CMPE 332 – Database Management Systems

**K-Town Car Share**

Phase 1 – Database Design and Implementation

Ryan Fredrickson - 10130487 Andrew McClelland - 10150229 Rony Besprozvanny - 10137022

Table of Contents

[Assumptions 2](#_Toc476143363)

[ER Schema 3](#_Toc476143364)

[ER Explanation 4](#_Toc476143365)

[Relational Schema 5](#_Toc476143366)

[Relational Schema Explanation 6](#_Toc476143367)

[SQL Tables with Sample Data 7](#_Toc476143368)

[Parking Location 7](#_Toc476143369)

[Car 7](#_Toc476143370)

[Car\_Rental\_History 8](#_Toc476143371)

[Car\_Maintenance\_History 8](#_Toc476143372)

[Rental\_Comment 9](#_Toc476143373)

[Reservations 9](#_Toc476143374)

[KTCS\_Member 10](#_Toc476143375)

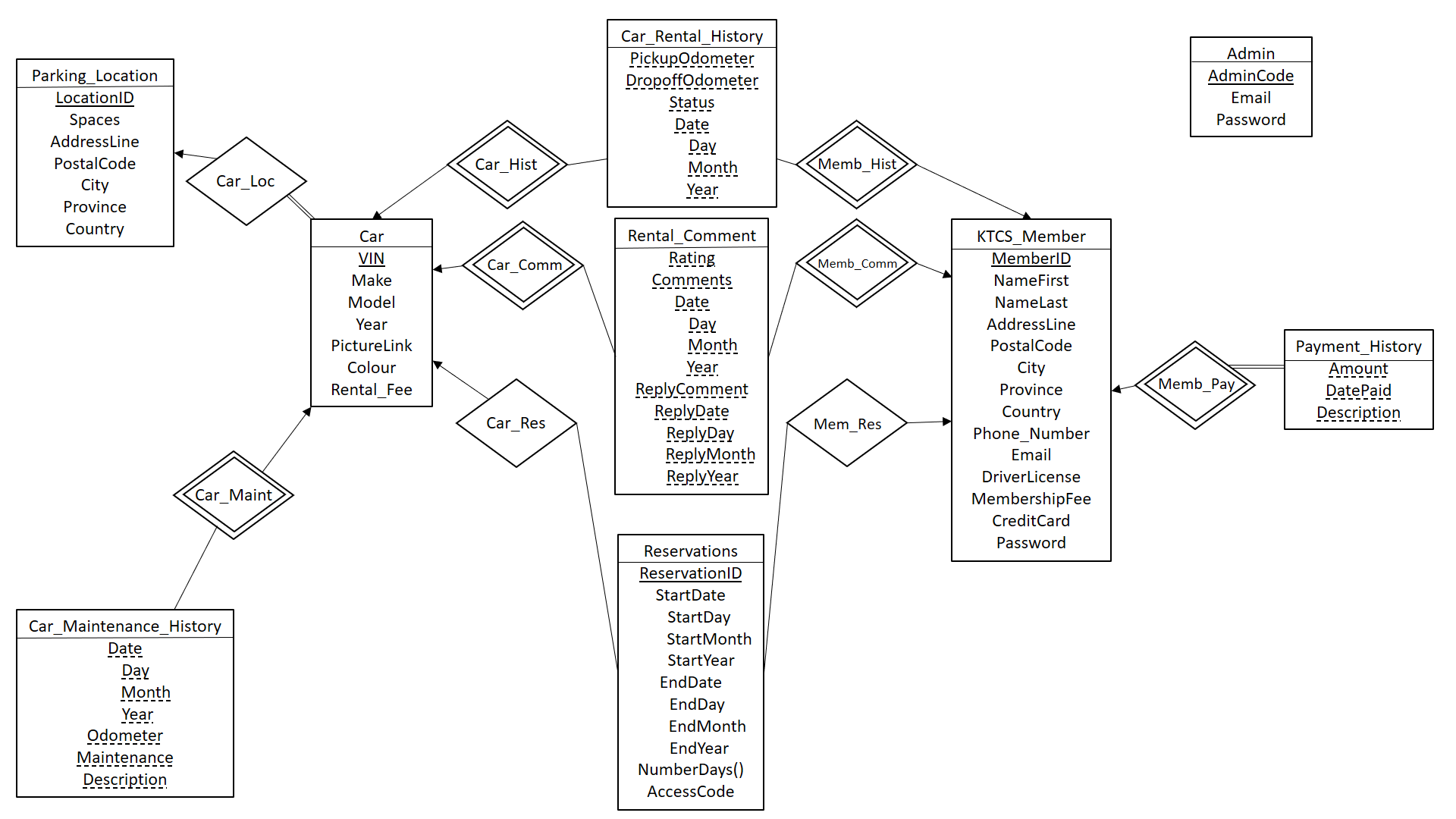
[Payment\_History 11](#_Toc476143376)

