Project Phase 3: Logical Database Design & Implementation

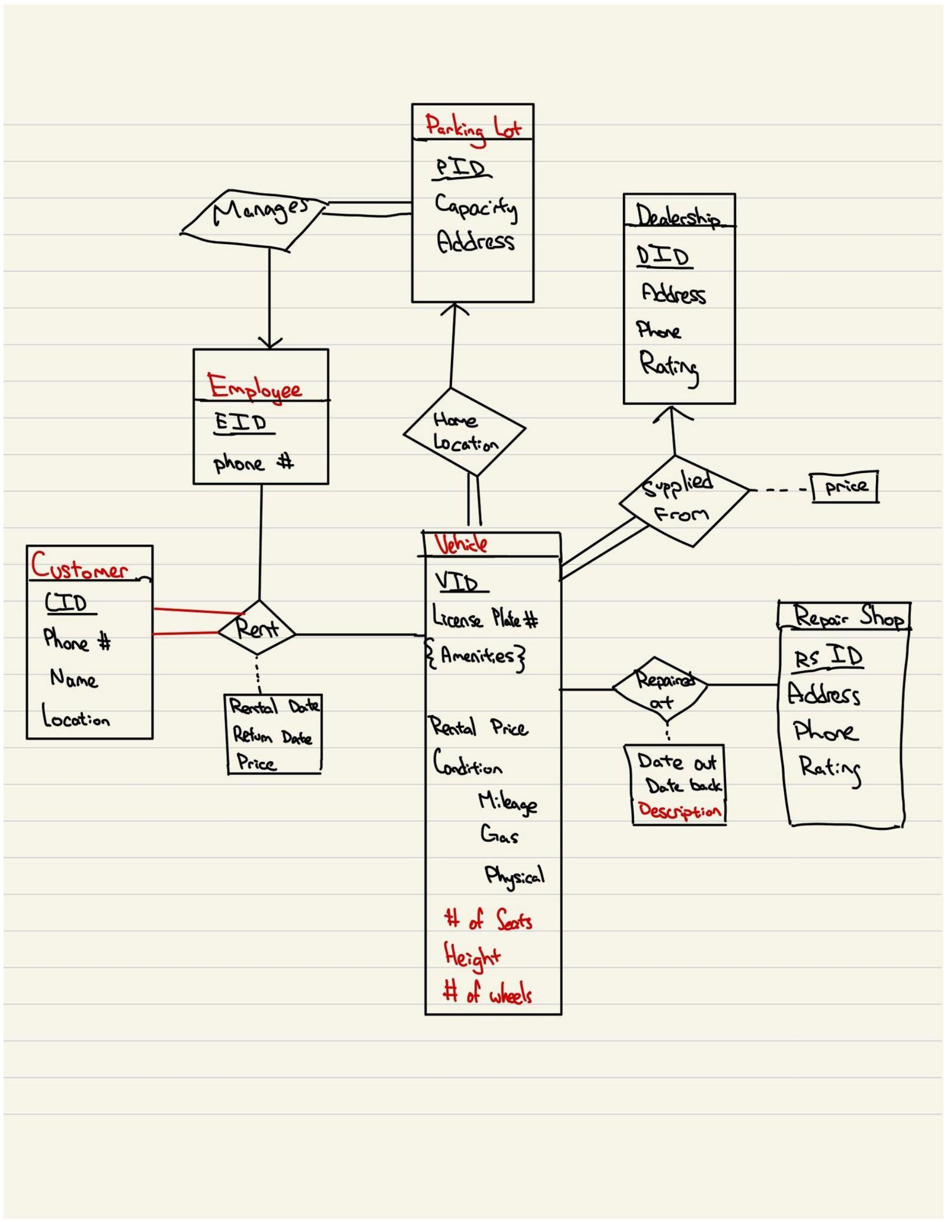
Dilan Bhat, Jessica Yu, Nicholas Lin, Olivia Agatep

For our project, we are modeling a vehicle rental management system. The database represents a single vehicle rental company such that the company can keep track of their vehicles, along with their various attributes, that they can offer to customers who are looking to rent out a vehicle. Our database also includes additional information related to their vehicles such as repair shops, dealerships, and more.

The intended business users are the marketing and sales team of a car rental company that would benefit from keeping and being able to query data that they collect on the rentals they make or want to make. In this example, the data managers mentioned above would specifically be client-facing employees and the software team responsible for the data. The expectations of the database are to organize information about employees, customers, vehicles being rented, and where those vehicles are stored, sold and repaired. Our proposed solution can support new business processes in a car rental company by allowing users to easily query for information on the vehicles available, customer rental history and trends.

Overall, customers and car rental companies can both benefit from this database. Based on a majority of car rental companies, customers will give employees their requirements for what they are looking for in a car rental (number of seats, price, etc). Employees will then use a service that will query our database based on the given requirements. From a business perspective, the car rental company's management can also use the database to monitor the company's health in terms of the number of cars they have been renting out, which employee has been making the most sales, and more.

