CMP2806M Scalable Database Systems Assignment 1 Report

Entity Relationship Diagram

Figure 1 is the Entity Relationship Diagram (ERD) exported from MySQL that depicts the relationships between entities (tables) in the database created for the vehicle management centre. An ERD is a visual representation of the information flow and logical organisation of a data model. The ERD attempts to highlight entities (tables), attributes (characteristics or property of the entities) and relationships, they help inspect the flow of data throughout the model. The explanation of choices and relationships in the ERD is below in the Entities and Attributes section.

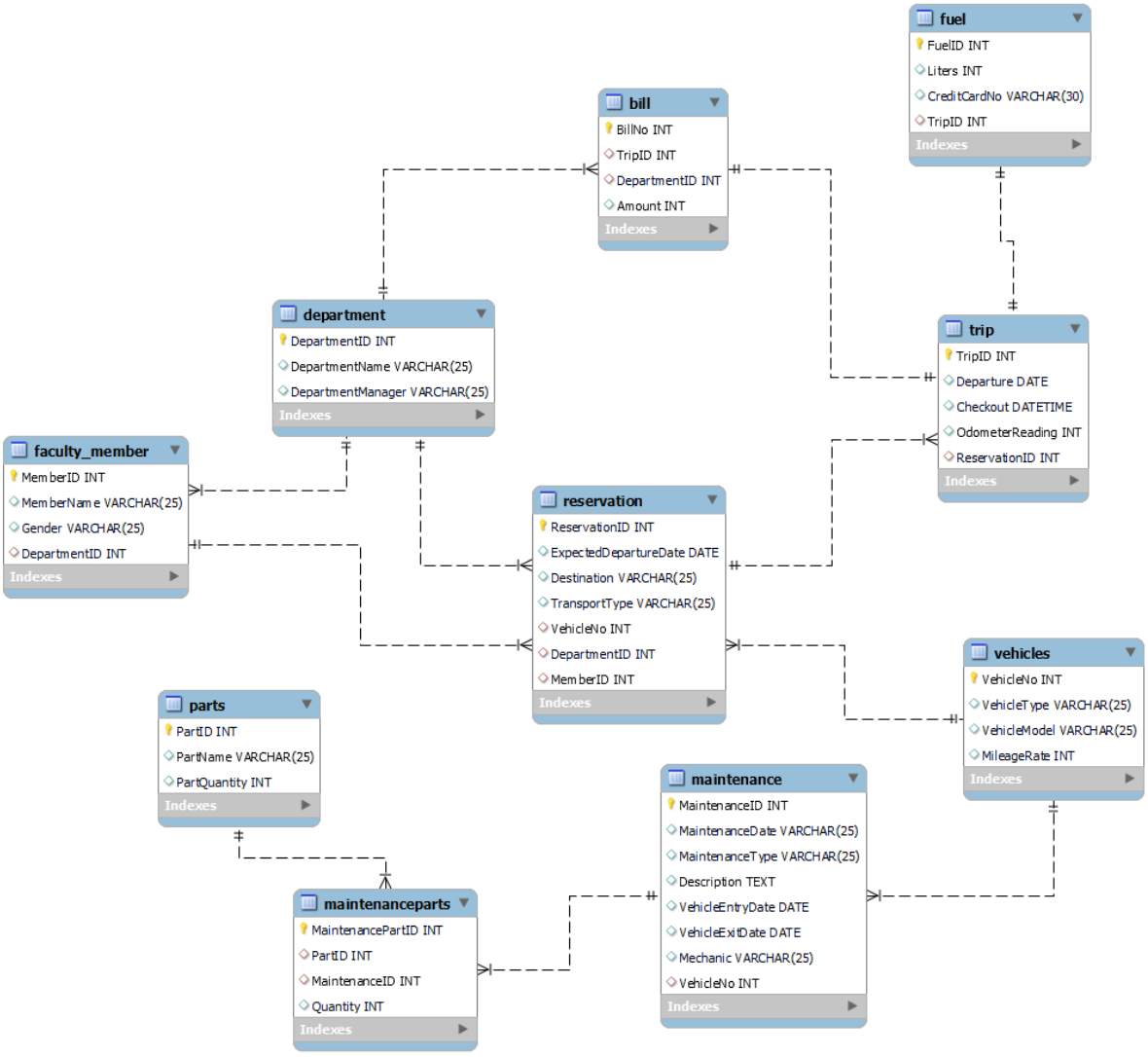


Figure 1. An Entity Relationship Diagram of the vehicle management centre database

Entities and Attributes

Entities are a subject of stored information, such as a person, group, category of objects, or idea. They explain the nature content that is being stored, which often maps to one or more related tables.

Attributes are characteristics or features that describes an entity, such as in the VMC model, faculty_member having an attribute MemberName (IBM, 2022).

Faculty Member

As outlined in the brief, there are faculty members that belong to a department, thus there must be a foreign key of DepartmentID in the faculty member table. The attributes, starting with the primary key MemberID whom auto increments when another member is added. This stores the ID of the faculty member, allows the table to create relationships, by using the MemberID as a foreign key, between the faculty member and reservation table so the faculty member can create a reservation for a vehicle. MemberName and Gender are also attributes attached to a faculty member to help identify the member. Its relationships include having a many:1 relationship with the department table due to many faculty members can belong to a singular department, and a 1:many relationship with the reservation table as 1 faculty member can create many reservations.

Department

DepartmentID is the primary key which allows it to have relationships with many other tables. From the ERD it is seen that the DepartmentID is a foreign key to the faculty_member entity as every member of the faculty belongs to a singular department thus must have a DepartmentID linked to them, it is a relationship of 1:m as many faculty members belong to 1 department. Furthermore, the department entity is linked to the bill entity as the brief outlined that the department is billed for any expenses due to the