FINAL REPORT

Vehicle Management Centre Database

Alfie Atkinson

2023

# Contents

**Introduction Page 2**

**Requirements and Objectives Page 2**

**Design and Implementation Page 3**

Entity Relationship Diagram Page 3

**Database Creation and Tables Page 4**

Vehicles Table Explanation Page 4

Departments Table Explanation Page 5

Faculty Table Explanation Page 6

All other tables Page 7

**Transactions and Reports Page 14**

Display all vehicles on a given date Page 14

Display how many vehicles each department has used so far Page 15

Display the total mileage driven by a department in a year Page 16

Show details of a particular bill Page 24

Display those who booked vehicles but didn’t use them Page 26

Extras Page 28

**Triggers Page 31**

Update parts inventory Page 31

Restock part Page 32

Update vehicle odometer Page 33

Updating vehicle availability Page 34

**Conclusion Page 37**

# Introduction

The purpose of the following report is to document the design and implementation of a database for the New Century University's (NCU) Vehicle Management Centre (VMC). The report will provide an overview of the requirements and objectives of the VMC, the design and structure of the database, as well as the various queries and procedures that were developed to support the VMC's operations. This report will also include screenshots and explanations of the database's tables, relationships, and triggers. Furthermore, it will cover the testing and validation of the database, and the challenges faced during the development process.

# Requirements and Objectives

The current manual system for managing vehicles at NCU is prone to errors and inefficiencies, making it difficult to track and report on usage and maintenance of the university's fleet. The implementation of a database system for the VMC is crucial in addressing these issues and ensuring efficient operation. The objective of the database is to automate the reservation, sign-out, and trip completion forms as well as track usage, maintenance, and inventory of the vehicles. This will allow for easier tracking of vehicle usage by department and faculty member, efficient scheduling of maintenance, and accurate billing for fuel and mileage. Additionally, the database will provide a comprehensive set of reports to allow for better decision-making and management of the VMC. Overall, the goal of the database is to improve the efficiency and effectiveness of the VMC by providing real-time data and insights.

Objectives:

* To design and implement a database system for the NCU VMC that streamlines the vehicle management process.
* To automate the reservation, sign-out, and trip completion forms, as well as tracking the usage, maintenance, and inventory of the vehicles.
* To provide an easy way to track vehicle usage by department and faculty member, schedule maintenance, and bill for fuel and mileage.
* To provide a comprehensive set of reports on vehicle usage and maintenance to allow for better decision-making and management of the VMC.
* To improve the efficiency and effectiveness of the VMC by providing real-time data and insights into the usage and maintenance of the NCU's vehicles.
* To develop procedures that support the VMC's operations.
* To develop views to provide an easy way to see available vehicles, and mileage by vehicle and faculty member for the last month.
* To develop triggers to update the parts inventory when parts are used, update the odometer reading when a trip is completed, and update vehicle availability when it is signed out or goes in for maintenance.

# Design and Implementation

The design and implementation of the database for the NCU’s VMC was approached with a focus on data integrity and efficient data management. This was achieved using ER diagrams and database normalization techniques.

Timeline

Description automatically generated

This ER diagram for the database was created to visually represent the relationships between the different entities, such as vehicles, faculty members, and departments. It served as a blueprint for the structure of the database and helped to ensure that the data was organized in a logical and consistent manner.

In the database, I applied normalization by breaking down the data into smaller, more manageable tables and establishing relationships between them. For example, in the vehicles table, the vehicle\_registration field is used as the primary key and is referenced in the foreign key of the sign\_out\_forms table. This ensures that the data is stored in a consistent format and eliminates any data redundancy. Another example of normalization can be seen in the faculty\_members and departments tables, where the faculty\_member\_department\_id field in the faculty\_members table is used to establish a relationship with the department\_id field in the departments table. This allows for easy tracking of which faculty members belong to which department and eliminates the need for storing department information in multiple tables.

As a result of these design and implementation techniques, the database can provide accurate and reliable data for the VMC's operations. This is essential for the efficient management of the university's fleet and is a key aspect of the overall goal of improving the efficiency and effectiveness of the VMC.

# Database Creation and Tables



The vehicles table in the vehicle\_management\_database contains information about the vehicles in the fleet, including the vehicle's registration, type, make, model, year, colour, fuel type, odometer reading, last maintenance date, registration date, acquisition date, and availability. The primary key is the vehicle's registration.

In this table, the vehicle\_registration column uses a VARCHAR data type with a length of 7. This was chosen because vehicle registration numbers typically have a fixed length of 7 characters, and allocating less space for these records helps to improve the efficiency of the database. Additionally, the vehicle\_type, vehicle\_make, vehicle\_model, vehicle\_colour, and vehicle\_fuel\_type columns also use the VARCHAR data type with a length of 45. This allows for a wide range of string values to be stored in these columns, while still conserving space in the database. The vehicle\_year column uses the YEAR data type, which is more efficient for storing year values and also helps to ensure that the data is stored in a consistent format. The vehicle\_odometer\_reading, vehicle\_last\_maintenance\_date, vehicle\_registration\_date, and vehicle\_acquisition\_date columns use the DATE data type, which is best suited for storing date values. Finally, the vehicle\_availability column uses the TINYINT data type, which is used to store small integer values and is perfect for storing the availability status of the vehicles. Overall, the choice of data types in the vehicles table was carefully considered to ensure that the data is stored in an efficient and consistent manner.

