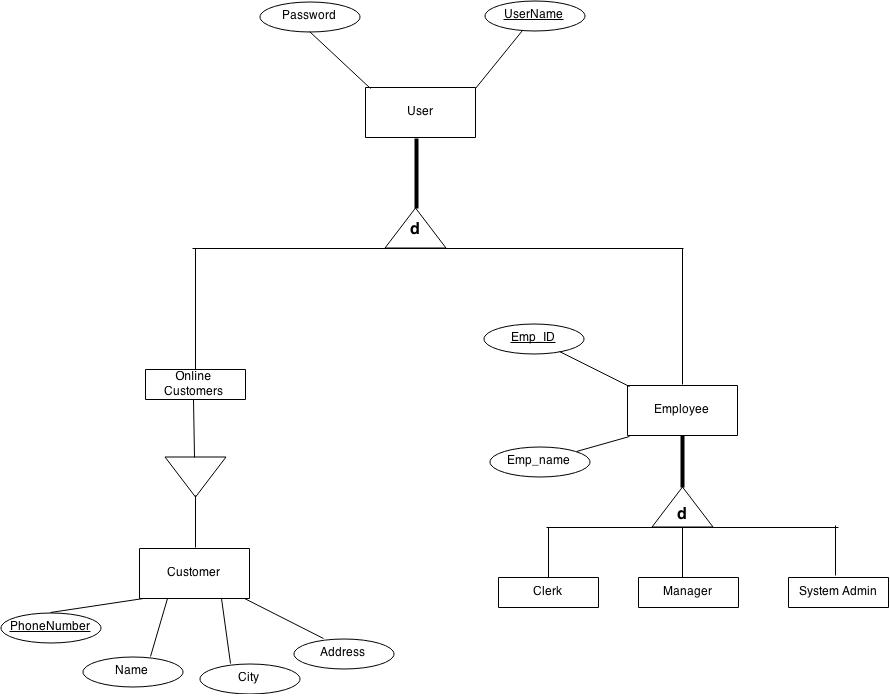
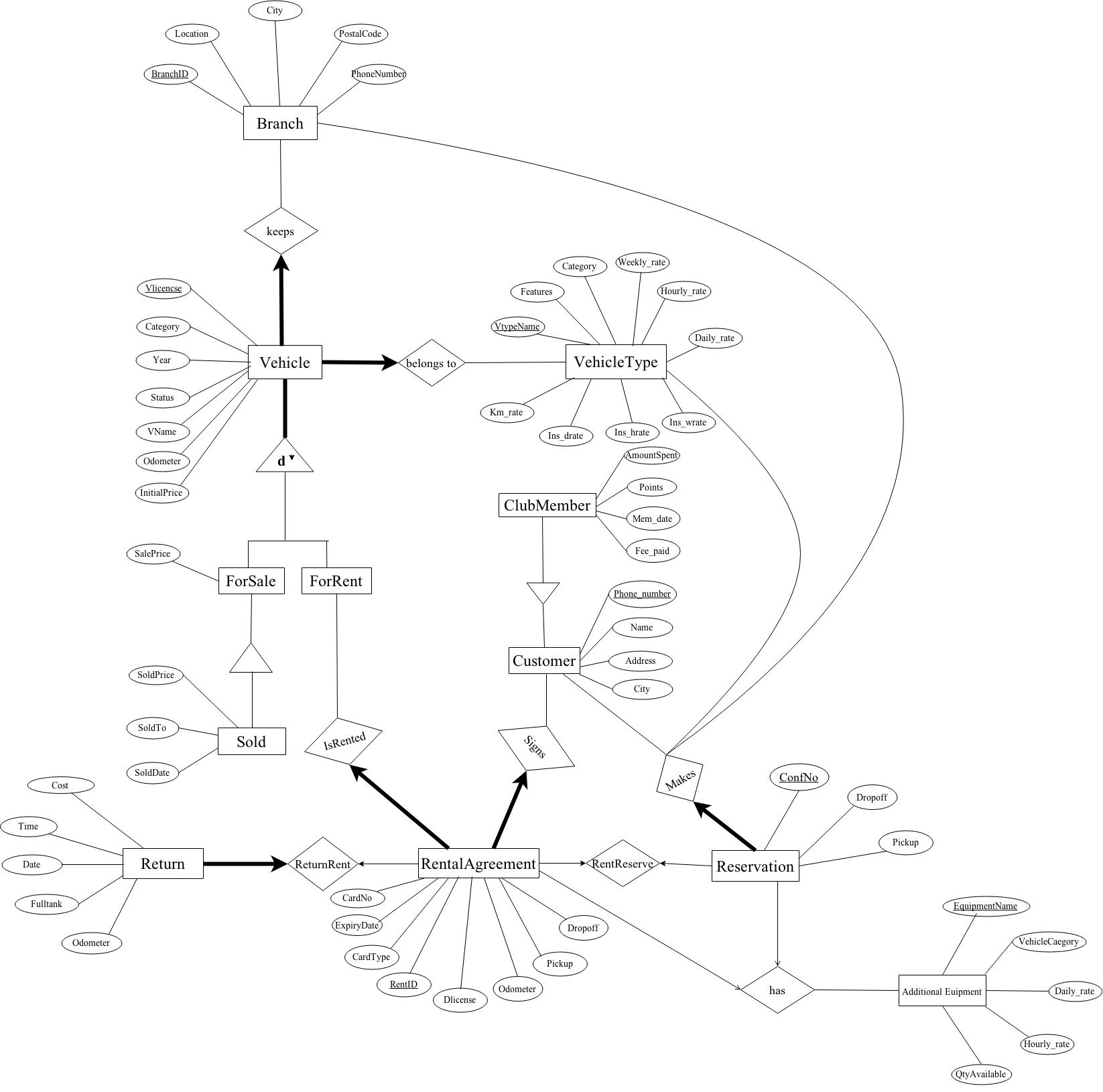
**DATABASE DESIGN:**