faculty member thus it is a foreign key in the bill table, it has a relationship of 1:m as 1 department has many bills. The foreign key DepartmentID in the reservation table allows the department to reserve vehicle for its faculty and possibly query how many reservations they have total within the department. The attributes DepartmentName and DepartmentManager help identify what departments the faculty members belong to and the supervisors they have.

Reservation

The reservation entity contains the primary key ReservationID which stores the ID of the reservation, it also contains the foreign keys of VehicleNo, DepartmentID and MemberID. The ReservationID is used as a foreign key in the trip table as every faculty member must checkout each vehicle, also, the department must know the amount of fuel used or how many miles they have completed which is linked to the trip table. Moreover, there is a VehicleNo foreign key in this entity as the department would like to know what vehicle is being reserved, so the database could be possibly queried to check the vehicle is available on a certain date, it has a many:1 relationship with the vehicles table. The attributes ExpectedDepartureDate, Destination and TransportType are all essential as outlined in the requirements, it is assumed the faculty member will depart on the date thus the car is reserved for the whole day but is then returned the same day. The TransportType denotes what the transport is for, such as, for a lecture, seminar etc.

Parts

PartID is the primary key for the parts table, this helps keep a relationship between the parts in inventory and the parts that need maintenance, thus it is a foreign key in the maintenanceparts table. The attributes PartName and PartQuantity provide the name of the part such as, filter or belt, and provide the number of the parts that are in the inventory of the vehicle management centre.

Maintenanceparts

This table consists of the parts that need maintenance. The primary key MaintenancePartID is used to keep track of the part that is in maintenance, it has a foreign key PartID which allows it to be linked to the parts table so that the part name can be identified if it is needed to be queried. Another foreign key MaintenanceID allows it to be linked to the maintenance table so then it can be linked further to the specific vehicle the part belongs to. It has a many to 1 relationship with the maintenance table as many parts could be having maintenance. The quantity attribute denotes the amount of those parts that are in maintenance.

