**Depot**

Depot (depotID, street, postcode, fax)

Primary Key depotID

Depot is a Strong entity because of the cardinalities of the existing relationships (\*..1).

This mean that Depot does not contain any foreign keys.

The composite attribute address is now "street" and "postcode".

Multi-Valued attributes such as phone[1..4] are now created as a separate table called “DepotPhone”. There are no ON UPDATE or NO DELETE as there are no foreign keys.

**DepotPhone**

DepotPhone (depotID, phone)

Primary Key depotID, phone

foreign key depotID references Depot(depotID) ON UPDATE CASCADE ON DELETE CASCADE

DepotPhone is a weak entity. It only has a relation to the Depot table.

The DepotID is added to the table as a foreign key to identify the appropriate phone numbers. Each Depot can hold 1..4 phone numbers. (1..4 Relationship)

ON UPDATE CASCADE is used on depotID because depot can change.

ON DELETE CASCADE is used because if a depot doesn't exist, phone doesn't exist.

**Client**

Client (ClientID, street, postcode)

Primary Key (ClientID)

Client is a strong entity. The composite attribute address is now "street" and "postcode".

Client is a parent of the PersonalClient and CompanyClient entities as it follows an "Optional, OR" approach. The multi-malued attribute phone is no longer apart of the Client table and is now stored in a separate table called ClientPhone. Mandatory for a Client to exist (1..1). There are no ON UPDATE or NO DELETE as there are no foreign keys.

**PersonalClient**

PersonalClient (ClientID, fName, lName, title, driversNum)

Primary Key ClientID

Foreign key ClientID references Client(ClientID) ON UPDATE CASCADE ON DELETE CASCADE

Personal Client is a weak entity, and contains a relation to the Client table.

Personal Client included a composite attribute called name, where they're split off to "fName, lName and title". One to many Nominated Drivers are allowed drive a vehicle.

ON UPDATE CASCADE because we want to be able to update the data.

ON DELETE CASCADE is used so that if we want to delete a Personal Client.

**CompanyClient**

CompanyClient(ClientID, cName, Representative\_clientid)

Primary Key ClientID

Foreign key ClientID references Client(ClientID) ON UPDATE CASCADE ON DELETE CASCADE

Foreign key Representative\_clientid references PersonalClient(ClientID) ON UPDATE CASCADE ON DELETE RESTRICT

Company Client is a weak entity, and contains a relationship with the Client table.

Company Client also has a relationship with Personal Client, where 0..1 Personal Clients can represent a company. The cName attribute is classified to be optional because of the participation on the multiplicity.

* representative\_clientid: ON UPDATE CASCADE Because we want to be able to update the client information. ON DELETE RESTRICT Because we don't want be able to delete a client if they're listed as a represtative for a company.
* clientID: ON UPDATE CASCADE Because we want to be able to update the client information. ON DELETE CASCADE Because we want the company information to be deleted if they're no longer a customer.

**ClientPhone**

Client Phone (ClientID, phone)

Primary Key ClientID, phone

Foreign Key ClientID references Client (ClientID) ON UPDATE CASCADE ON DELETE CASCADE

Client Phone is a weak entity, and contains only the phone numbers of clients.

ClientID is represented as a foreign key. It is mandatory for a client to have a phone number. They can also include a 2nd phone number, however this is the maximum.

ON UPDATE CASCADE because we want to be able to update data.

ON DELETE CASCADE because we want to be able to delete information if they're no longer a customer.

**VehicleType**

VehicleType (mixID (INTEGER PRIMARY KEY AUTOINCREMENT), make, model, doors, body, VehicleTrim)

Primary Key (mixID)

VehicleType is considered to be a strong entity. It is mandatory for a Vehicle to have a VehicleType (One to one relationship) Subsequently contains no foreign keys (Strong entity)

A check constraint has been added to see if the correct identification

of trim and body has been selected. NO EXISTING FOREIGN KEYS SO NO ON UPDATE AND ON DELETE.

**Vehicle**

Vehicle (mixID, regNum, depotID, fleetNum, colour, nextServDate, nextServKilom, nextServDepotID)

Primary Key (regNum)

Foreign Key (depotID) references Depot (depotID) ON UPDATE CASCADE ON DELETE RESTRICT

Foreign Key (nextServDepotID) references Depot(DepotID) ON UPDATE CASCADE ON DELETE RESTRICT

Foreign Key (mixID) references VehicleType(mixID) ON UPDATE CASCADE ON DELETE CASCADE

Vehicle can be seen as a mix of a strong and weak entity. Vehicle contains derived relations from multiple tables such as Depot and VehicleType. DepotID does not have a not null type because the relationship to vehicle is optional. (Zero to One) Due to Vehicle and ScheduledService having 1..1 cardinalities, they have been merged together.

