Cars4u Rental (database)

- Entity list with attributes cardinality and participation
- Er diagram
- Database schema
- Notes

Entity/Attributes List

1. Address

- **PK:** Address_ID
- A_StreetName
- A_StreetNo
- A_city
- A_PostalCode

One Address can be assigned to 1 or more Employees
One Address can be assigned to 1 or more Customers
One Address can be assigned to 1 or many Branches

This is an entity that represents the address for branch, customer, and employee

——

2. Customer's Address

- **PK:** (Customer_ID, Address_ID)

One Customer's address can only have one address

The use of this entity is to assign the unique address of the customer to the customer entity

——————————————————————————————————————

3. **Customer**

- **PK:** Customer_ID
- CU_Fname
- CU_Lname
- CU_Email
- CU_TelNo
- CU_Driver'sLicenseNo (*)

One customer can only have one customer address
One customer could have 1 or more cars

This is a entity dedicated for customer info

————————————————————————————————

4. **Branch Address**

- **PK:** (Address_ID, Branch_ID)

One branch address can only have one address

The use of this entity is to assign the unique address of the branch to the branch entity

————————————————————————————————

5. **Branch**

- **PK:** Branch_ID
- BS_No.OfEmployees
- BS_No.OfCars

One branch can have zero or more cars
One branch has many employees

This is a entity dedicated for branch info

————————————————————————————————

6. **Class_Info**

- **PK:** Class_ID
- CI_ClassPrice
- CI_ClassType

One class can have zero or many cars

The use of this entity is to normalize the car entity and separate the class info from the car entity
————————————————————————————————

7. **Car:**

- **PK:** LicensePlate (char)
- C_Model
- C_ModelStoreLocation
- C_Make

- C_YearMade
- C_Colour
- C_No.OfSeats
- C_Vin (id no.)

A car can exist in only one branch
A car can have only 1 customer at time
Zero or many cars exist in one class

This entity is about the car details

——————————————————————————————————————

9. Price:

- **PK:** Price_ID
- P_Employee(Y/N)
- P_Less2Weeks(Y/N)
- P_CertaiW.Promotion(Y/N)
- P_Upgrade(Y/N)
- P_RentTimeSpan(Y/N)
- P_DropOffCharge(Y/N)
- P_(Cars4U)Employee(Y/N)

One or many prices can be assigned to one class
One price can be assigned to one receipt

This is the pricing entity which enfolds many types of discounts

——————————————————————————————————————

10. Receipt:

- **PK:** Receipt_ID
- R_RentOutDate
- R_RentInDate
- R_RentOutBranch
- R_RentInBranch
- R_MilesBefore
- R_MilesAfter
- R_FuelBefore (readings by quarters)
- R_FuelAfter (readings by quarters)
- R_RentTimeSpan
- R_RentStatus

One or many receipt can be given to one customer

One receipt can have only one price at a time

Entity to gather the info of rentals

——

11. Employee's Address

- **PK:** (Customer_ID, Address_ID)

One Employee address can only have one address

The use of this entity is to assign the unique address of the employee to the employee entity

——

12. **Employee**

- **PK:** EmployeeProfile_ID
- E_DriversLiscence
- E_Fname
- E_Lname
- E_Email
- E_TelNo
- E_Role (in huddersfield headquarters —> 1 managing director, 2 directors, 1 sales, rest marketing)

One employee can have zero or more customers
One employee can have one address
One Employee can work in one branch

