HW MOTORS

F28DM Coursework 1, Heriot-Watt University

Ross Davidson, Ryan Anderson, Jose Fernandes, Kirshna Mattapalli,
Jack Miller and Blazej Niezabitowski

Contents

Introduction 2
ER Diagram 3
Relationship Schema 4
MySQL 6
Indexes 14

Introduction

Our group has been hired by HW Motors to create a Database, using mySQL, to manage their new vehicle rental company. This database will allow them to manage the hiring out of their cars and vans throughout the UK.

This must meet a series of requirements such as, but not necessarily limited to:

- Booking 7 days or more applies a discount
- Renting of the exact vehicle requested
- A wide range of vehicles, from small cars to Minivans.
- A clear start and end date
- The ability for all bookings to be made via a webpage or mobile app
- User registration
- Vehicle specifications

We will be assuming that vehicles can be rented from and returned to any base, although each vehicle will have its own home base that handles tasks such as its servicing, including MOTs. It is targeted at renting to the public, and only the public. They will only rent out cars to people above 25 years of age, and 30 for MPVs, vans and luxury vehicles, with actual pictures of the vehicle being hired shown to the person making the booking. Lastly, the only person who can make the booking is the Main Driver, though multiple co-drivers are supported.

ER Diagram

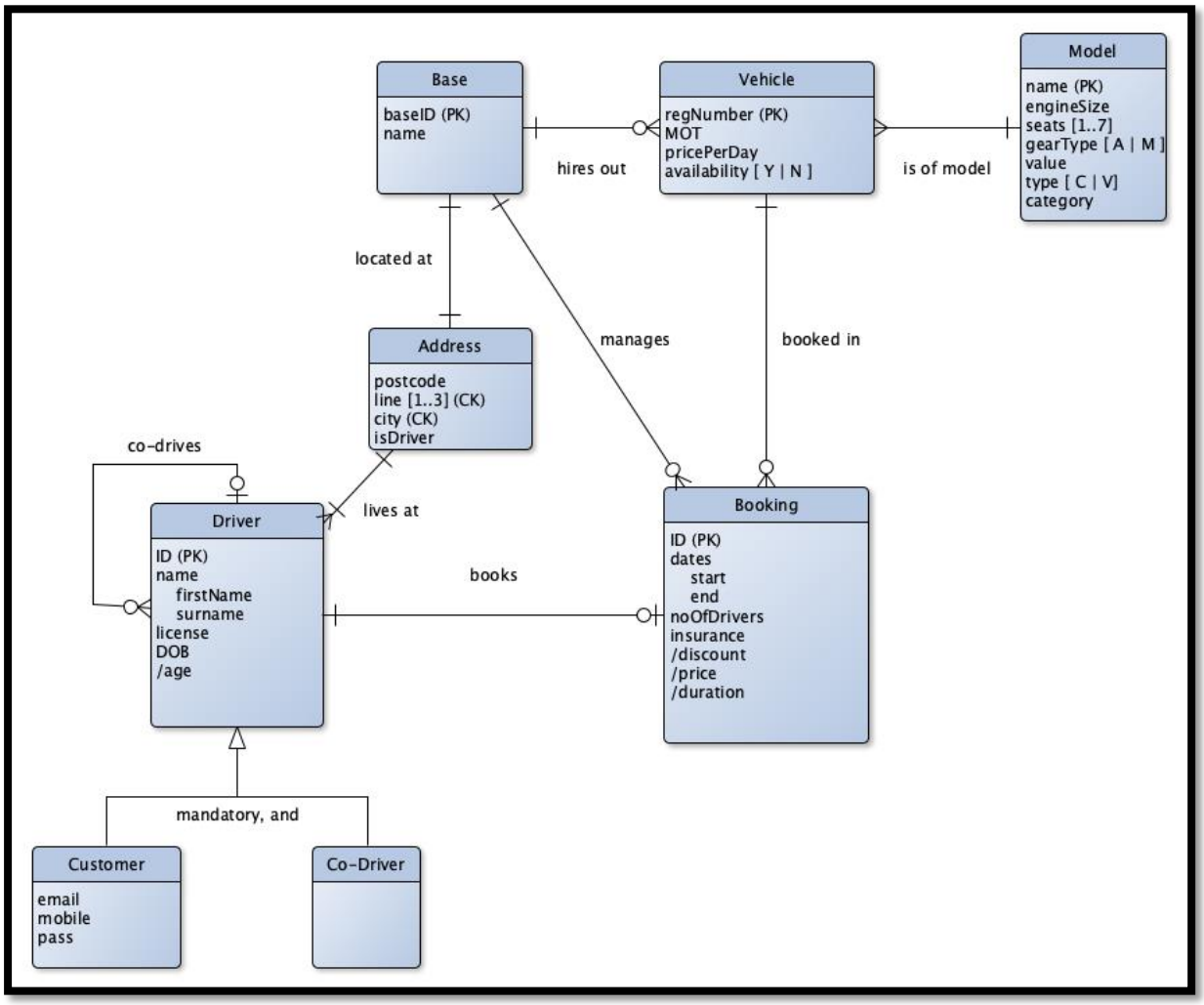

Fig 1. Our ER Diagram for the Database

With this Diagram, we have planned out how the database tables will relate to each other. For example, each base will have a unique ID, and its address, which is drawn from an address table, where we make use of composition to make a key. Each Vehicle is hired out from a base, and booked into a specific booking when made, with the base managing the Bookings. The bookings will derive the duration, overall price and duration from other fields, such as duration being based upon the start and end dates.

A driver must live at an address, and could potentially be a co-driver as well, allowing for a recursive relationship. While the driver *must* be a customer, they could also be a co-driver. The driver can then make a booking for their vehicle. The vehicles themselves are broken down into their models containing the vehicles specifications.

Relationship Schema

Below is our planned schema for this ER diagram, expanded to help with the creation of the actual database. Underline denotes a primary key while italics denotes references or foreign keys.

Base (

baseID,
name,
postcode,
line1

)

Notes: postcode and line1 are drawn from the Address table

Vehicle (

regNumber,
baseID,
bookingID,
MOT,
pricePerDay,