**ER Diagram:**

**ER Diagram1:**



**ER Diagram2:**



**ER Diagram to Tables:**

**First set of Tables**

The following is a list of the tables and the constraints obtained from the ERD-to-Tables process.

**User Hierarchy:**

From the ER diagram 1, it is a total disjoint hierarchy. Super Class doesn’t participate in any of the relations such that there are no table for super class and one table for each subclass containing all the super class and sub class attributes.

* **Employee Hierarchy:**

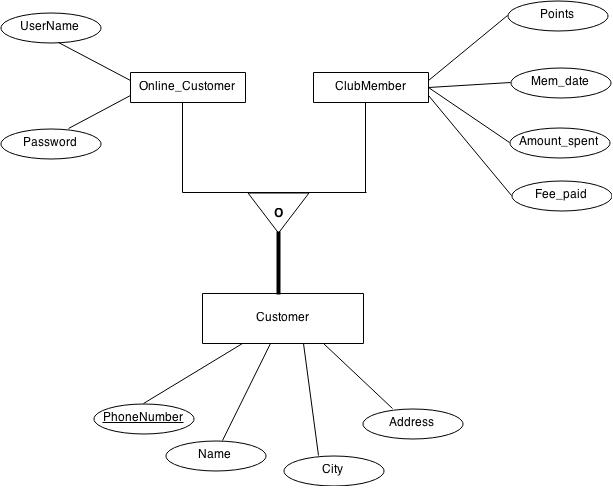
From the ER diagram 1, ISA relationship between them and their parent set is disjoint and total. Therefore, we don't need to create new tables for them. We just need to add an attribute “**Type**” in the Employee set to distinguish the three subsets.

**Employees** (Emp\_id, Emp\_name, Username, Password, Type)

* Represents: entity set Employees, User, clerk, manager, system admin.
* Primary key: Emp\_id
* Constraints:
* Type can be Clerk, Manager and System Administrator.

**Customer Hierarchy:**

From the ER diagram 1 and 2, it is a total Overlapping hierarchy.



**Customer** (PhoneNumber, Name, Address, City)

* Represents: entity set Customer.
* Primary key: Phone\_number
* Constraints:
* Customer can be Online Customer, Club Member.