[Admin 11](#_Toc476143377)

# Assumptions

* Each member will pay for the entire month during which they sign up, regardless of signup day (so Jan 1 and Jan 29 signups are charged for all of January).
* Each member will pay for car / reservation as soon as they book the car.
* There are no cancellations / refunds.
* A car is available for the day unless it has been reserved.
* A car reserved for a day begins at 12:00 am on that day and ends 11:59 pm on the end reserved day.
* Members cannot reserve for than 14 days in advance.
* Every car is in its specified location at the start of each rental.
* Parking location is never over capacity → a location starts under or at max capacity, and is never exceeded because all cars returned to same location.
* The user fills out gas amount as part of the pickup / drop-off form.
* Replies to comments are limited to one reply and only an admin can reply.
* Car rental history is only created when a car gets rented.
* Car maintenance history is only created when a car gets sent in for maintenance.
* Assume only one picture per car, one phone number per member, one email per member, one Credit Card per member.
* Each time maintenance is performed, only one type of maintenance can be displayed.
* No Username is required – email will be used instead.

# ER Schema



## ER Explanation

To optimize the database system, the following decisions were made. These include the use of weak-entity sets and various cardinality and participation modes. The purpose of using weak-entity sets and relationships is due to the close association seen between the objects. The weak entity set object (i.e. Car\_Rental\_History) is dependent and relies on the strong entity, as it cannot exist without the Car entity set. As can be seen in the Relational Schema section, in the weak-entity relationships, the primary key of the strong entity becomes a primary key of the weak-entity set.

The ER schema above uses a variety of cardinality modes such as one-to-many/many-to-one relationships. As an example, the Reservations table has a many to one relationship with the Car and KTCS\_Member objects. This is because you can have many Reservation objects for one car and member. The many to one relationship is represented in the Relational Schema seen below by adding the primary key of the one side (i.e. The Car and KTCS Member) as attributes to the many side (i.e. Reservations). Finally, total participation is used solely for the Payment\_History to KTCS Member relationship. This is because one of the assumptions the team made is that as soon as KTCS Member is created there will be at least one Payment\_History object created which contains the monthly fee to be charged. Payment\_History objects will also be created whenever a user reserves a car.

Within the Admin table, it was decided to make it a dedicated table that is not in a relationship with anything else. This is because the Admin will only contain 3 attributes: the user name, password and admin code that is generated for the admin. Admins will sign into the website by providing all 3 fields. Admins do not have a name or any personal information associated with them and because of this the Admin table does not need to have an ISA relationship with the KTCS\_Member.

Another feature of the ER diagram seen above is the use of the derived variable ‘NumberDays()’ for the Reservations table. The purpose of this derived variable is to compute the number of days that the user will be renting the car. This will be done by getting the ‘StartDate’ and ‘EndDate’ and subtracting them to get the number of days.

# Relational Schema

## Relational Schema Explanation

Based on the ER diagram, the relational schema was created (as seen above). In terms of handling the one to many/many to one relationships, the primary key of the one side was added as an attribute to the many side. This created a foreign key relationship. As an example, the Parking\_Location shares a one to many relationship with Car so Parking\_Location’s primary key of LocationID was added as an attribute in Car.

Another note with the relational schema is the various optimizations the team made in order to make the schema more efficient. In the ER diagram the dates for the various objects (ie. Car\_Rental\_History) were a composite value comprised of day, month and year. In SQL there is a predefined date type that takes care of the various aspects of the date. Thus, in the relational schema there is no need to have day, month and year as it is just replaced by a date object.

In addition, in the ER schema there was a derived variable ‘NumberDays()’, but the derived variable is not present in the relational schema or the SQL. This is because the number of days the user is reserving the car for will be computed by querying the StartDate and EndDate and subtracting them.