availability,
modelName

)

Notes: baseID and bookingID are drawn from the Base and Booking tables respectively. Availability determines whether the vehicle is able to be booked, while price per day is how much that vehicle costs to hire on a daily basis. modelName is drawn from the Model table.

Model (

name,
engineSize,
seats,
gearType,
value,
type,
category

)

Notes: Engine size is in cubic CM, gearType specifies Automatic or Manual, type specifies Car or Van. Category is the specific subset of a vehicle, such as Sports, luxury, etc. As these cars are extremely expensive, value can go up to 9 digits, with 2 passed the decimal place.

Booking (

ID,
startDate,
endDate,
noOfDrivers,
insurance,

driverID,
regNumber: varchar(10),
baseID: varchar(7)

)

Notes: discount, price and duration are derived values not shown in the schema. Duration draws from end and start dates, while discount then derives from that if it is greater than 7 days. Price then is derived from that, using a formula such as "price*0.80" if the discount rate is 20% for example. regNumber is a foreign key referencing the Vehicle table, while baseID references the Base table and determines which base the vehicle is to be collected from. The driverID is drawn from the Driver table and specifies the person who made the booking.

Driver (

ID,
firstName,
lastName,
license,
DOB,
postcode,
line1

)

Customer (

dID,
email,
mobile,
password

)

Co-Driver (

mainDriver,
dID

)

Notes: First and last name are combined into a composite attribute. Age is a derived value (not shown) based from current date and DOB. Postcode and line1 are determined by the Address table, much like it is for Base. Email, mobile and password belong to a more specialised subset of data, split into Customer, with another in the form of coDriver. This is a "mandatory and" specialisation, where a driver MUST be a customer but can also be a coDriver. mainDriver specifies the ID of the Driver who made the booking. bookingID references the booking table, and can be null if the driver has no current bookings.

Address (

postcode,
line1,
line2,

```
    line3,
    city,
    isDriver
)
```

Notes: This has no primary key of its own, but instead has a composite key made up of line and city that to sends to the base or driver table. This table is a weak entity, existing purely to supply data to the other two. City is set to a length of 58 as the longest city name in Britain is Llanfairpwllgwyn-gyllgogerychwyrndrobwllllantysiliogogogoch and it is feasible a driver or base could reside there. The isDriver field determines whether the entries are private data belonging to a driver, or the public operating address of the base, for GDPR purposes.