**Online\_customer** (Phone\_number, Username, Password)

* Represents: entity set Online\_customer, Customer,User.
* Primary key: Phone\_number
* Foreign Keys:
* (Phone\_number) references Customer

**Clubmember** (Phone\_number, Points, Mem\_date, Amount\_spent,Fee\_paid)

* Represents: entity set Clubmember, Customer.
* Primary key: Phone\_number
* Foreign Keys:
* (Phone\_number) references Customer

**Branch** (BranchID, Location, City, Postal\_code, Phone\_number)

* Represents: entity set Branch.
* Primary key: BranchID

**Vehicletype** (Vtype\_name, Features, Weekly\_rate, Daily\_rate, Hourly\_rate, Km\_rate, Ins\_wrate, Ins\_drate, Ins\_hrate)

* Represents: entity set Vehicletype.
* Primary key: Vtype\_name

**Vehicle Hierarchy:**

From the ER diagram 2, it is a total disjoint hierarchy.

**Vehicle**(Vlicense, Vname, Vtype\_name, Category, Year, Initial\_price, Odometer, BranchID, ForRentFlag)

* Represents: entity set Vehicle, ForRent, ForSale and relationship sets belongs to and keeps.
* Primary key: Vlicense
* Foreign Keys:
* (BranchID) references Branch
* (Vtype\_name) references Vehicletype
* Constraints:
* each vehicle should belongs to a vehicle type.
* each vehicle must be kept in a branch.
* ForRentFlag indicates whether vehicle is ForRent or not(ForSale).

**Forsalevehicles** (Vlicense, SalePrice, SoldFlag, SoldPrice, SoldDate, SoldTo)

* Represents: entity set Forsalevehicles, Vehicle, Sold.
* Primary key: Phone\_number
* Foreign Keys:
* (Vlicense) references Vehicle
* Constraints:
* SoldFlag indicates whether the vehicle is sold or not.

**Additional\_equipment** (EquipmentName, VehicleCategory, Daily\_rate, Hourly\_rate, AvailabaleQty)

* Represents: entity set Additional\_equipment.
* Primary key: EquipmentName

**Reservation** (Confno, Phone\_number, Vtype\_name, Vlicense, BranchID, Pickup\_time, Dropoff\_time, Equipment)

* Represents: entity set Reservation and relationship sets Makes, RentReserve and Can Have.
* Primary key: Confno
* Foreign Keys:
* (Phone\_number) references Customer
* ( Vtype\_name ) references VehicleType
* ( BranchID )references Branch
* ( Vlicense )references Vehicle
* ( Equipment ) references Additional\_equipment ( EquipmentName )
* Constraints:
* each reservation is made by a Customer.

**Rentalagreement** (RentId, ConfNo, Phone\_number, Dlicense, Vlicense, CardNo, ExpiryDate, CardType, Equipment, Odometer, Pickup\_time, Dropoff\_time)

* Represents: entity set Rentalagreement and relationship sets Signs, IsRented, RentReserve, has and ReturnRent.
* Primary key: RentId
* Foreign Keys:
* ( ConfNo )references Reservation
* ( Phone\_number )references Customer
* ( Vlicense ) references Vehicle
* ( Equipment ) references Additional\_equipment ( EquipmentName )
* Constraints:
* each Rentalagreement is signed by a Customer.
* each Rentalagreement contains a associated vehicle.

**Returnvehicle** (RentId, Dropoff\_time, Fulltank, Odometer, Cost)

* Represents: entity set Returnvehicle, Rentalagreement.
* Primary key: RentId
* Foreign Keys:
* (RentId) references Rentalagreement

**Functional Dependencies:**

The following is a list of FD's for each table:

**Customer** (Phone\_number, Name, Address, City)

* Phone\_number --> Name, Address, City (Key)

**Online\_customer** (Phone\_number, Username, Password)

* Phone\_number --> Username, Password (Key)

**Clubmember** (Phone\_number, Points, Mem\_date, Amount\_spent)

* Phone\_number 🡪Points, Mem\_date, Amount\_spent (Key)

**Additional\_equipment** (EquipmentName, VehicleCategory, Daily\_rate, Hourly\_rate, AvailabaleQty)

* EquipmentName 🡪VehicleCategory, Daily\_rate, Hourly\_rate, AvailabaleQty (Key)