Maintenance

The primary key MaintenanceID allows the database to link this table and the maintenance parts table to keep track of which parts are currently in maintenance. The attributes are as such due to the nature of the requirements; the description of the maintenance, the type of maintenance occurring, the date which the car enters and exits. The mechanic's name is also labelled in this. The foreign key VehicleNo from the vehicles table connects the vehicle identification to the maintenance table to check which car is having maintenance. It has a m:1 relationship with the vehicles table as 1 vehicle could be having many pieces of maintenance.

Vehicles

VehicleNo is the primary key, and it keeps track of the identification of the car. The vehicle type and model are also kept in this table with the mileage rate tying into the vehicle. This allows the user to query the amount it costs the department for a specific trip as this table links to the trip table which has the odometer reading if the faculty_member has filled out the amount of fuel used as seen below in Fig 2. It has a 1:many relationship with the maintenance table as 1 vehicle could be having many different parts of maintenance completed. It also has a 1:many relationship with the reservation table as 1 vehicle can have many reservations on it for different dates.

```
SELECT f.liters*t.OdometerReading as cost
from trip t
inner join fuel f
WHERE t.TripID = f.TripID;
```

Figure 2. Example of a query for the VMC.

Trip

TripID is the primary key for this table it allows it to link with the fuel and bill table to keep track of the amount of fuel used and the bill for that trip. This is the form that contains the attribute checkout which contains the date and time of which the faculty_member signed the form so it can be returned the same day. It also contains the odometer reading to see how far the person used the vehicle and the departure date. It contains the foreign key ReservationID so each trip can be linked to a reservation when departing. It has a 1:1 relationship with the fuel table 1 trip tracks 1 amount of fuel, also having a 1:1 relationship with the bill table as 1 trip will have 1 bill for it.

Fuel

The primary key FuelID this allows it to keep a record of which trips used a certain amount of fuel, hence the litres attribute. The CreditCardNo attribute keeps a record of the card used to pay the amount of fuel which ties into the trip table and then the reservation table, so we know which faculty_member paid for it. It has a 1:1 relationship with the trip table as 1 trip will have 1 amount of fuel used.

Bill

This entity was required as outlined in the brief; a bill is created so that the department can pay for it. The primary key is BillNo and has two foreign keys, the departmentID and TripID, this allows the bill table to have a relationship between the trip table and the department table so that the department can check how many bills they must pay. Thus, it has a 1:1 relationship with the trip table as 1 trip will have 1 bill, it also has a many:1 relationship with the department table due to 1 department having many bills.

Normalisation

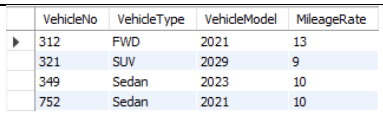
Normalisation consists of decomposing the database relations by breaking the attributes into smaller relationships to eliminate redundant data thus the database can take up less storage space. It allows the database to be more accessible to the user as each table has partial or functional dependencies with other tables.

Populated Table

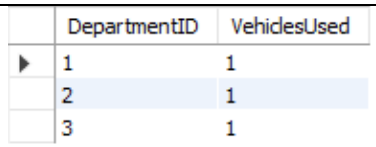
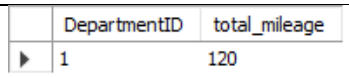
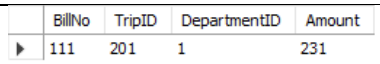
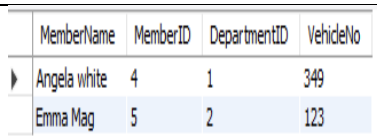
Every entity in the database has been populated with some arbitrary data to verify functionality of the relational database. Inserting data into the database allows me to practically test my queries to ensure they are outputting the desired results, the data inserted can be found in the SQL script. There are however some setbacks to my database such that I have not calculated the price based on the mileage and fuel litres in the fuel table, that are used in the bill entity; thus I inserted some data to ensure it was filled.