MySQL

Below is the script used to create the database tables and their relationships, all using the InnoDB storage engine.

```
-- phpMyAdmin SQL Dump

-- version 4.8.3

-- https://www.phpmyadmin.net/

--

-- Host: localhost

-- Generation Time: Feb 12, 2020 at 10:40 AM

-- Server version: 5.6.41-log

-- PHP Version: 5.6.40


SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";

SET AUTOCOMMIT = 0;

START TRANSACTION;

SET time_zone = "+00:00";



/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;

/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;

/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;

/*!40101 SET NAMES utf8mb4 */;


--

-- Database: `bn24`
```

```
--

-- --------------------------------------------------------

--
-- Table structure for table `Address`
--

CREATE TABLE `Address` (
  `line1` varchar(50) NOT NULL,
  `line2` varchar(30) DEFAULT NULL,
  `line3` varchar(30) DEFAULT NULL,
  `city` varchar(58) NOT NULL,
  `postcode` varchar(8) NOT NULL,
  `isDriver` tinyint(1) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Table structure for table `Base`
--

CREATE TABLE `Base` (
  `baseID` char(5) NOT NULL,
  `name` varchar(20) NOT NULL,
  `postcode` varchar(300) NOT NULL,
  `line1` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------
```

```
--
-- Table structure for table `Booking`
--

CREATE TABLE `Booking` (
  `ID` char(5) NOT NULL,
  `driverID` varchar(7) NOT NULL,
  `startDate` date NOT NULL,
  `endDate` date NOT NULL,
  `noOfDrivers` int(1) NOT NULL,
  `insurance` varchar(10) NOT NULL,
  `regNumber` char(7) NOT NULL,
  `pricePerDay` double(6,2) NOT NULL,
  `discount` tinyint(1) DEFAULT NULL,
  `price` double(6,2) NOT NULL,
  `duration` int(2) NOT NULL,
  `baseID` char(5) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Table structure for table `CoDriver`
--

CREATE TABLE `CoDriver` (
  `dID` varchar(7) NOT NULL,
  `mainDriver` varchar(7) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------
```

```
--
-- Table structure for table `Customer`
--

CREATE TABLE `Customer` (
  `dID` varchar(7) DEFAULT NULL,
  `email` varchar(80) NOT NULL,
  `mobile` varchar(13) DEFAULT NULL,
  `pass` varchar(32) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Table structure for table `Driver`
--

CREATE TABLE `Driver` (
  `ID` varchar(7) NOT NULL,
  `firstName` varchar(20) NOT NULL,
  `surname` varchar(20) NOT NULL,
  `license` varchar(16) NOT NULL,
  `DOB` date NOT NULL,
  `postcode` varchar(300) NOT NULL,
  `line1` varchar(50) NOT NULL,
  `Age` int(2) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Table structure for table `Model`
```

```
--

CREATE TABLE `Model` (
  `name` char(30) NOT NULL,
  `engineSize` int(10) NOT NULL,
  `gearType` enum('M','A') NOT NULL,
  `noOfSeats` tinyint(1) NOT NULL,
  `value` double(7,2) NOT NULL,
  `type` enum('C','V') NOT NULL,
  `category` varchar(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Table structure for table `Vehicle`
--

CREATE TABLE `Vehicle` (
  `regNumber` varchar(7) NOT NULL,
  `baseID` char(5) NOT NULL,
  `bookingID` varchar(7) NOT NULL,
  `MOT` char(12) NOT NULL,
  `pricePerDay` double(5,2) DEFAULT NULL,
  `availability` tinyint(1) NOT NULL,
  `modelName` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Indexes for dumped tables
--
```

```
--
-- Indexes for table `Address`
--
ALTER TABLE `Address`
  ADD PRIMARY KEY (`line1`,`postcode`);


--
-- Indexes for table `Base`
--
ALTER TABLE `Base`
  ADD PRIMARY KEY (`baseID`),
  ADD KEY `line1` (`line1`,`postcode`);


--
-- Indexes for table `Booking`
--
ALTER TABLE `Booking`
  ADD PRIMARY KEY (`ID`),
  ADD KEY `driverID` (`driverID`),
  ADD KEY `regNumber` (`regNumber`),
  ADD KEY `baseID` (`baseID`),
  ADD KEY `Booking_Index` (`insurance`,`regNumber`) USING BTREE;


--
-- Indexes for table `CoDriver`
--
ALTER TABLE `CoDriver`