This table is called departments and it contains information about the various departments at the university, including a unique ID, department name, department head ID, phone number, email, and location. It has a primary key on department\_id.

In the departments table, I used the INT data type for the department\_id column as it is a unique identifier for each department and will be used as a primary key. I also used INT for the department\_head\_id column to store the ID of the department head. The department\_name, department\_phone, department\_email, and department\_location columns are all VARCHAR with a length of 45. This is because these are all text fields and the length of 45 will provide enough space for the expected input while also being efficient in terms of storage. The department\_id is set as an auto\_incrementing primary key, which ensures that each department has a unique ID that is automatically generated by the database.

The table faculty\_members contains information about the faculty members at New Century University. It includes columns for the faculty member's ID (which is set as an auto-incrementing primary key), the department ID they belong to, their first and last name, email, phone number, and travel authorization status.

The datatype for ID, department ID, and travel authorization status is INT. The first and last name, email, and phone number are VARCHAR with a length of 45, 15 respectively. The table also includes a foreign key constraint to ensure that the department ID entered for each faculty member is a valid ID in the "departments" table. This ensures data integrity and consistency between the two tables.

For the sake of conciseness, from this point forward I will only show the code of the tables with minimal description as most of my reasoning for datatypes and foreign keys has already been stated.



Reservations table for storing any reservations made with intent to sign out a vehicle.



Trip completion form table to store the details of any completed trip in a vehicle.



Sign out forms table to store the details of a vehicle being signed out.



Maintenance logs table for logging the details of a vehicle going in for maintenance.



Maintenance detail forms table for storing the details and parts used on a specific job during a vehicle’s maintenance. I have made this many to one with the maintenance logs so that if there are multiple tasks performed on the vehicle by a different person each task, it can still be stored.

Parts usage forms table to log the usage of a part for a maintenance task.

Parts table to store the current inventory of parts.



Staff members table to store the staff members to sign out vehicles.



Mechanics table to store the mechanics performing maintenance tasks.

# Transactions and Reports

In the brief typical queries that would be needed were stated, I will be showcasing how I implemented these and them working with the dummy code I entered into the database (found in dummy-data.sql). The functionalities necessary are as follows:

**Display all vehicles available on a given date.**

The code below creates a view in the vehicle management database called available\_vehicles which will select the vehicle’s registration, make, model, year, and colour and head the columns appropriately but only when the vehicles availability is 1 (true, meaning it is available).

A view is appropriate here as it can update dynamically as the records in the table change and the view can be selected rather than the whole vehicles table which would contain a lot of unnecessary data with column headers which are not as easily readable for the user.



When entering the query “SELECT \* FROM `available\_vehicles`;” I get the following output, which is correct as it shows all the vehicles bar the two that are stated as unavailable.

**Graphical user interface, application, table

Description automatically generated**

**Display how many vehicles each department has used so far.**

The code below creates a view in the database called department\_vehicle\_usage which will select the department id, name, and number of vehicles the used by counting the rows when sign out forms, faculty members and departments are joined based on the criteria shown. It then groups these by department id and orders in descending order by the number of vehicles used.

A view is good here as it is an efficient way to group data from multiple tables together into one easy to read place based on the specific criteria needed.



When entering the query “SELECT \* FROM `department\_vehicle\_usage`;” I get the following output, which shows the correct number of vehicles used per department based on dummy data.

Graphical user interface, text, application

Description automatically generated

**Display total mileage driven by a department in a year.**

For this I actually implemented multiple versions due to it saying in the brief that they generate a monthly report of mileage by vehicle, faculty and department. The way I went about this was to create a view that will show the mileage driven by each vehicle, faculty member and department in the previous month, along with views to show the all time mileage of every vehicle and department and on top of this a view to show the mileage of a department for last year as shown below.













On top of this, since there are multiple views for mileage by department and mileage by vehicle based on the timeframe, I chose to create a procedure for each which takes in a string argument of ‘ALLTIME’, ‘YEARLY’, or ‘MONTHLY’ and selects the correct view based on this argument. This streamlines all of the functionality into one simple query which can show different outputs based on the user input. The procedures are as follows:



Since the different versions all show mileage in the same way but for different entities I will only show the functionality of the different timeframes in the show\_mileage\_by\_department procedure.