This entity is for employee info

——

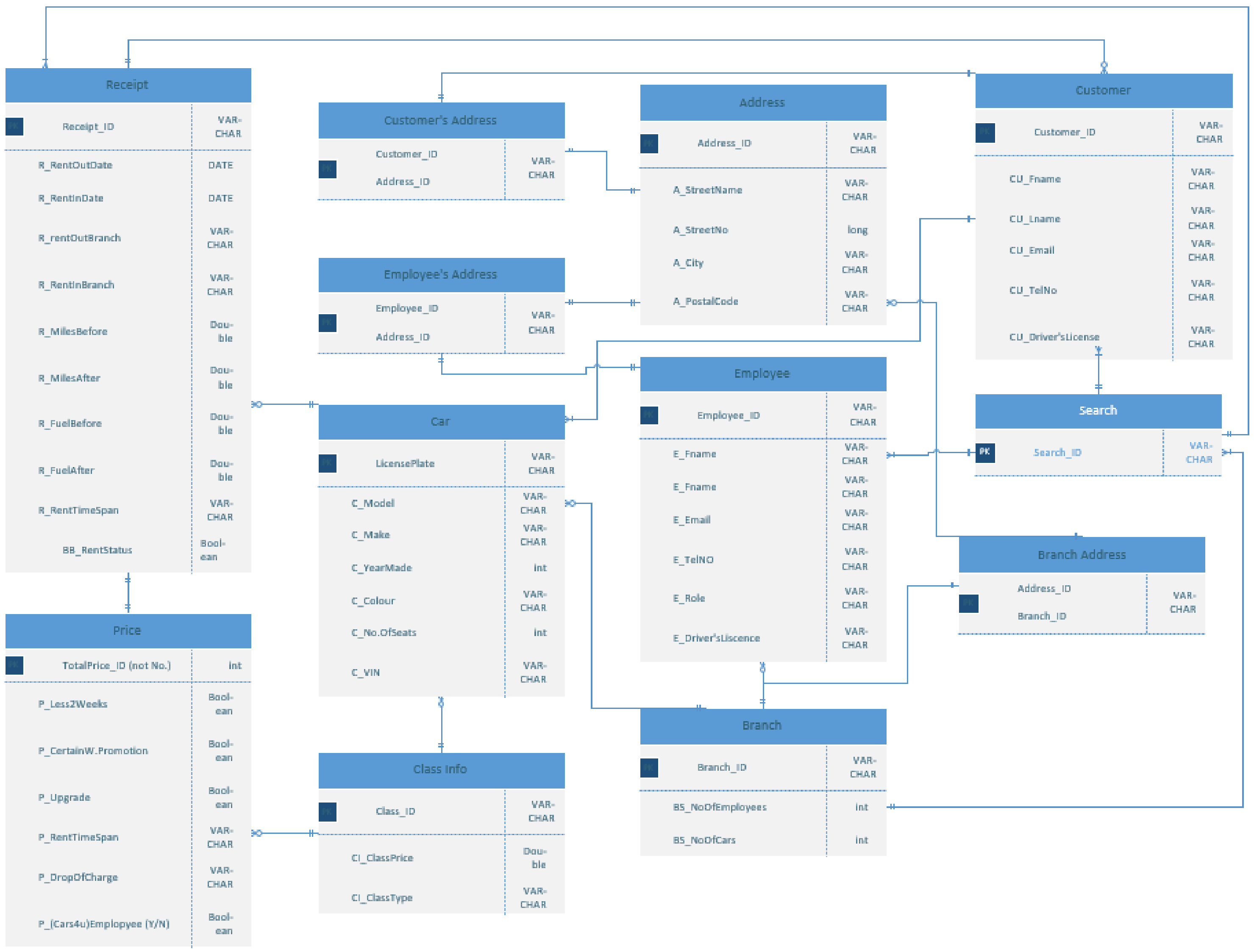
13. Search

- **Pk:** Search_ID

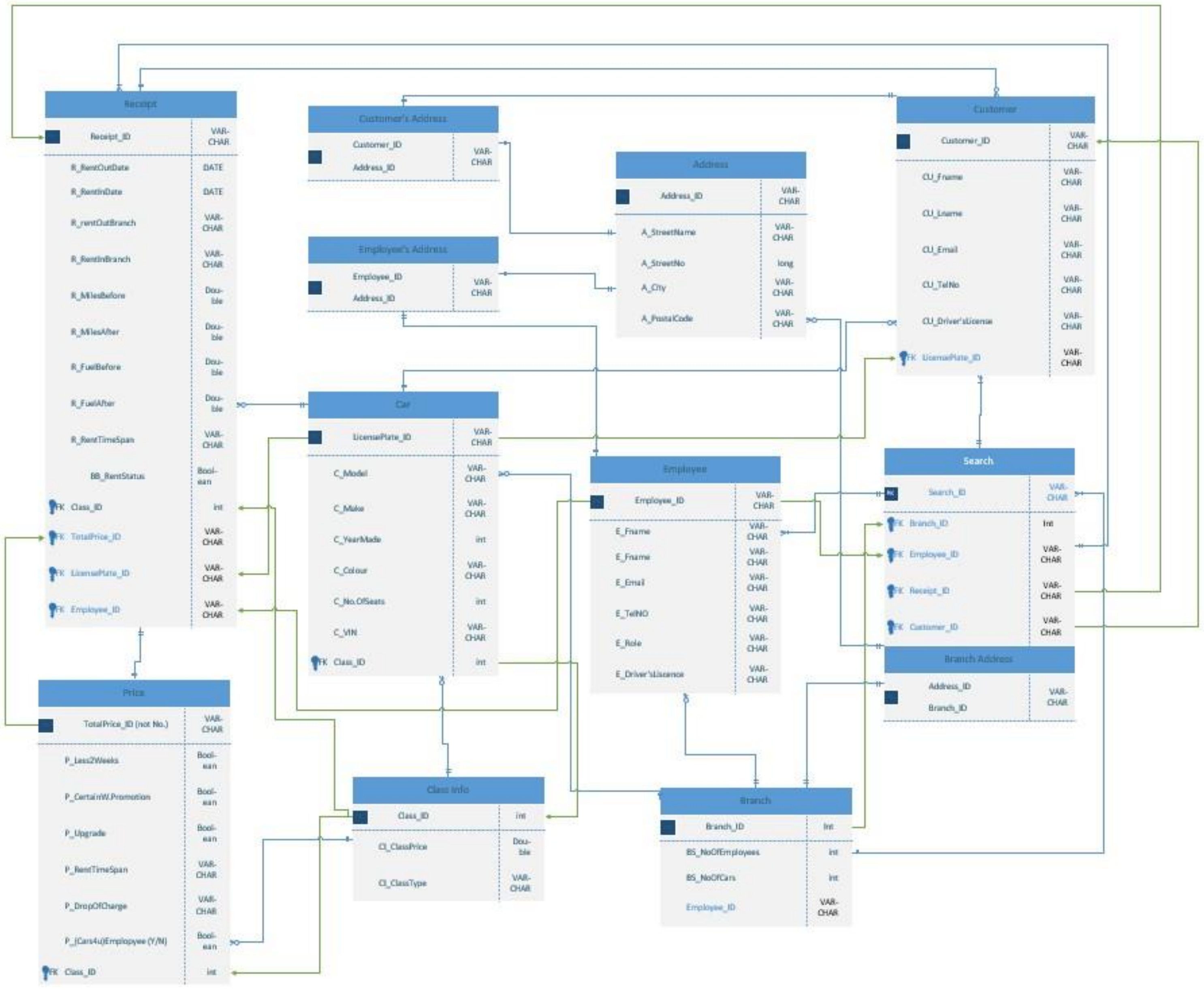
One search can be for zero or many branches
One search can be for zero or more employees
One search can be for zero or more receipts
One search can be for zero or more customers