**Branch** (BranchID, Location, City, Postal\_code, Phone\_number)

* BranchID 🡪Location, City, Postal\_code, Phone\_number (Key)

**Employees** (Emp\_id, Emp\_name, Username, Password, Type)

* Emp\_id 🡪Emp\_name, Username, Password, Type (Key)

**Forsalevehicles** (Vlicense, SalePrice, SoldFlag, SoldPrice, SoldDate, SoldTo)

* Vlicense 🡪 SalePrice, SoldFlag, SoldPrice, SoldDate, SoldTo (Key)

**Rentalagreement** (RentId, ConfNo, Phone\_number, Dlicense, Vlicense, CardNo, ExpiryDate, CardType, Equipment, Odometer, Pickup\_time, Dropoff\_time)

* RentId 🡪 ConfNo, Phone\_number, Dlicense, Vlicense, CardNo, ExpiryDate, CardType, Equipment, Odometer, Pickup\_time, Dropoff\_time (Key)

**Reservation** (Confno, Phone\_number, Vtype\_name, Vlicense, BranchID, Pickup\_time, Dropoff\_time, Equipment)

* Confno🡪Phone\_number, Vtype\_name, Vlicense, BranchID, Pickup\_time, Dropoff\_time, Equipment (Key)

**Returnvehicle** (RentId, Dropoff\_time, Fulltank, Odometer, Cost)

* RentId🡪Dropoff\_time, Fulltank, Odometer, Cost (Key)

**Vehicle** (Vlicense, Vname, Vtype\_name, Year, Initial\_price, Odometer, BranchID, ForRentFlag, Status)

* Vlicense 🡪 Vname, Vtype\_name, Year, Initial\_price, Odometer, BranchID, ForRentFlag, Status (Key)

**Vehicletype** (Vtype\_name, Features, Weekly\_rate, Daily\_rate, Hourly\_rate, Km\_rate, Ins\_wrate, Ins\_drate, Ins\_hrate)

* Vtype\_name --> Features, Weekly\_rate, Daily\_rate, Hourly\_rate, Km\_rate, Ins\_wrate, Ins\_drate, Ins\_hrate (Key)

**Reality Check:**

After Normalization and applying the reality check our final set of tables are,

**Final Set of tables:**

**Customer** (Phone\_number, Name, Address, City, Clubmember, Roadstar)

**Online\_customer** (Phone\_number, Username, Password)

**Clubmember** (Phone\_number, Points, Mem\_date, Amount\_spent)

**Additional\_equipment** (EquipmentName, VehicleCategory, Daily\_rate, Hourly\_rate, AvailabaleQty)

**Branch** (BranchID, Location, City, Postal\_code, Phone\_number)

**Customer** (Phone\_number, Name, Address, City, Clubmember, Roadstar)

**Employees** (Emp\_id, Emp\_name, Username, Password, Type)

**Forsalevehicles** (Vlicense, SalePrice, SoldFlag, SoldPrice, SoldDate, SoldTo)

**Rentalagreement** (RentId, ConfNo, Phone\_number, Dlicense, Vlicense, CardNo, ExpiryDate, CardType, Equipment, Odometer, Pickup\_time, Dropoff\_time)

**Reservation** (Confno, Phone\_number, Vtype\_name, Vlicense, BranchID, Pickup\_time, Dropoff\_time, Equipment)

**Returnvehicle** (RentId, Dropoff\_time, Fulltank, Odometer, Cost)

**Vehicle** (Vlicense, Vname, Vtype\_name, Category, Year, Initial\_price, Odometer, BranchID, ForRentFlag, Status)

**Vehicletype** (Vtype\_name, Features,Category, Weekly\_rate, Daily\_rate, Hourly\_rate, Km\_rate, Ins\_wrate, Ins\_drate, Ins\_hrate)

**SQL Definitions:**

**Customer:**

CREATE TABLE Customer (

Phone\_number varchar(15) PRIMARY KEY,

Name Varchar(30),

Address varchar(30),

City varchar(20),

Clubmember tinyint(1),

Roadstar tinyint(1))

