, which would become our completed forms on the website.

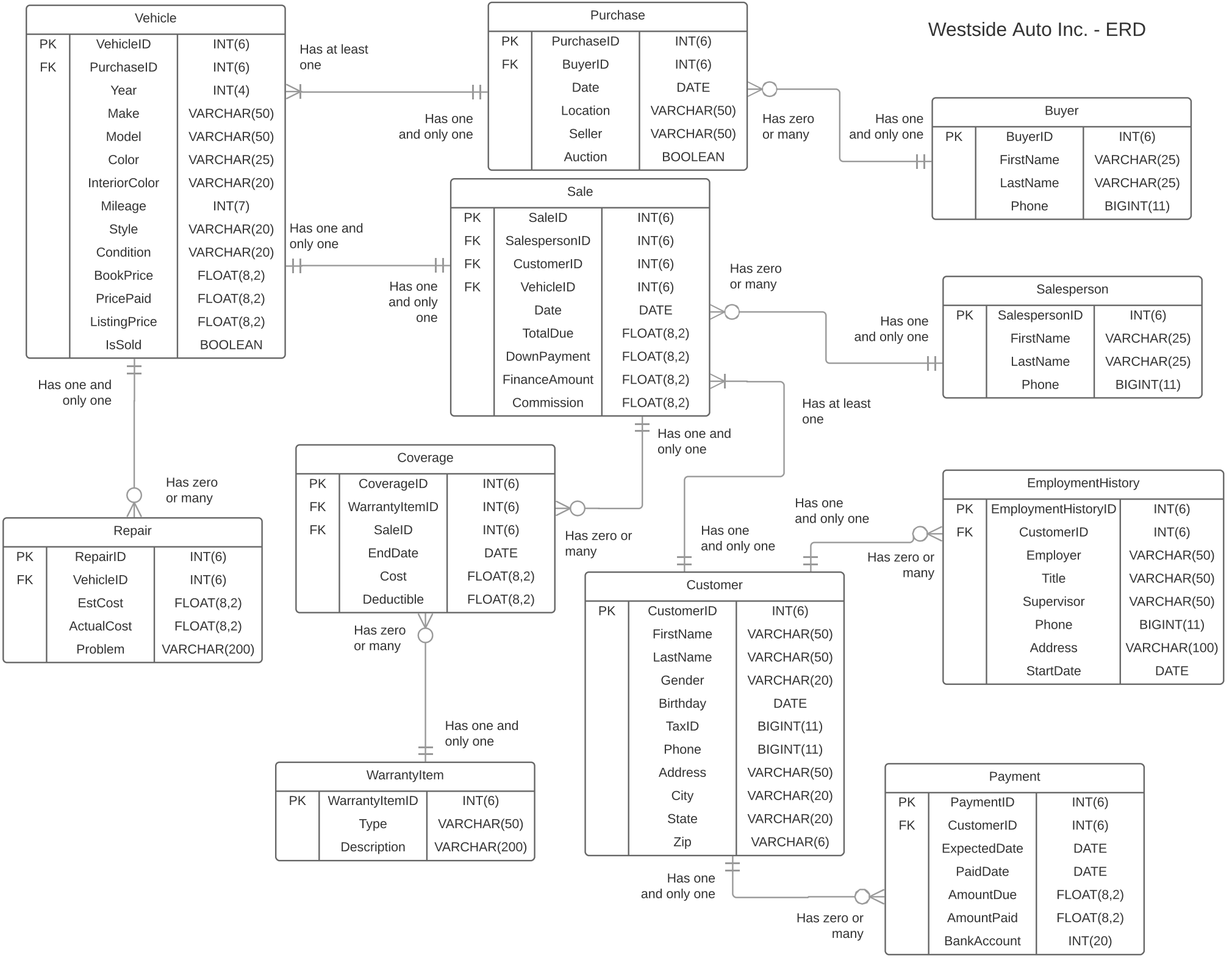
Westside Auto Inc. - Conceptual ERD



Westside Auto Inc. - Logical ERD



Westside Auto Inc. - ERD



Implementation

The first step in implementation was to create a template web page with all style guidelines completed and the basic structure of the website broken out. We duplicated this page to give us all of the pages that we have now.

We began working on the simplest forms (the forms without the need of foreign keys), so that when we reached the more involved forms such as purchasing a vehicle, selling a vehicle, and viewing customer information, we could easily add sample data via the web UI.

After completing the simple web pages, we moved on to the most involved forms and updated our database and physical ERD as needed.