* depotID: ON UPDATE CASCADE because we want to be able to update the depotID.

ON DELETE RESTRICT because we don't want to be able to delete a depot if there are still cars registered to a depot.

* nextServDepotID: ON UPDATE CASCADE because we want to update the depotID when there is a next service. ON DELETE RESTRICT because we do not want a depot to be deleted.
* mixID: ON UPDATE CASCADE because if we want to be able to update information.

ON DELETE CASCADE because we want to be able to delete vehicles in bulk. (In case we want to sell all of one type of car for instance, a Jeep Wrangler because it is un-safe and there is class action against the maker.)

**ServiceHistory**

ServiceHistory (regNum, depotID, date, cost, description)

Primary Key (regNum, depotID, date)

Foreign Key (regNum) references Vehicle (regNum) ON UPDATE CASCADE ON DELETE NO ACTION

Foreign Key (depotID) references Depot (depotID) ON UPDATE CASCADE ON DELETE SET DEFAULT

Service History is a weak entity. Service History can contain 0 or multiple past services.

Service History contains the foreign keys depotID and regNum from their appropriate parent tables as the Service History entities’ multiplicities are 0..\*.

* regNum: ON UPDATE CASCADE because we want to be able to update the regNum information ON DELETE NO ACTION because we want to be able to keep this information for the future.
* depotID: ON UPDATE CASCADE because we want to be able to update the depotID

ON DELETE SET DEFAULT because we want to be able to use the '99' depotID which shows that entry was originally made at a depot that does not exist anymore

**Booking**

Booking (bookingID, ClientID, depotID, mixID, startDate, hireDays, colour)

Primary Key (bookingID)

Foreign Key (ClientID) references Client(ClientID) ON UPDATE CASCADE ON DELETE CASCADE

Foreign Key (mixID) references VehicleType (mixID) ON UPDATE CASCADE ON NO ACTION

Foreign Key (depotID) references Depot (depotID) ON UPDATE CASCADE ON DELETE NO ACTION

Booking is a weak entity. It has an additional primary key of bookingID to Bookings can be made by one client, and can have 0 to many bookings. The tables VehicleType and Depot also follow the same structure as the Client table. The foreign keys from the three tables are added to booking of the 0..\* multiplicity.

* ClientID: ON UPDATE CASCADE because we want to be able to update information

ON DELETE CASCADE when a client is delete, we want to be able to get rid of their booking information.

* mixID: ON UPDATE CASCADE because we want to be able to update information

ON DELETE NO ACTION because we don't want to be able to delete a future booking.

* depotID ON UPDATE CASCADE because we want to be able to update information

ON DELETE NO ACTION because we don't want to be able to delete a future booking.

**HiredVehicle**

HiredVehicle (regNum, tariffID, depotID, clientID, date, cardType, cardNo, kilometrage, days, policyNumber)

Primary Key (regNum, date, clientID)

Foreign Key (regNum) references Vehicle (regNum) ON UPDATE CASCADE ON DELETE NO ACTION

Foreign Key (tariffID) references DailyTariff (tariffID) ON UPDATE CASCADE ON DELETE RESTRICT

Foreign Key (depotID) references Depot (depotID) ON UPDATE CASCADE ON DELETE SET DEFAULT

Foreign Key (clientID) references Client (clientID) ON UPDATE CASCADE ON DELETE CASCADE

HiredVehicle is considered as a weak entity. HiredVehicle has relations to 7 different tables, which only four foreign keys are implemented. (Vehicle, Depot, DailyTariff and Client)

These foreign keys were derived because of the multiplicities (0..\*)

* regNum: ON UPDATE CASCADE because we want to be able to update existing data.

ON DELETE NO ACTION because company might need access to this information.

* tariffID: ON UPDATE CASCADE because we want to be able to update existing data.

ON DELETE RESTRICT because we don't want to delete any tariffs if they have been used.

* depotID: ON UPDATE CASCADE because we want to be able to update existing data.

ON DELETE SET DEFAULT because we want to show that the booking was made at a depot that no longer exists

* ClientID: ON UPDATE CASCADE because we want to be able to update the existing data.

ON DELETE CASCADE because we want to be able to delete client data.

**HiredVehicleDriver**

HiredVehicleDriver (regNum, date, clientID, DriverClientID)

Primary Key (regNum, date, clientID, DriverClientID)

Foreign Key (regNum, date, clientID) references HiredVehicle (regNum, date, clientID) ON UPDATE CASCADE ON DELETE RESTRICT

Foreign Key (DriverClientID) references PersonalClient (clientID) ON UPDATE CASCADE ON DELETE RESTRICT

HiredVehicleDriver is a weak entity. As the entity is not mentioned on the ER Diagram or Dictionary, we assume that it has a (1..1 | 1..\*) relationship on hired vehicles. We also assume it has a 0..\* relationship with the PersonalClient entity. This entity has all of its attributes related to other tables, meaning that they're all primary and foreign keys.