ENGINE = InnoDB;

**Online\_Customer:**

CREATE TABLE Online\_Customer (

Phone\_number varchar(15) PRIMARY KEY,

Username varchar(20) not null unique,

Password varchar(20),

FOREIGN KEY ( Phone\_number ) REFERENCES Customer ( Phone\_number )

ON DELETE CASCADE ON UPDATE CASCADE)

ENGINE = InnoDB;

**ClubMember :**

CREATE TABLE ClubMember (

Phone\_number varchar(15) primary key,

Points int DEFAULT 500,

Mem\_date DATETIME,

Amount\_spent float,

Fee\_paid tinyint(1),

FOREIGN KEY ( Phone\_number ) REFERENCES Customer ( Phone\_number )

ON DELETE CASCADE ON UPDATE CASCADE)

ENGINE = InnoDB;

**Employees :**

CREATE TABLE Employees (

Emp\_id int PRIMARY KEY AUTO\_INCREMENT,

Emp\_name varchar(30),

Username varchar(30) not null unique,

Password varchar(15) not null,

Type varchar(10))

ENGINE = InnoDB;

**Branch :**

CREATE TABLE Branch (

BranchID int primary key,

Location varchar(64),

City varchar(20),

Postal\_code varchar(15),

Phone\_number varchar(15))

ENGINE = InnoDB;

**VehicleType:**

CREATE TABLE VehicleType(

Vtype\_name varchar(20) primary key,

Features varchar(30),

Category varchar(20),

Weekly\_rate float not null,

Daily\_rate float not null,

Hourly\_rate float not null,

Km\_rate float not null,

Ins\_wrate float not null,

Ins\_drate float not null,

Ins\_hrate float not null)

ENGINE = InnoDB;

**Vehicle:**

CREATE TABLE Vehicle(

Vlicense int primary key,

Vname varchar(30) NOT NULL,

Vtype\_name varchar(20) not null,

Category varchar(20) not null,

Year int,

Initial\_price double,

Odometer float,

BranchID int not null,

ForRentFlag tinyint(1) not null,

Status tinyint(1) not null,

FOREIGN KEY ( BranchID )references Branch ( BranchID ) ON DELETE NO ACTION ON UPDATE CASCADE,

FOREIGN KEY ( Vtype\_name ) REFERENCES VehicleType ( Vtype\_name ) ON DELETE NO ACTION ON UPDATE CASCADE)

ENGINE = InnoDB;

**ForsaleVehicles:**

CREATE TABLE ForsaleVehicles(

Vlicense int primary key,

SalePrice double,

SoldFlag tinyint(1) not null,

SoldPrice double,

SoldDate DATE,

SoldTo varchar(20),

FOREIGN KEY ( Vlicense ) REFERENCES Vehicle ( Vlicense ))

ENGINE = InnoDB;

**Additional\_equipment** :

CREATE TABLE Additional\_equipment(

EquipmentName varchar(20) primary key,

VehicleCategory varchar(15) not null,

Daily\_rate float not null,

Hourly\_rate float not null,

AvailabaleQty int)

ENGINE = InnoDB;

**Reservation:**

CREATE TABLE Reservation(

Confno int primary key AUTO\_INCREMENT,

Phone\_number varchar(15) not null,

Vtype\_name varchar(20) not null,

Vlicense int not null,

BranchID int not null,

Pickup\_time datetime,

Dropoff\_time datetime,

Equipment varchar(20),

FOREIGN KEY ( Phone\_number )references Customer ( Phone\_number ) ON DELETE NO ACTION ON UPDATE CASCADE,

FOREIGN KEY ( Vtype\_name ) REFERENCES VehicleType ( Vtype\_name ) ON DELETE NO ACTION ON UPDATE CASCADE,

FOREIGN KEY ( BranchID )references Branch ( BranchID ) ON DELETE NO ACTION ON UPDATE CASCADE,

FOREIGN KEY ( Vlicense )references Vehicle ( Vlicense ) ON DELETE NO ACTION ON UPDATE CASCADE,

FOREIGN KEY ( Equipment ) REFERENCES Additional\_equipment ( EquipmentName ))