Form overview

Customer form

An employee can search for the specific customer through a table on the customer page. Once a customer is selected for edit, all user information is displayed and can be updated. This page can be directed to the payment form and employment history form, both of which are used to add additional information (apart from general customer information) to the customer. This page also displays the vehicles that a customer has purchased, with a link to the vehicle form page.

Employment history form

The purpose of this form is to add any previous employment history for the given customer.

Payment form

The purpose of this form is to add a new payment entry that the customer makes. Entries include expected date, paid date, amount due, amount paid, and the account number from the bank that is making payments.

Buyer form

The purpose of the buyer form is to create a new buyer employee that will hold information such as: first name, last name, and phone number. Their ID number is auto-generated at the point of insertion into the database.

Purchase form

The purchase form is to be used for when a buyer is travelling and makes a purchase of a vehicle (or vehicles). The following information is captured through the form: the buyer's ID, the date the car was purchased on, the vehicle's seller's name, and the location at which the vehicle was purchased at. We then capture the vehicle information of: year, make model, style, condition, mileage, color, interior color, book price, and price paid. The book price is the official evaluated price of a used vehicle that is used as a starting point for negotiation. The price paid is the price that was negotiated to be paid for the vehicle. It is company policy that Westside Auto will not pay for a vehicle that is more than the book price. There are also repair costs that may need to be accounted for along with the description of the repair that will be taken in. You can also add more repairs with the vehicle or add more vehicles.

Sale form

The sale form is used as intake for when a customer purchases a vehicle. This form handles the creation of sale, customer, and coverage instances. The salesperson's ID is captured along with all customer information such as: first name, last name, gender, birthday, tax ID, phone, address, city, state, and zip code. An auto generated CustomerID will be assigned to the customer on insert into the database. On this form we've also displayed all vehicles that are listed as for sale: vehicles only appear in the selection table when they have been given a listing price, and if their "IsSold" field in the database is currently false (it sets to true once the sale has been made). The salesperson can also include warranties that offer different coverage for the vehicle they sell. The information captured is cost, end date, and deductible. Total cost is calculated as the sum of the listing price, warranty costs, and commission. Finance amount is calculated from the total cost minus the down payment.

Salesperson form

The purpose of this form is to add new salespeople to the database that can be used when a vehicle is being sold to a customer. The following information is captured: first name, last name and phone number.

Warranty item form

The purpose of this form is to add any new coverage on items, such as leather interior or new sound system. They can add more variety of what can be replaced as time goes on. The form will take in the type of warranty it is and the description of it. A warranty ID number will also be generated when a new warranty is added to the system.

Queries

NOTE: All " ? " within the SQL statements refer to values being inserted through a prepared statement, or other variables set in PHP.

Note: All rows in BLUE are advanced queries.

buyer.php	
Insert a Buyer:	`INSERT INTO Buyer (FirstName, LastName, Phone)` `  VALUES (?, ?, ?);`

warrantyitem.php	
Insert into WarrantyItem:	`INSERT INTO WarrantyItem (Type, Description)` `  VALUES (?, ?);`

salesperson.php	
Insert into Salesperson: This creates a salesperson so that when a customer purchases a vehicle, a salesperson can select their own name from a dropdown list to associate with that purchase.	`INSERT INTO Salesperson (FirstName, LastName, Phone)` `  VALUES (?, ?, ?);`

sale.php

Select Salesperson:
 	
Gets all information for a salesperson so that a dropdown box can be populated.
	`SELECT SalespersonID, FirstName, LastName`
	`  FROM Salesperson`
	` ORDER BY LastName;`

Insert into Customer:
 	
Creates a new customer to satisfy the assumption of a customer only existing if they also have a sale (this is also why a customer cannot be added on the customer.php page).
	`INSERT INTO Customer (FirstName, LastName, Gender,`
	`            Birthday, TaxID, Phone, Address,`
	`            City, State, Zip)`
	`     VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?);`

Insert into Coverage:
 	
Creates a new coverage for the vehicle.
	`INSERT INTO Coverage (SaleID, WarrantyID, EndDate, Cost,`
	`            Deductible)`
	`     VALUES ((SELECT MAX(SaleID) AS SaleID`
	`       FROM Sale), ?, ?, ?, ?);`

Insert into Sale:
 	
Inserts a new sale item, which is then used to create new coverages. Sale grabs the max customer ID, as the max customer ID is always the customer that is being created specifically for this sale.
	`INSERT INTO Sale (CustomerID, VehicleID, SalespersonID,`
	`                 Commission, DownPayment, FinanceAmount,`
	`                 TotalDue, Date)`
	`     VALUES ((SELECT MAX(CustomerID) AS CustomerID FROM`
	`              Customer), ?, ?, ?, ?, ?, ?, ?);`