With the query “CALL `show\_mileage\_by\_department`('ALLTIME');” I receive the output:

**Graphical user interface, text, application

Description automatically generated**

With the query “CALL `show\_mileage\_by\_department`('YEARLY');” I receive:

Graphical user interface, text, application, email

Description automatically generated

And finally with the query “CALL `show\_mileage\_by\_department`('MONTHLY');”:

Graphical user interface, text, application

Description automatically generated

Which are all correct in the context of the dummy data I used.

**Show details of a particular bill.**

For this I firstly created a view to display the bills of each completed trip based on the amount of fuel purchased if any, multiply the amount by a given fuel price rate and round it to two decimal places concatenated with a £ symbol (if fuel purchased is null then it is 0). This is paired with the departure and return dates of the trip along with the credit card used.



I then created a procedure to allow the user to select a specific bill based on the id of the completed trip:



When using this procedure in the query “CALL `show\_bill`(2);” I get the intended result:

Table

Description automatically generated

**Display those who booked vehicles but didn’t use them.**

Once again for this I created a view to make it easier to read the information and to allow for this to be used in future developments of this database if necessary. The view is as follows:



This view will select any reservations where the departure date has passed and yet there has been no sign out form for this particular reservation, we can see this when executing the query “SELECT \* FROM `unused\_reservations`;”:

Graphical user interface, text

Description automatically generated with medium confidence

On top of these I also included some extra views to fill out some extra functionalities asked for within the brief itself such as a parts usage report and a vehicle maintenance summary for each month. These are as follows:



And:

  
The parts usage summary shows the date of part use, the mechanic who used the part, and the part type, name and quantity, it also shows the cost of the usage based on the price of the part and the quantity used. This is possible by joining the parts, parts\_usage\_forms and mechanics tables. It also only selects rows where the date is within the previous month and then orders by date.

The vehicle maintenance summary shows the date the maintenance log was created, the registration of the vehicle, the expected return date and the actual return date based on the latest maintenance detail form date. This also only selects rows where the date is within the previous month and orders by date.

We can test these views by selecting them with a select statement such as:

“SELECT \* FROM `last\_month\_parts\_usage\_summary`;”

Table

Description automatically generated

And “SELECT \* FROM `last\_month\_vehicle\_maintenance\_summary`;”

Table

Description automatically generated

Which both return the expected results.

# Triggers

In this database, triggers can be used to automatically update inventory, odometer readings, and vehicle availability when certain actions are performed, such as completing a trip or signing a vehicle out for maintenance. This can help to ensure that the data in the database is accurate and up-to-date and can also save time and reduce errors associated with manual data entry. Overall, the use of triggers can help to improve the efficiency and accuracy of the VMC's operations and can make it easier for staff to manage and report on the university's fleet of vehicles.

This trigger is made to update the quantity of a certain part in the parts inventory when a new insert is made in parts usage forms to a new value based on the quantity of the part used.



We can test this by checking the current state of the parts table:

Table

Description automatically generated

And then inserting a new row into parts usage forms where 5 oil filters are used and checking again:

Table

Description automatically generated

This next trigger will restock a part when it drops below its minimum allowed quantity.



We can test this by using one more oil filter to make the quantity drop below 5, which should make it restock by adding the minimum amount to the current quantity, giving us 9.

Table

Description automatically generated

The following trigger will update a vehicle’s odometer after it has completed a trip.



We can test this by checking the vehicles table before and after completing a trip with the Toyota Camry:

Graphical user interface

Description automatically generated

Table

Description automatically generated

The odometer reading went up the correct amount, meaning this trigger is functioning properly.

We can test the next two functions together as they have similar functionality, these will update a vehicle to unavailable when it is either signed out or in for maintenance.



If we check the available vehicles view currently:

Graphical user interface, application, table

Description automatically generated

And then sign out the Toyota Camry and put the Ford Explorer in for maintenance we get this:

Graphical user interface, table

Description automatically generated

We can see they have been removed from the available vehicles view, meaning their availability was successfully changed to 0.

The final trigger is to make a vehicle available again when a trip is completed.



We can test this by completing the trip from before with the Toyota Camry, and checking the available vehicles view again:

Graphical user interface, application, table

Description automatically generated

Showing this is also fully functional.

# Conclusion

In summary, the application of a database system for the New Century University's Vehicle Management Centre has significantly enhanced the efficiency of managing the university's fleet of vehicles. The database design, including the use of database normalization, has ensured efficient data management. The implementation of automated reservation, sign-out, and trip completion forms, as well as the ability to track usage, maintenance, and inventory, has streamlined the vehicle management process.

Additionally, the database provides a comprehensive set of reports on vehicle usage and maintenance to allow for better decision-making and management of the VMC. The use of triggers to update the inventory, odometer reading, and vehicle availability has further improved the efficiency of the system. Overall, the database has been an asset to the VMC, and its continued use and maintenance will ensure the smooth operation of the university's vehicle fleet.

The database could be expanded to include a system for managing driver's licenses and certifications as well as including a system for managing maintenance costs, which would allow for better forecasting and budgeting.