ENGINE = InnoDB;

**RentalAgreement:**

CREATE TABLE RentalAgreement(

RentId int primary key AUTO\_INCREMENT,

ConfNo int,

Phone\_number varchar(15) not null,

Dlicense int,

Vlicense int not null,

CardNo int,

ExpiryDate date,

CardType Varchar(20),

Equipment varchar(20),

Odometer float not null,

Pickup\_time datetime,

Dropoff\_time datetime,

FOREIGN KEY ( ConfNo )references Reservation ( ConfNo ),

FOREIGN KEY ( Phone\_number )references Customer ( Phone\_number ) ON DELETE no action ON UPDATE CASCADE,

FOREIGN KEY ( Vlicense ) REFERENCES Vehicle ( Vlicense ) ON DELETE no action ON UPDATE CASCADE,

FOREIGN KEY ( Equipment ) REFERENCES Additional\_equipment ( EquipmentName ))

ENGINE = InnoDB;

**ReturnVehicle:**

CREATE TABLE ReturnVehicle(

RentId int primary key,

Dropoff\_time datetime,

Fulltank tinyint(1),

Odometer float,

Cost float,

FOREIGN KEY ( RentId ) REFERENCES RentalAgreement (RentId) ON DELETE CASCADE ON UPDATE CASCADE)

ENGINE = InnoDB;

For handling the club membership renewal process, have created the trigger and schedule the events, similar to **CRON** jobs.

**Triggers:**

/\*Trigger Club\_Membership \*/

DELIMITER $$

CREATE TRIGGER club\_member\_update

AFTER UPDATE ON ClubMember

FOR EACH ROW BEGIN

UPDATE Customer

SET Clubmember = new.Fee\_paid

WHERE Phone\_number = old.Phone\_number;

END $$

DELIMITER ;

ClubMember(**Fee\_paid**) 🡪 Denotes whether the customer has paid the membership annual fee.

Customer(**Clubmember**) 🡪 Denotes the ClubMember status of the customer.

Columns ClubMember(**Fee\_paid**) and Customer(**Clubmember**) are been in sync such that the changes made in the Fee\_paid column will reflect back in the Customer(**Clubmember**) table with the help of the above mentioned trigger.

**Event Scheduler (CRON job):**

SET GLOBAL event\_scheduler = ON;

/\*Event Club\_Membership\*/

DELIMITER $$

CREATE EVENT club\_membership

ON SCHEDULE EVERY 1 day

STARTS CURRENT\_TIMESTAMP

ENDS '2016-04-17 00:00.00'

DO BEGIN

UPDATE ClubMember set Fee\_paid=0,Mem\_date=sysdate()

where floor(abs(((DATEDIFF(sysdate(),Mem\_date))/365)))>=1;

END $$

DELIMITER ;

Have Scheduled an event (cron job) such that every day it will runs the job to check the membership date and sysdate() interval, once the cycle got completed(1 year), it will update the ClubMember(**Fee\_paid**) column to 0 and the mem\_date to sysdate().

With the help of the trigger, the corresponding value will be reflected back on the Customer(**Clubmember**) table. So that it will be taken in to consideration while evaluating the cost.

**STORED PROCEDURE:**

Have Created the stored procedure to kill the processes, in-order to handle the active number of connections.

/\*Stored procedure kill\_process\*/

DELIMITER $$

CREATE PROCEDURE kill\_process()

BEGIN

DECLARE a VARCHAR(20);

DECLARE no\_more\_rows BOOLEAN;

DECLARE c\_kill CURSOR FOR SELECT CONCAT('KILL ',id,';') FROM INFORMATION\_SCHEMA.PROCESSLIST where db like '%team06%' and command not like '%Query%';

DECLARE CONTINUE HANDLER FOR NOT FOUND

SET no\_more\_rows = true;

OPEN c\_kill;

get\_process: LOOP

FETCH c\_kill INTO a;

IF no\_more\_rows THEN

CLOSE c\_kill;

LEAVE get\_process;

END IF;

SET @s = a;

PREPARE stmt FROM @s;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

END LOOP get\_process;

END$$

DELIMITER ;