* regNum, date, clientID: ON UPDATE CASCADE because we want to be able to update information ON DELETE RESTRICT so we don't accidently delete something we need to keep. Also we don't want to delete a client if they are still listed as a designated driver.
* DriverClientID: ON UPDATE CASCADE because you info update to flow through the whole database.
* ON DELETE RESTRICT because you don't want to be able to delete a client that is a driver.

**DailyTariff**

DailyTariff (tariffID, conditions)

Primary Key (tariffID)

DailyTariff is a strong entity. Daily Tariff contains the information of conditions that apply to a VehiceType, and as well as conditions that can apply. (0..\* Tariffs can be assigned to Vehicles) NO FOREIGN KEYS SO NOT ON UPDATE OR DELETE CONSTRAINTS.

**DailyTariffCost**

DailyTariffCost (mixID, tariffID, rentalPrice)

Primary Key (mixID, tariffID)

Foreign Key (mixID) references VehicleType (mixID) ON UPDATE CASCADE ON DELETE CASCADE

Foreign Key (tariffID) references DailyTariff (tariffID) ON UPDATE CASCADE ON DELETE CASCADE

DailyTariffCost is derived from the tables VehicleType and DailyTariff. This is because both of the cardinalities of the tables are many to many. Another relation is created to store the information of the foreign keys from the two tables, and as well as the relationship attribute rentalPrice.

* mixID: ON UPDATE CASCADE because we want to be able to update existing information.

ON DELETE CASCADE because we don't want to store the costs if something exists.

* tariffID: ON UPDATE CASCADE because we want to be able to update existing information

ON DELETE CASCADE because if the tariff doesn't exist, the cost doesn't exist.

**Insurance**

Insurance (insuranceID, policyType, cost)

Primary Key (insuranceID)

Insurance is a strong entity. A Client may choose to have insurance, where the multiplicities are set to Zero to one. There are no derived foreign keys added to the entity as it is a strong entity. NO UPDATE OR DELETE CONSTRAINTS BECAUSE THERE ARE NO FOREIGN KEYS.

**Invoice**

Invoice(invoiceID, returnDate, qualityCheck, regNum, tariffID, clientID, datePaid)

Primary Key (invoiceID)

Foreign Key (regNum, higherDate, clientID) references HiredVehicle (regNum) ON UPDATE CASCADE ON DELETE NO ACTION

Invoice is generated for a hired vehicle, however there may not be Invoice

generated due to the multiplicity being zero to one. Foreign Keys are derived from the HiredVehicle table which includes regNum, tariffID and ClientID.

The /finalCost attribute is derived from multiple tables, where a calculation is needed to

determine the absolute final cost of the vehicle hired.

* regNum, higherDate, clientID: ON UPDATE CASCADE because we want to be able to update these attributes in the table.

ON DELETE NO ACTION because upon generating the invoice, the regNum , higherDate and ClientID should not be deleted.

* policyNumber: ON UPDATE CASCADE because we want to be able to update information

ON DELETE RESTRICT because we do not want policyNumber to be deleted for the generation invoice

**Insures**

Insures (policyNumber, insuranceID)

Primary Key (policyNumber, insuranceID)

Foreign Key (insuranceID) references Insurance(insuranceID) ON UPDATE CASCADE, ON DELETE RESTRICT

Foreign Key (policyNumber) references HiredVehicle(policyNumber) ON UPDATE CASCADE ON DELETE CASCADE

Foreign Key (higherDate) references HiredVehicle (date) ON UPDATE CASCADE ON DELETE RESTRICT

The insures table is created to contain the insuranceID and the policyNumber,

which was a relational attribute from the HiredVehicle to the Insurance Table.

The policyNumber exists in the HiredVehicle entity, and the insuranceID exists in the

insurance table, making them both foreign keys.

* insuranceID: ON UPDATE CASCADE because we want to update the existing data.

ON DELETE RESTRICT because we do not want to delete used insurance

* policyNumber: ON UPDATE CASCADE because we want to be able to update the existing data. ON DELETE CASCADE has been selected because when a policyNumber exists, it doesn't need to be linked with an existing insurance.

**Records**

Records (invoiceID, cost)

Primary Key (invoiceID, cost)

Foreign Key (invoiceID) references Invoice (invoiceID) ON UPDATE CASCADE ON DELETE CASCADE

Records entity is an additional entity to store the calculated cost of the rented car from a top level interface.

* ON UPDATE CASCADE because we want to be able to update.
* ON DELETE CASCADE because once cost is payed, record of invoice will be removed.

**Final Invoice (View)**

The Final Invoice is a view that the program can query to see all the information from invoice along with the total cost.