In terms of constraints of various variables, this is not seen in the Relational Schema but it is present in the SQL table create statements found in the following section. Constraints were added for Rating, Maintenance and Status. The ‘Rating’ attribute constraint ensured that Rating would only be an integer from 1-4. The ‘Maintenance’ attribute constraints are that it can only take on the values of either ‘scheduled’, ‘repair’ or ‘body work’. Finally, the ‘Status’ attribute constraints are that it can only take on the values of either ‘normal’, ‘damaged’, or ‘not running’.

# SQL Tables with Sample Data

## Parking Location

CREATE TABLE parking\_location (

LocationID int(11) NOT NULL,

AddressLine varchar(40) NOT NULL,

PostalCode varchar(20) NOT NULL,

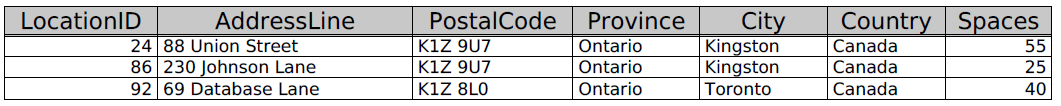
Province varchar(30) NOT NULL,

City varchar(40) NOT NULL,

Country varchar(40) NOT NULL,

Spaces int(11) NOT NULL,

primary key (LocationID));



## Car

CREATE TABLE car (

VIN int(11) NOT NULL,

Make varchar(15) NOT NULL,

Model varchar(15) NOT NULL,

Year int(11) NOT NULL,

LocationID int(11) NOT NULL,

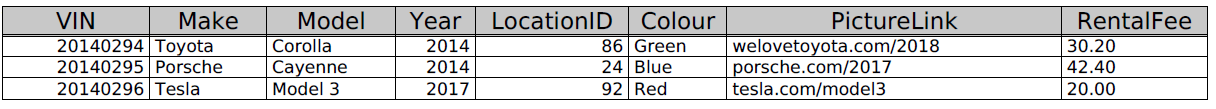
Colour varchar(10) NOT NULL,

PictureLink varchar(50) NOT NULL,

RentalFee decimal(10,2) NOT NULL,

primary key (VIN),

foreign key (LocationID) references Parking\_Location(LocationID));



## Car\_Rental\_History

CREATE TABLE car\_rental\_history (

VIN int(11) NOT NULL,

MemberID int(11) NOT NULL,

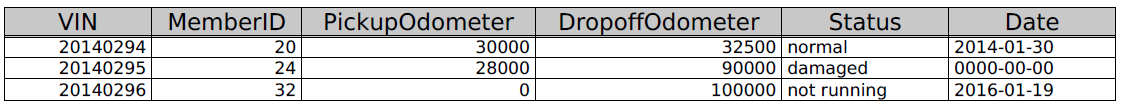
PickupOdometer int(11) NOT NULL,

DropoffOdometer int(11) NOT NULL,

Status varchar(11) NOT NULL CHECK (Status = 'normal' OR Status = 'damaged' OR Status = 'not running') ,

Date date NOT NULL

primary key (VIN, MemberID, PickupOdometer, DropoffOdometer, Status, Date));



## Car\_Maintenance\_History

CREATE TABLE car\_maintenance\_history (

VIN int(11) NOT NULL,

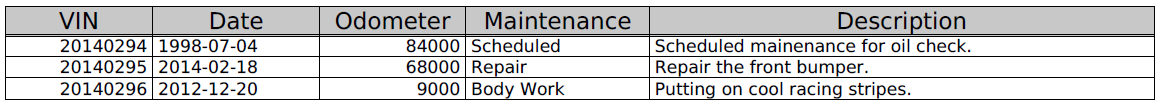
Date date NOT NULL,

Odometer int(11) NOT NULL,

Maintenance varchar(40) NOT NULL CHECK (Maintenance = 'scheduled' OR Maintenance = 'repair' OR Maintenance = 'body work'),

Description varchar(80) NOT NULL,

primary key (VIN, Date, Odometer, Maintenance, Description));



## Rental\_Comment

CREATE TABLE rental\_comment (

VIN int(11) NOT NULL,

MemberID int(11) NOT NULL,

Rating int(11) NOT NULL CHECK (Rating > 0 AND Rating < 5),

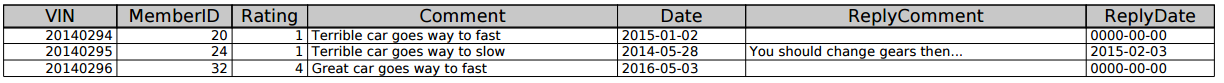
Comment varchar(500) NOT NULL,

Date date NOT NULL,

ReplyComment varchar(500) NOT NULL,

ReplyDate date NOT NULL,

primary key (VIN, MemberID, Rating, Comment, Date, ReplyComment, ReplyDate));



