IN1010 Data Modeling Exercise 2 – Classic Car Club

In this exercise you have to decide what the entities (tables) are, which attributes should belong in which tables, and what the relationships should be.

A classic car club where members pay a fee to belong and can book out various classic cars for up to 5 days is developing a database to replace its existing paper-based records system. The customer's membership fee is translated into club points. The database needs to record members by their unique membership number, name, address, date of birth and club points. The system needs to record bookings of cars with a unique booking id, a start date and a number of days. The cars available to members need to be put in the database. Each car has a registration number, make, model, mileage and band. When a booking is complete the system should store the invoice information which should show the end date of the booking and the cost of the car in club points.

MEMBERS: (ENTITY)
-Member_Number (Primary Key)
-Member_Name (Attribute)
-Member_Address (Attribute)
-Member_DOB (Attribute)
-Club_Points (Attribute)

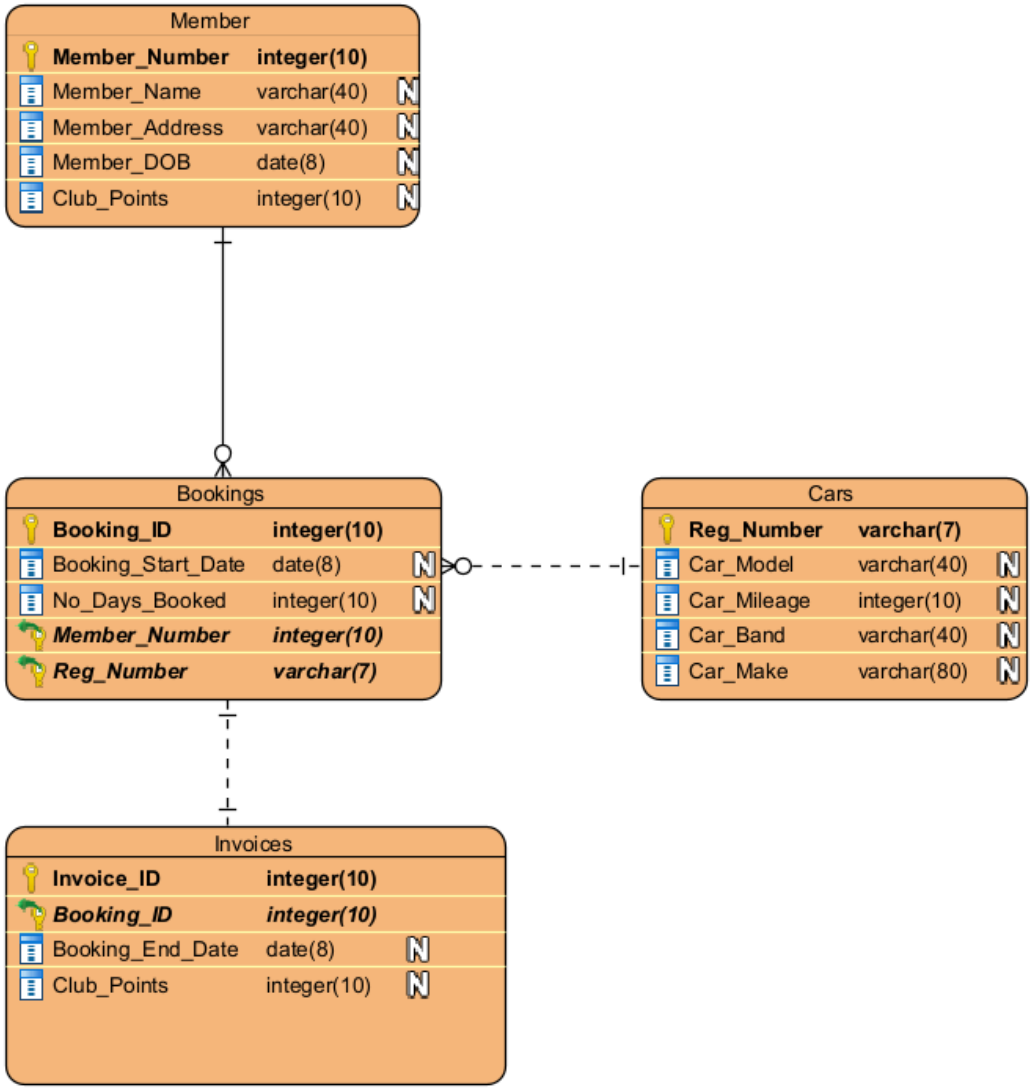
BOOKINGS: (ENTITY)
-Booking_ID (Primary Key)
-Booking_Start_Date (Attribute)
-No_Days_Booked (Attribute)
-Member _ID (Foreign Key)
-Reg_Number (Foreign Key)

CARS: (ENTITY)
-Reg_Number (Primary Key)
-Car_Model (Attribute)
-Car_Mileage (Attribute)
-Car_Band (Attribute)
-Car_Make (Attribute)

INVOICE: (ENTITY)
-Invoice_ID (Primary Key)
-Booking_ID (Foreign Key)
-Booking_End_Date (Attribute)
-Club_Points (Attribute)

Develop data model in Visual Paradigm to represent the above scenario.

Hint: The relationship between two of the tables is one we haven't used before, but it is on the Visual Paradigm relationship menu.



Member --> Bookings = One to Many (A member can have multiple bookings, but each booking only applies to one member, showing it is a One to Many)

Car --> Bookings = One to Many (A car can have multiple bookings over time, but a booking can only have 1 car, this making it a One to Many)

Booking --> Invoice = One to One (Each booking creates one invoice, and each invoice only applies to one booking, making it a One to One)