  ADD KEY `dID` (`dID`),
  ADD KEY `mainDriver` (`mainDriver`);


--
-- Indexes for table `Customer`
```

```
--
ALTER TABLE `Customer`
  ADD KEY `dID` (`dID`);

--
-- Indexes for table `Driver`
--
ALTER TABLE `Driver`
  ADD PRIMARY KEY (`ID`),
  ADD KEY `line1` (`line1`,`postcode`),
  ADD KEY `License_Index` (`license`);

--
-- Indexes for table `Model`
--
ALTER TABLE `Model`
  ADD PRIMARY KEY (`name`),
  ADD KEY `engine_Index` (`engineSize`,`gearType`),
  ADD KEY `seats_Index` (`noOfSeats`);

--
-- Indexes for table `Vehicle`
--
ALTER TABLE `Vehicle`
  ADD PRIMARY KEY (`regNumber`),
  ADD KEY `baseID` (`baseID`),
  ADD KEY `bookingID` (`bookingID`),
  ADD KEY `Vehicle_Index` (`pricePerDay`) USING BTREE,
  ADD KEY `modelName` (`modelName`);

--
-- Constraints for dumped tables
```

```
--


--
-- Constraints for table `Base`
--
ALTER TABLE `Base`
  ADD CONSTRAINT `Base_ibfk_1` FOREIGN KEY (`line1`,`postcode`) REFERENCES `Address` (`line1`, `postcode`);


--
-- Constraints for table `Booking`
--
ALTER TABLE `Booking`
  ADD CONSTRAINT `Booking_ibfk_1` FOREIGN KEY (`driverID`) REFERENCES `Driver` (`ID`),
  ADD CONSTRAINT `Booking_ibfk_2` FOREIGN KEY (`baseID`) REFERENCES `Base` (`baseID`),
  ADD CONSTRAINT `Booking_ibfk_3` FOREIGN KEY (`regNumber`) REFERENCES `Vehicle` (`regNumber`);


--
-- Constraints for table `CoDriver`
--
ALTER TABLE `CoDriver`
  ADD CONSTRAINT `CoDriver_ibfk_1` FOREIGN KEY (`dID`) REFERENCES `Driver` (`ID`),
  ADD CONSTRAINT `CoDriver_ibfk_2` FOREIGN KEY (`mainDriver`) REFERENCES `Driver` (`ID`);


--
-- Constraints for table `Customer`
--
ALTER TABLE `Customer`
  ADD CONSTRAINT `Customer_ibfk_1` FOREIGN KEY (`dID`) REFERENCES `Driver` (`ID`);


--
-- Constraints for table `Driver`
```

```
--

ALTER TABLE `Driver`

  ADD CONSTRAINT `Driver_ibfk_1` FOREIGN KEY (`line1`,`postcode`) REFERENCES `Address` (`line1`, `postcode`);


--

-- Constraints for table `Vehicle`

--

ALTER TABLE `Vehicle`

  ADD CONSTRAINT `Vehicle_ibfk_1` FOREIGN KEY (`baseID`) REFERENCES `Base` (`baseID`),

  ADD CONSTRAINT `Vehicle_ibfk_2` FOREIGN KEY (`bookingID`) REFERENCES `Booking` (`ID`),

  ADD CONSTRAINT `Vehicle_ibfk_3` FOREIGN KEY (`modelName`) REFERENCES `Model` (`name`);

COMMIT;


/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;

/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;

/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Notes: This code is a dump from the generated database

Indexes

Below we specify five indexes that we believe could be useful to have indexed due to frequent queries. This does not include primary keys, as they are automatically indexed in mySQL.

1. pricePerDay in Vehicle
2. combined engineSize and gearType in Model
3. noOfSeats in model
4. license in Driver
5. combined insurance and regNumber in booking

We determined these 5 through debate. The price per day is something that would frequently be looked up when booking a vehicle and would affect the drivers choices. For sports cars, engine size is important, and often will be paired with whether or not the vehicle is automatic or manual. Similarly, the number of seats in a vehicle is very important for say, a large family making a booking for a holiday.

The drivers license is a very important piece of data, and must be unique, so a good piece of data to index. And lastly, when you book a vehicle, you need to have to pay some insurance towards that vehicle should you have an accident, thus combining those two into one index is logical.

The below code would create all of the indexes specified above.

```
CREATE INDEX Vehicle_Index
ON Vehicle (pricePerDay)
CREATE INDEX engine_Index
ON Model (engineSize,gearType)
CREATE INDEX seats_Index
ON Model (noOfSeats)
CREATE INDEX License_Index
ON Driver (license)
CREATE INDEX Booking_Index
ON Booking (insurance,regNumber)
```