This entity will hold foreign keys from other entities which will make it easier to reach

Entity Relationship diagram (normalized)



Database Schema



Assumptions:

- I started with creating an entity for address because I realized I have many other entities that would share the same attributes and this is not efficient, so what I did was that I created the address entity to share it with the branch address, customer's address, and employee's address. And then the branch address will be a participant in the branch entity. The same thing will occur with both the employee and customer. This not only would be more efficient, but it will prevent the insert, add, delete anomalies from occurring.

- I Also created a class entity which would have the class price and type because it being in the same entity as car causes anomalies and doesn't normalize it to the 3rd level.

- An efficient move would be to merge the employee's address and the customer's address and make it a singular table. However, this wouldn't be the best move considering the insert, delete, update anomalies.

- The price entity I faced a problem because the employee discount clashed with other discounts which made dependencies that need to be the way they are to ensure the best usage and prevent data loss or scattering. So, I added the employee discount as a Boolean data type for the performance to be better.

- A Search entity was created with multiple foreign keys to make it possible and efficient for the user to search data easier and faster

---

- *"Address"* is an entity that represents the address for branch, customer, and employee
- The use of this *"customer's address"* is to assign the unique address of the customer to the customer entity

- *“Customer”* is an entity that holds customer info
- The use of *“Branch Address”* entity is to assign the unique address of the branch to the branch utility
- *“Branch”* is an entity that holds customer info
- The use of *“Class Info”* entity is normalize the car entity and separate the class info from the car entity
- *“Car”* is an entity that stores information for the car
- *“price” entity has all the discount options*
- *“Receipt” entity for info of rentals*
- *“Employee’s Address” entity assigns the unique address of the employee to the entity*
- *“search” entity will make it easier to reach wanted info from other entities*

Primary keys:

- Address_ID in Address
- Customer_ID and Address_ID in Customer’s address
- Customer_ID in Customer
- Address_ID and Branch_ID in Branch Address
- Branch_ID in Branch
- Class_ID in Class Info
- Licence_ID in Car
- Price_ID in Price
- Receipt_ID in Receipt
- Customer_ID and Address_ID in Employee’s Address
- EmployeeProfile_ID in Employee
- Search_ID in Search

Foreign Keys:

- PK Branch_ID is a foreign key in Search
- PK Employee_ID is a foreign key in Search
- PK ReceiptID is a foreign key in Search
- PK Customer_ID is a foreign key in Search

- PK Employee_ID is a foreign key in Receipt
- PK LicencePlate_ID is a foreign key in Receipt
- PK TotalPrice_ID is a foreign key in Receipt
- PK Class_ID is a foreign key in Receipt
- PK Class_ID is a foreign key in Car
- PK Class_ID is a foreign key in Price
- PK LicencePlate_ID is a foreign key in Customer

Surrogate key:

- Search_ID
- Address_ID
- TotalPrice_ID
- Branch_ID
- Class_ID

Compound keys:

- Customer_ID and Address_ID for the customer’s address table
- Employee_ID and Address_ID for the Employee’s address table
- Address_ID and Branch_ID for the customer’s address table