## Reservations

CREATE TABLE reservations (

ReservationID int(11) NOT NULL,

MemberID int(11) NOT NULL,

VIN int(11) NOT NULL,

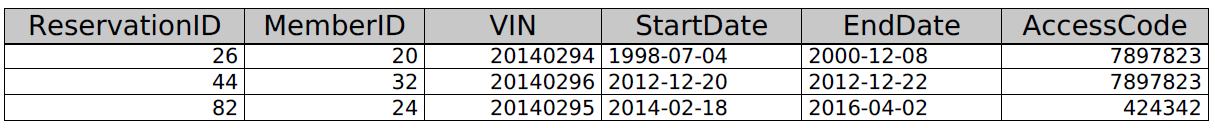
StartDate date NOT NULL,

EndDate date NOT NULL,

AccessCode int(11) NOT NULL

primary key (ReservationID),

foreign key (VIN) references Car (VIN),

 foreign key (MemberID) references KTCS\_Member(MemberID));

## KTCS\_Member

CREATE TABLE ktcs\_member (

MemberID int(11) NOT NULL,

NameFirst varchar(40) NOT NULL,

NameLast varchar(40) NOT NULL,

AddressLine varchar(40) NOT NULL,

PostalCode varchar(20) NOT NULL,

Province varchar(30) NOT NULL,

City varchar(40) NOT NULL,

Country varchar(40) NOT NULL,

PhoneNumber varchar(10) NOT NULL,

Email varchar(100) NOT NULL,

DriverLicense varchar(100) NOT NULL,

MembershipFee decimal(10,2) NOT NULL,

Password varchar(20) NOT NULL,

Credit\_Card varchar(20) NOT NULL,

primary key (MemberID));

*Please note*: the data dump for the ‘KTCS\_Member’ table had too many attributes (14) to fit on one page. Therefore, we replicated the large 14-column output into two tables in order to fit them on the page. All values in the tables below are identical to the actual MySQL data dump for the ‘KTCS\_Member’ table.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **MemberID** | **NameFirst** | **NameLast** | **AddressLine** | **PostalCode** | **Province** | **City** |
| 20 | Andrew | Hello | 10 Hello Lane | K7L1V2 | Ontario | Kingston |
| 24 | Ryan | Freddy | 10 Goodbye Lane | K7L8F3 | Ontario | Kingston |
| 32 | Rony | Bes | 10 Database Lane | K7H8Z3 | Ontario | Toronto |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Country** | **PhoneNumber** | **Email** | **DriverLicense** | **MembershipFee** | **Password** | **Credit\_Card** |
| Canada | 647111111 | [hello@hello.com](mailto:hello@hello.com) | B2746-1026-336-3048 | 30.00 | Ilikedogs | 5301063177596486 |
| Canada | 6472345678 | [bye@bye.com](mailto:bye@bye.com) | C9384-1026-232-3048 | 30.00 | Ilikecats | 5231094377093486 |
| Canada | 6478439900 | dbms@dbms.com | F9020-3231-333-9809 | 30.00 | ilikebirds | 5237483277093486 |

## Payment\_History

CREATE TABLE payment\_history (

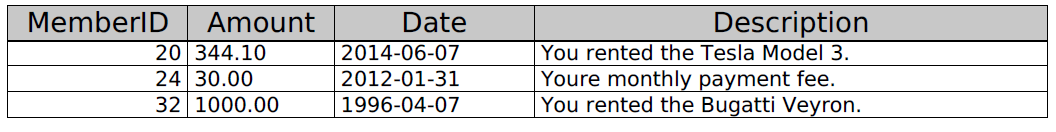
MemberID int(11) NOT NULL,

Amount decimal(10,2) NOT NULL,

Date date NOT NULL,

Description varchar(100) NOT NULL,

primary key (MemberID, Amount, Date, Description));



## Admin

CREATE TABLE admin (

AdminCode int(11) NOT NULL,

Email varchar(100) NOT NULL,

Password varchar(20) NOT NULL,

primary key (AdminCode));