Select from Vehicle:
 	
Generates a list of vehicles that are available to be sold. IsSold allows the PHP to filter vehicles and only show vehicles that have not been sold. ListingPrice allows the PHP to filter vehicles that have not been sold but now *can* be sold because they have been given a listing price.
	`SELECT VehicleID, Make, Model, Year, Color, Mileage, Style,`
	`       InteriorColor, IsSold, ListingPrice`
	`  FROM Vehicle`
	` WHERE IsSold = 0`
	`   AND ListingPrice IS NOT NULL;`

Select from WarrantyItem:
 	
Creates a new warranty item from which a salesperson can generate coverage from when they are selling a vehicle to a customer.
	`SELECT WarrantyItemID, Type`
	`  FROM WarrantyItem`
	` ORDER BY Type;`

vehicle.php

Update a Vehicle:
 	
Updates a vehicle for when a salesman is ready to sell the vehicle incase errors were made.
	`UPDATE Vehicle`
	`  SET Make=?, Model=?, Year=?, Style=?, Color=?,`
	`  InteriorColor=?, Mileage=?, `Condition`=?,`
	`  BookPrice=?, PricePaid=?, ListingPrice=?`
	`WHERE VehicleID=?;`

Update a Repair:
 	
Updates a repair for the corresponding vehicle. This is necessary for when the repair has been completed and they need to add in the "actual cost" of the repair.
	`UPDATE Repair`
	`  SET ActualCost=?, Problem=?`
	`WHERE RepairID=? AND VehicleID=?;`

View a list of Vehicles:
 	
This allows a user to view a condensed list of vehicles from which to choose to edit from.
	`SELECT VehicleID, Make, Model, Year,`
	`  Color, Mileage, Style, InteriorColor`
	`FROM Vehicle;`

getVehicle.php

Select Buyer and Purchase Info: Used to get the Buyer information and Purchase information for a specific vehicle.	`SELECT * FROM Vehicle NATURAL JOIN Purchase NATURAL JOIN Buyer WHERE VehicleID = ?;`
Select all Repairs: Gets all the repairs for a specific vehicle.	`SELECT * FROM Repair WHERE VehicleID = ?;`

Select Vehicle Coverage:
 	
Get all the coverage information for a specific vehicle
	`SELECT *`
	`FROM coverage NATURAL JOIN warrantyitem`
	`WHERE SaleID = (SELECT SaleID`
	`FROM sale`
	`WHERE VehicleID = ?)`

purchase.php

Insert into Purchase: This creates a purchase record when for when a buyer purchases a vehicle.	`INSERT INTO Purchase (BuyerID, Date, IsAuction, Seller, Location)` `VALUES (?, ?, ?, ?, ?);`
Insert into Vehicle: This creates a record for the vehicle that is purchased by the buyer.	`INSERT INTO Vehicle (PurchaseID, Make, Model, Year, Color, Mileage, ``Condition``, BookPrice, PricePaid, Style, InteriorColor, IsSold)` `VALUES ((SELECT MAX(PurchaseID) AS PurchaseID FROM Purchase), ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, 0);`
Insert into Repair: This creates a record of the repairs needed for a vehicle purchased by the buyer.	`INSERT INTO Repair (VehicleID, EstCost, Problem)` `VALUES ((SELECT MAX(VehicleID) AS VehicleID FROM Vehicle), ?, ?);`

Select a Buyer:
 	
This allows the buyer to indicate who is purchasing the vehicles on the form.
	`SELECT BuyerID, FirstName, LastName`
	`FROM Buyer`
	`ORDER BY LastName;`

customer.php

Update a Customer:
 	
Updates a customer's information incase they moved addresses, changed last name, changed phone numbers, or there was an error when entering information at the sale stage.
	`UPDATE Customer`
	`SET FirstName=?, LastName=?, Gender=?, Birthday=?, TaxID=?, Phone=?, Address=?, City=?, State=?, Zip=?`
	`WHERE CustomerID=?;`

Select from Customer: Gets a condensed list of all customers from which a user can pick which one to view and edit.	`SELECT CustomerID, FirstName, LastName, Gender, Phone` `FROM Customer;`

getcustomer.php

Select a Customer:
 	
Get all information for a selected customer.
	`SELECT *`
	`  FROM Customer`
	`WHERE CustomerID = ?;`

Select all Vehicles for a Customer:
 	
