Name: Kokil Dhakal

**Table of Contents**

[Project Direction Overview 2](#_Toc72234346)

[Use Cases and Fields 2](#_Toc72234347)

[Structural Database Rules 6](#_Toc72234348)

[Conceptual Entity-Relationship Diagram 7](#_Toc72234349)

[Initial DBMS Physical ERD 7](#_Toc72234350)

[Summary and Reflection 8](#_Toc72234351)

# Project Direction Overview

I would like to design a database for a car dealership company “CarMaxDeal” which will have their own website. This company uses an application that can access database to track all the cars they have for sell. This company also tracks inventory of the car like make, model, year, color etc. of the cars. Similarly, they must track of all sales information like sale date, sold car type, salesperson, customer to whom it was sold etc. Company also tracks the customers and employee who were involved in the transaction for a specific car.

A customer can check all the available cars and their features as well as their price using company web site. While an employee can check the status of car, price and initiate the transaction if a customer likes it and want to buy.

Here are some brief examples of how someone can use the database. Customer uses company’s website to check all the available cars, colors, make, model, and all other features etc. Customer can also check price of car. if they like it, they go nearby company’s dealer. If a customer decided to buy, employee check availability and initiate transaction and customer buy car.

Database stores information about cars’ inventory, sales information, customers information, Employee information, information about the car status, information about car features and information about transactions.

# Use Cases and Fields

Update, if necessary, the five or more use cases that enumerate steps of how the database will be typically used and identify significant database fields needed to support each use case.

**Vehicles receiving from vender use case:**

Managing the inventory of new and used vehicles would includes

1. Dealership order/receives vehicles from Venders
2. This cause addition of new or used vehicles in the database inventory.

Significant fields for this use case are listed below

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it’s needed |
| VehicleID | Vehicle identification number generated by dealerships ‘application. | This is needed to track the vehicle. This uniquely identify the specific vehicle |
| Make | This field store Car company | This is needed to track the vehicle’s company |
| Model | This field stores Car model | This is needed to know what kind of model of car in the inventory |
| Color | This field Car color | This is needed to know what kind of color of car in the inventory |
| Year | This field stores Car makes year | This is needed to know what year of car in the inventory. |
| VIN | Vehicle Identification Number of the car. | This is needed to uniquely identify car |
| VenderID | Stores the type of service getting from venders | To track whether it is receiving used car, new car, and part for its inventory. |
| Condition | Condition of car received | Track whether it is new or used |
| ReceivedDate | Car received data | Track car received date |
| DealershipID | It stores which dealership sold a particular vehicle | Track the dealership and vehicle |
| Market\_price | It stores price decided by dealership | Needed to track MP of vehicle |

**Vender use case:**

1. When company receives car from vender its database updated with new car
2. When company sell a car, its database updated with one less car in inventory.

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it is needed |
| VenderID | Vender Identification Number | Uniquely identify vender |
| VenderName | Vender name | To track Vender |
| VenderAddress | Vender Address | To track the Vender address |
| PhoneNumber | Vender Phone number | To track the vender phone |

**Tracking sales use case**

Tracking sell is important for the company to know what kind of car more frequently selling and which dealer is doing best and which need an improvement.

1. Employee long in their account and check for car of customer choice in the system
2. Employees sell the car to customers
3. Selling one car does delete one entity instance from the car inventory.

Significant fields for this use case are

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it’s needed |
| Date of sale | The date of care sale | Needed to track when car was sold for report analysis |
| CustomerID | Customer to whom car was sold | Need to track the customer to whom car was sold |
| VehicleID | Type of vehicle it sold | This tracks type of vehicle sold |
| EmployeeID | Employee who involved in selling car (Sale person) | This tracks Employee involved in transaction of the car |
| Transaction ID | Type of Trasaction | This field tracks the price at which vehicle was sold. It is needed for report generation. |
| SaleID | It stores sale Identification number | It is needed to track sales |
| Sale\_Price | It stores sale price of car | It is needed to track sold price of car. |

**Employee hire/leave use case**

This use case required ability of database to store employee information. When new employee is hire, their information are stores in the database. Significant field are given below.

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it’s Needed |
| EmployeeID | It Stores the Employee identification number | Needed to track the Employee uniquely identify the Employee. |
| FirstName | This is First name of Employee | Need to display Employee who is involved in dealing with customer |
| LastName | This is last name of Employee | Need to display Employee who is involved in dealing with customer |
| Salary | This is salary of employee | This field tracks the salary of the employee. This is needed to determine total cost of the company. |
| Position | This position of employee | Needed for cost analysis |
| DOB | date of birth of employee | Needed to identify employee, also to know how long employee being employed in current company |
| Address | Address of employee | To identify employee, another means of communication via mail. |
| DepartmentID | It stores employee’s department | To track employee and department |

**Customer Tracking use case:**

Managing the relationship with the customers, including storing customer information, tracking customer interactions and purchase history, and providing customer service. This use case would require the ability to store customer information, track customer interactions, and provide customer service. Whenever customer come in to buy car, Salesman check whether customer is new to dealerships, if not new, they verify the customer information if customer is new, new record will be added to database. Significant fields are given below.

