1. UML Diagram

Our UML diagram contains more than five entities and includes at least two types of relationships.

2. Assumptions

The “rents” relationship is 1:1 from the Rentals relation to the Customers relation because each rental is booked under exactly one customer. The same relationship is 0..* from Customers to Rentals because customers can have 0 or multiple rentals over a timespan. We have modeled Rentals as an entity rather than an attribute of Customers to avoid redundancy within the Customers relation since a customer can have multiple rentals.

Similarly, the GarageLocation entity is related to Garages as a 1..* relationship to allow for multiple garages to be located in the same city. The relationship between Garages and Garage Location is 1..1, since each garage is assigned exactly one location. This relationship is called “located”. Here, latitude and longitude together make up the primary key for GarageLocation, so that each garage may be uniquely identified by the pair of values for latitude and longitude. Garages is a separate entity from GarageLocation because there could be multiple garages in the same city and redundancy should be avoided.

The Rentals and CarData entities are related by the “rental car” relationship. Each rental is associated with exactly one car (1..1), but the cars can be used in 0 or many rentals (0..*). Rentals and CarData are modeled as separate entities to avoid redundancy since the cars can be used multiple times for different rentals. In addition, the CarData consists of multiple attributes such as GarageId, HourlyRate, Make, and Model, so this approach will optimize space for storing data.

In addition, the “rental feedback” relationship is necessary because customer feedback is collected based on specific rental experiences. Each rental may or may not have a customer satisfaction entry, making the relationship 0..1. There is *at most* one satisfying entry per rental. The 1:1 relationship from CustomerSatisfaction to Rentals exists because each customer review corresponds to only one rental. CustomerSatisfaction has been modeled as a separate entity rather than an attribute within Rentals to avoid null values in Rentals since customers may choose to not leave feedback on their rental.

The CustomerSatisfaction table is a weak entity because it relies on RentalId and CustomerId, the primary keys of the Rentals and Customers table. Therefore, RentalId and CustomerId are also foreign keys. RentalId and CustomerId together form the primary key of CustomerSatisfaction to ensure that each customer can only leave one review per rental.

Finally, the “parked” relation is 0..* from Garages to CarData because a Garage may contain 0 or many cars. The relationship from CarData to Garages is 1..1 since a car may only belong to one garage. The data in Garages is stored as a separate entity rather than as attributes within CarData to avoid redundancy within the CarData table. Since, Garages can have multiple cars, the garage data should not be repeated within the CarData.

BCNF Normalization

We are applying BCNF to normalize our database. Here are the function dependencies for each table within our database.

1. Customers
    a. CustomerId → Name, Email, PhoneNumber, Age
2. Rentals
    a. RentalId → CarId, CustomerId, StartTime, EndTime, GarageId, TotalCost, Miles, Purpose
3. CustomerSatisfaction
    a. RentalId, CustomerId → Rating, Comments
4. Garages
    a. GarageId → Profit, Capacity, Latitude, Longitude
5. CarData
    a. CarId → GaradeId, HourlyRate, Make, Model, Year, Availability
6. GarageLocation
    a. Latitude, Longitude → City

These functional dependencies adhere to the BCNF because the primary key for each relation determines all the other attributes in the same relation. There are no troublesome FDs, therefore there is no redundancy in the database.

We chose to use BCNF to normalize our data because it is “stricter” in that it has a stronger requirement for eliminating redundancy in relational database design. It is generally preferred over 3NF, so we chose BCNF for our normal form.

Relational Schema

Customers(CustomerId:INT [PK], Name:VARCHAR(100), Email:VARCHAR(100), PhoneNumber:VARCHAR(15), Age:INT)

Rentals(RentalId:INT [PK], CarId:INT [FK to CarData.CarId], CustomerId:INT [FK to Customers.CustomerId], StartTime:DATETIME, EndTime:DATETIME, GarageId:INT [FK to Garages.GarageId], TotalCost:DECIMAL(10, 2), Miles:INT, Purpose:VARCHAR(255))

CustomerSatisfaction(RentalId:INT [PK, FK to Rentals.RentalId], CustomerId:INT [PK, FK to Customers.CustomerId], Rating:INT, Comments:TEXT)

Garages(GarageId:INT [PK], Profit:DECIMAL(10,2), Capacity:INT, Latitude:DECIMAL(9,6) [FK to GarageLocation.Latitude], Longitude:DECIMAL(9,6) [FK to GarageLocation.Longitude])

CarData(CarId:INT [PK], GarageId:INT [FK to Garages.GarageId], HourlyRate:DECIMAL(10,2), Make:VARCHAR(100), Model:VARCHAR(100), Year:INT, Availability:BOOLEAN)

GarageLocation(Latitude:DECIMAL(9,6) [PK], Longitude:DECIMAL(9,6) [PK], City:VARCHAR(100))