Queries and Testing

Previously, data was inserted into the database to test functionality, I will be performing queries on the inserted data to ensure that the results obtained are correct, therefore my database would be behaving as expected. The queries for the typical transactions as mentioned in the brief are contained in the SQL script.

Query	Expected Output/Explanation	Result
1. Display all available vehicles on a given date.	For this query it has been implemented as a stored procedure so that the user could input a date based on what they wanted. In the database it is assumed the vehicle is reserved for the full day but is also returned the same day. Thus only 1 vehicle is	

VehicleNo	VehicleType	VehicleModel	MileageRate
312	FWD	2021	13
321	SUV	2029	9
349	Sedan	2023	10
752	Sedan	2021	10

	reserved on the day I have set as a parameter which is Vehicle No '123' so there are 4 other cars that are available for that day.	
2. Display how many vehicles each department has used so far.	Due to the inserted data and my checkout attribute in the trip table, thus there are 3 vehicles with reservation ID 101, 102,103 which belong to departments 1, 2 and 3, respectively. Thus, the expected output should be 1 vehicle used for department 1,2 and 3.	
3. Display the total mileage driven by a department this year.	For this query I have implemented it as a stored procedure so that the user could input the department, they would like to see the total mileage driven by. I used department 1 in the calling of the procedure, they have only 1 car that departed on a trip and which has travelled 120 miles on the odometer reading, therefore the expected output is 120.	
4. Show details of a particular bill	For this query I have implemented it as a stored procedure so the user could input the bill number they choose and retrieve the data from that particular bill. The billNo I have chosen is '111' so it should output the TripID, DepartmentID and Amount.	
5. Display those who booked vehicles but not actually used them.	For this query I have created it is a view called bookedNotUsed, displaying the view by selecting * from bookedNotUsed. It should display the MemberName, MemberID, DepartmentID and VehicleNo. In the data, there is 5 reservations but only 3 trips made, thus there should be 2	

	people with reservations but not used the car yet.	

Queries

1.
```
-- Display all available vehicles on a given date.
DELIMITER //
create procedure availableVehicles(IN dateWanted varchar(10))
BEGIN
select * from vehicles where VehicleNo not in (select VehicleNo from reservation where ExpectedDepartureDate=dateWanted);
END//
call availableVehicles('2023-01-02');
```

2.
```
-- Display how many vehicles each department has used so far.
select r.DepartmentID,count(r.VehicleNo) as VehiclesUsed from  reservation  r
inner join trip t on (r.ReservationID=t.ReservationID)
group by r.DepartmentID;
```

3.
```
-- Display the total mileage driven by a department this year.
DELIMITER //
create procedure showMileageOfDepartment(IN DepartmentID int)
BEGIN
select r.DepartmentID,sum(t.OdometerReading) as total_mileage from  reservation r
inner join trip t on (r.ReservationID=t.ReservationID)
where year(r.ExpectedDepartureDate)='2023' and r.DepartmentID = DepartmentID;
END //
call showMileageOfDepartment(1);
```

4.
```
-- Show details of a particular bill.
DELIMITER //
create procedure showBill(IN BillNo int)
BEGIN
select BillNo, TripID, DepartmentID, Amount from bill where Bill.BillNo=BillNo;
END //
call showBill(111);
```

5.
```
-- Display those who booked vehicles but not actually used them.
create view bookedNotUsed as
select f.MemberName,r.MemberID,r.DepartmentID,r.VehicleNo from reservation r
LEFT JOIN faculty_member f
ON r.MemberID = f.MemberID
where r.ReservationID not in (select ReservationID from trip where Departure<now());
SELECT * from bookedNotUsed;
```

References

1. IBM (2022) *Key concepts: Entity, attribute, and entity type*. Available at: https://www.ibm.com/docs/en/imdm/12.0?topic=concepts-key-entity-attribute-entity-type (Accessed: January 18, 2023).