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it is Needed |
| CustomerID | Customer identification number which is unique | Uniquely identify the customer |
| LastName | Customer Last name | This is needed to address a customer while communicating with them. |
| FirstName | Customer First name | This is needed to address a customer while communicating with them |
| PhoneNumber | Customer Phone number | This is needed to address a customer while communicating via text or call with them |
| Email Address | Customer email address | This is needed to address a customer while communicating via email with them |
| VehicleID | It stores type of services customer received | This is needed to track the type of services receiving by customer |

**Transaction use case:**

When doing transaction whether it is for selling or buying car , company must use database to get information like vehicleID, customerID for transaction purpose

Significant fields are

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it is needed |
| TransactionID | Uniquely identifies payment type | Needed to track payment type |
| CustomerID | Identify customer | To track who is paying |
| SaleID | Identify vehicle for purchase for parts replacement | To track for what payment if for |
| PaymentID | Type of payment used | To track type of payment |
| EmployeeID | It stores employee involved in transaction | To track the employee who initiate transaction |

**Dealership use Case:**

- Dealership sells vehicle which directly interact with the company database.

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it is needed |
| DealershipID | Uniquely identify dealership | To track dealership |
| Name\_dealership | Name of dealership | To track dealership |
| Address\_dealership | Address of dealership | To locate dealership |
| Phone\_dealership | Phone number of dealership | Dealership info |
| VenderID | IT stores the vender info | To track Vender |

**Payment uses case:**

1. When Employee initiates transaction, payment in involved
2. We need database to get some information about the transaction.

Significant field involved are:

|  |  |  |
| --- | --- | --- |
| Field | What it stores | Why it is needed |
| PaymentID | It stores unique payment identification number | To uniquely identify each payment |
| TransactionID | It stores the transaction information | To track each transaction that involved payment |
| CustomerID | It stores customer involved in payment | To track customer who invoved in payment |
| Date of payment | Data of payment | Track the date of payment |
| Payment\_amout | Total payment price | To track total amount of price in each payment. |

# Structural Database Rules:

From the above use cases, following entities can be identified

|  |  |  |
| --- | --- | --- |
| Vehicle | Vender | Customer |
| Employee | Department | Sale |
| Payment | Dealership | Transaction |

1. Each **Dealership has many vehicles**; each **vehicle** has a dealership
2. Each **vehicle** may sell to a customer; Each **customer** may buy many vehicles.
3. Each dealership may have many parts; each part (unique serial number) has a dealership
4. Each **vehicle** may sell by a dealership; each **dealership** may sell many vehicles.
5. Each **part** is sold by a dealership; each **dealership** may sell many parts.
6. Each **dealership** may have many departments; each **department** has a dealership.
7. Each **department** may provide many services to customers; and each **service is** received from a department
8. Each **department** has many employees; each **employee** has its own department.
9. Each **employee** may initiate a transaction at a time; each **transaction** is initiated by an employee.
10. Each **customer** may involve many transactions: each **transaction** is from a customer.
11. Each **transaction** has one payment; Each **payment** has one transaction
12. Each **Vender** may have many dealerships; Each **dealership** may have many Venders(M:N)

For specialization-generalization, I have found two entities that need specialization. Those are Vehicle and Employee. Hence, I have included two structural rules based on this.

13.A vehicle is Sedan, SUV, Hatchback, or None of these.

14. An Employee is Mechanic, Salesperson, or none of these.

15. a department is service, sales, HR, or none of these.

# Conceptual Entity-Relationship Diagram

Diagram

Description automatically generated

# Initial DBMS Physical ERD

The Physical ERD I made which can be seen below chart. I use some basic rules for assigning Primary key and foreign key for each entity. In case of super type and subtype entities, sub type entities have its primary key and foreign key from its super type entities. For example, Sedan, SUV, Hatchback have its primary key and foreign key from its super type of Vehicle which is VehicleID. Also, there might be more than just these 3 vehicle types that is why it is partially complete which is denoted by “optional” inside Curley brackets. Also, sedan is not SUV, or it is not Hatchback or SUV is not Hatchback or vice versa which means these subtypes are disjoint which is denoted by “or” inside Curley brackets in UML and same goes with Department and Employee supertypes and their subtypes.

On the other hand, each entity will have its own primary key except subtypes. Since vender has many to many relations with dealership and we cannot map many to many relations as it affects data integrity, I use a vender\_dealership bridging entity. I use the association relationship to assign which entities get whose primary key for its foreign key. For example, if two entities have one to many relations as in “Dealership and vehicle”, entities with many side will have foreign key from one side. In this relation, dealership is one side (0..1) wile vehicle is many side (0..\*) that is why vehicle entity has primary key of dealership for its foreign key. This same rule goes to all the entities which have one to many relations. However, in case of one to one relation, both entity is suitable. For example, in case of ‘payment and transaction’, primary key of any one entity is suitable for foreign key for other entity. In my case, primary key of transaction is used as foreign for payment entity.

Diagram, engineering drawing

Description automatically generated

# Summary and Reflection

My database for car dealership named CarMaxDeal” which stores information about cars inventory, Sales information, Customer information, Service/maintenance information, Employee information, information about the service type customer getting. Customer can use web site to check all available cars and its features to buy. Dealership has their own application to track all the activities involved in car purchase or servicing a car at the dealerships. Database must support a person using website and searching for their favorite car to buy. Also, Database must support generating report for analysis for example daily, weekly, monthly, and yearly report analysis.

In this Iteration2, to make more entities I have added some use cases for my database design. I came up with 14 business rules and I have mentioned 10 entities, their relationships and plurality. I did make ERD for all those entities and their relationships.

In this Interaction 3, goal of this iteration is to convert conceptual ERD to DBMS Physical. There are three entities which need specialization-generalization representation namely Vehicle, Department and Employee. Each of these entities act as supertype entities which is also known as abstract entity and each of these have their own subtypes which is also called as specialized entities. Once supertype entity is identified, I made a specialized-generalized ERD also known as EERD. Next step is to make DBMS physical ERD. This is based on one to one, one to many and many to many mapping properties. Appropriate mapping is done to avoid data inconsistencies, data redundancies and/or other type of data anomalies.