Assumptions and Notes

Student Name: **Joel Pillar-Rogers** Student Id: **2017545** Student FAN: **pill0032**

Assumptions

I made a number of assumptions regarding the company instructions:

1) I have not implemented a method within my conceptual model to limit Clients hiring more than one Vehicle at a time. Instead, I have left this as something to be addressed through company policy.

2) I have not forced participation of Clients in Hires or Bookings (0..*), which means they can exist in the database without either. This is because a Client's first contact with the company could be a Hire or a Booking, so I couldn't force participation in either. The company can decide whether it would like to add Clients who have no bookings. I also think it follows from this that the driversLicenseNumber attribute should not be NULL.

3) I decided that a Client *hires* Vehicle relationship was far too complicated to be treated as a relationship and Hire had to be a separate entity (and the same for Booking).

4) I created a new attribute called *bookingID* to serve as a primary key for Hires and Bookings as I think this is a cleaner way to create unique tuples for these relations than having composite primary keys. I assigned it an Integer, 10.

5) The instructions were unclear on how to record tariffs. I think they want only a handful of tariffs (hence, String, 2 and an ambiguous use of 'only'). However, I couldn't make this work in the conceptual model without adding another table, as there needs to be a unique tariff for each set of conditions for each make/model combination (a many-to-many relationship). As this may end up exceeding 99 different tariffs I decided to change the data requirements to String, 3. During implementation, I assume it would be simpler to create another table for this cartesian product.

6) I have allowed Clients to optionally specialise into Drivers. This means Clients don't have to be Drivers, but Drivers are recorded as Clients. Hire nominates a Driver Client, which may be different to the Client hiring the Vehicle. Driver doesn't have any separate attributes and I am unclear how this would be implemented (a yes/nothing attribute in Client?).

7) I created a new attribute, companyName, for when a Client is representing a company.

8) I have assumed a Client can be both a Nominated Person for a Company and also a Driver at the same time (or neither).

9) I assumed a 0..1 multiplicity for Hire to Invoice. This is because an invoice is not generated until a Vehicle is returned and allows Hires to be entered without a corresponding Invoice. I have left it to company policy to ensure Invoices are created when Vehicles are returned.

10) I've assumed InsurancePolicy is unique for each Hire (rather than a set of policies that you choose from), so I've used a 1..1 multiplicity. This assumption makes it possible that Integer, 5 is insufficient for a long running business, but I've left this as suggested for now.

11) I combined the "date" of PastService with the "date-to-be-done-by" of ScheduleService as I think this can be treated as a redundancy in my model.

12) I gave Service a mandatory participation in its specialisations, as it must be either a PastService or a ScheduledService.

13) I gave Vehicle a mandatory participation in its relationship with Service (1..*) as every Vehicle should have at least the next ScheduledService recorded. This still allows a vehicle to have no PastServices, as requested.

14) I haven't created a separate attribute to record whether a PastService was a scheduled or repair service. The company can have a policy to note this in the description field.

Simple Multiplicity Assumptions

1) A VehicleType may have no associated Vehicles in the current fleet (0..*)
2) A Depot may have no Vehicles currently located there (0..*)
3) Only one Vehicle can be selected on each Hire (1..1)
4) Only one VehicleType can be chosen on each Booking (1..1)
5) A Vehicle may have never been Hired (0..*)
6) A VehicleType may have never been Booked (0..*)
7) Only one Client can be recorded on each Hire (excluding the Driver) (1..1)
8) Only one Client can be recorded on each Booking (1..1)
9) Only one Driver can be nominated on each Hire (1..1)
10) A Driver can exist in the database without being nominated on a Hire (0..*)
11) Some of the DailyHireTariffs may have never been used (0..*)
12) A Depot may have had no Vehicles hired from there (0..*)
13) Every GenericBooking must have only one pickup Depot (1..1)
14) A GenericBooking may have no drop off Depot recorded if it is a Booking (0..1)
15) Only one Vehicle can be recorded for each Service (1..1)
16) A Service can only take place at one Depot (1..1)
17) A Depot may have had no Services take place there (0..*)

Notes

I also made several modelling-related decisions:

1) I have attempted to use Enhanced Entity Relationship modelling techniques where possible, partly to minimise attribute redundancy but also to gain experience in implementing them. This is apparent in the shared subclass for Hire and Booking (GenericBooking) and in the specialisations of Service and Client. I realise the trade-offs to this approach are more tables, intersecting relationship lines and potentially a more confusing diagram to interpret.

2) By creating the GenericBooking subclass I can directly connect both Hires and Bookings to Depots to allow an easy view of which VehiclesTypes will be needed in the future.

3) I decided to make the optional *preferredColour* of future bookings an attribute of the booking table, rather than a separate and weak/dependent entity. This is a cleaner execution of the same concept but with fewer tables. I have provided optionality by allowing preferred colour to be NULL.

4) I decided to have Invoices *generated_For* Hires, rather than the opposite direction for this relationship. I believe this means the Hire primary key will be kept as a foreign key in the Invoice table. I think this is correct because Hires have optional participation in this relationship, whereas Invoices have mandatory participation and should be forced to list an existing Hire primary key when they are created.

5) Relatedly, I have decided to have InsurancePolicies *taken_On* Hires, rather than the reverse. This is so the mandatory participation of InsurancePolicy can be enforced, with NULL not allowed for the foreign key from Hire.

6) In the Service table, date is not the only primary key. There would be a composite key with fleetMembershipNumber, which would come into the table as a foreign key from Vehicle.

Tests

Finally, the test below is not supported by my database, but I actually think the test is incorrect, as future hires (Bookings) only choose a VehicleType, not a specific Vehicle. I believe all other tests are supported.

- List all vehicles that do not currently have a future hire booked