Gets all vehicles that the selected customer has purchased.
	`SELECT VehicleID, Make, Model, Year, Color,`
	`       Mileage, Style, InteriorColor`
	`  FROM Vehicle`
	`WHERE VehicleID IN (SELECT VehicleID`
	`                        FROM Sale`
	`                       WHERE CustomerID = ?);`

Select Avg. Number of Days Late:
 	
Gets the average number of days late a specific customers payments are.
	`SELECT ROUND(AVG(DateDifference),2) AS AvgDateDifference`
	`  FROM (SELECT DATEDIFF(PaidDate, ExpectedDate) AS`
	`              DateDifference`
	`          FROM Payment`
	`         WHERE CustomerID = ?) AS DateDifferences;`

Select Number of Late Payments:
 	
Gets the number of late payments that a specific customer has.
	`SELECT COUNT(DateDifference) AS NoLatePayments`
	`  FROM (SELECT DATEDIFF(PaidDate, ExpectedDate) AS`
	`              DateDifference`
	`          FROM Payment WHERE CustomerID = ?) AS`
	`              DateDifferences`
	`WHERE DateDifference > 0;`

Select all Customer Payments:
 	
Gets all the payments that a specific customer has, populates a table,, and shows the newest payment first.
	`SELECT *`
	`  FROM Payment`
	`WHERE CustomerID = ?`
	`ORDER BY PaymentID DESC;`

Select Employment History:
 	
Gets the employment history for a specific customer and populates a table.
	`SELECT *`
	`  FROM EmploymentHistory`
	`WHERE CustomerID = ?`
	`ORDER BY EmploymentHistoryID DESC;`

customer/newemploymenthistory.php

Select from Customer:
 	
This grabs the first name and last name of the corresponding customer so that the user knows which customer they are adding a payment to. This aids in usability.
	`SELECT FirstName, LastName`
	`  FROM Customer`
	` WHERE CustomerID = ?;`

Insert into EmploymentHistory:
 	
Creates a new EmploymentHistory so that administration can maintain a history of all employment history for the customer.
	`INSERT INTO EmploymentHistory (CustomerID, Employer, Title,`
	`            Supervisor, Phone, Address, StartDate)`
	`     VALUES (?, ?, ?, ?, ?, ?, ?);`

customer/newpayment.php

Select from Customer:
 	
This grabs the first name and last name of the corresponding customer so that the user knows which customer they are adding a payment to. This aids in usability.
	`SELECT FirstName, LastName`
	`  FROM Customer`
	` WHERE CustomerID = ?;`

Insert into Payment:
 	
Creates a new Payment so that administration can maintain a history of all payments for the customer.
	`INSERT INTO Payment (CustomerID, ExpectedDate, PaidDate,`
	`            AmountDue, AmountPaid, BankAccount)`
	`     VALUES (?, ?, ?, ?, ?, ?);`

Post mortem

Changes we could make in the future

1. On the vehicle page, if a vehicle has been sold, we would no longer be able to change any fields on it.
2. Add a screen for adding a new repair to a vehicle after the vehicle has been purchase (via the Vehicle page)
3. On the sale screen, when you click on a warranty item, it then populates a div directly below it with the description of said warranty item.
4. As per known deficiency 1.a, adding in an option for amortization period, and then an onchange function to calculate the monthly payments over that amortization period.
    a. On the customer page under the payment section, having a disabled field to show the total remaining amount on the account (finance amount minus sum of all payments), and have the expected date and amount due auto filled based on the amortization period and finance amount.
5. On the sale page, being able to add the first down payment information would be nice.

What we learned from this project:

1. A normalized database will save headaches, going from conceptual to logical to physical made the design process much easier.
2. Having the database running right away made the implementation of php files very easy.
3. Starting with pages that require no foreign keys made the design process very natural.
4. Data types are very important.
5. It is very difficult to understand how some data interacts with others when you don't have a person to speak to. There is only so much that assumptions can do, but when you completely do not understand how one value is important or what it means in terms of business logic, it makes making the data useful very difficult.
6. Don't use LucidChart to create ERDs, use draw.io
7. Mixing javascript, jQuery, and PHP make for complex files.

Known deficiencies

1. There is no way to identify the monthly payment or amount still owed to the company.
    a. This deficiency was ignored due to time constraints and the original overlooking of the need to have a finance amortization period. Amortization period was never shown on any of the forms.
2. Further information is needed about the relationship between deductible, end date, and cost of warranty. The definition of deductible is not understood, and so deductible serves no purpose other than holding data.