ER Diagram:



Explanation of diagram:

1. Vehicle: VID is unique identifier, and attributes include license plate number, amenities (can have multiple), price, condition (including mileage, gas, and physical condition), number of seats, height, and number of wheels
2. Customer: CID is unique identifier, attributes include phone number (assume only one), full name, and location/address
3. Employee: EID is unique identifier, attributes include phone number (assume only one)

A rental will have a rental check out and return date, along with the price. Rentals include the vehicle rented out, the employee who completes the transaction, and the customer who is renting the car. A customer can rent out multiple vehicles and an employee can help with multiple transactions.

4. Repair Shop: RSID is unique identifier, attributes include address of shop, phone number (assume only one), and rating

A repair between a vehicle and repair shop includes the start and end date of repair and a description of what work was done. A repair shop can repair multiple vehicles.

5. Dealership: DID is unique identifier, attributes include address, phone number (assume only one), and rating

A vehicle must have been supplied from a dealership and has a price attached.

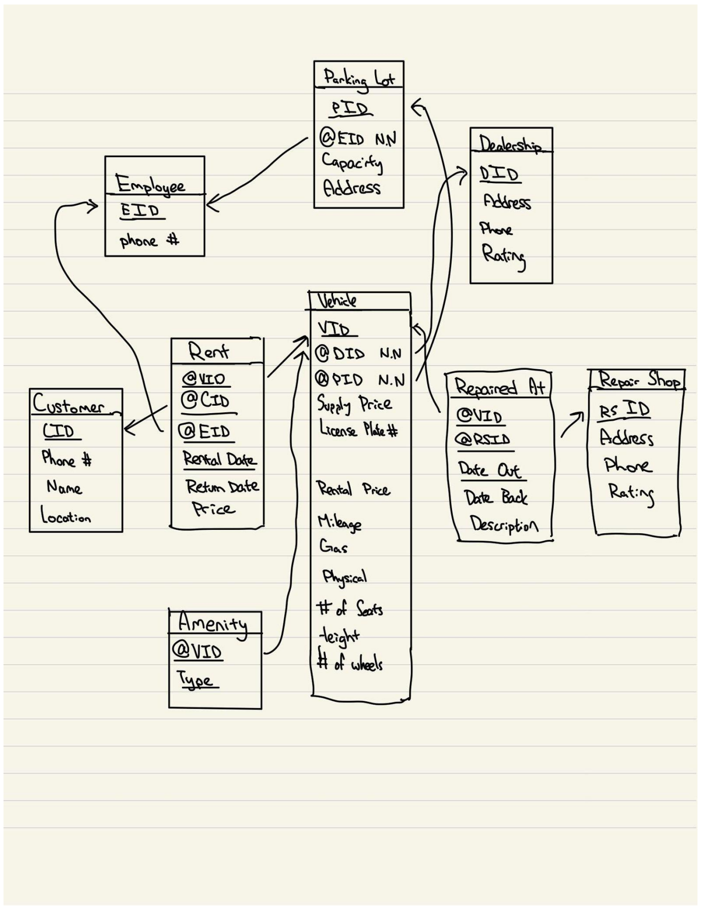
6. Parking Lot: PID is unique identifier, attributes include the vehicle capacity and location

Every vehicle is stored in a parking lot, in which these lots can have multiple vehicles. Parking lots must also be managed by an employee at the company.

Explanation of changes:

1. Solid line between employee and rent since it's connecting an entity to a relationship set.
2. Relationship between customer entity set and rent relationship set is total participation because in order for the customer to exist in the database, they must have rented a car before. A many relationship is also represented here now since different people can rent the same car on different days.
3. Specialization of vehicles (trucks, sedans, SUVs) has been removed from the diagram since vehicles likely have the same height, number of seats and wheels. These attributes have been moved to the vehicle entity set instead.
4. To add more details about a repair (such as changed motor), the 'Repaired At' relationship set now includes a description.
5. Entity set names have been modified to be in singular form.

Relational Schema and Explanation:



1. Per the rules, we create a table representing each entity set in the ER diagram. The table fields match the entity's attributes also.
2. We add three tables:
    a. 'Rent' because of the many-to-many relationship between Customer, Employee, and Vehicle. This table will have the primary keys of Customer, Employee, and Vehicle as primary foreign keys, along with the attributes that belong to a Rent relationship set and Rental Date as a primary key
    b. 'Repaired At' because of the many-to-many-relationship between Vehicle and Repair Shop. Similarly, this table will have the primary keys of those tables as primary foreign keys for the Repaired At table, along with the attributes that belong to the relationship set, with Date Out as a primary key
    c. 'Amenity' because of the multivalued type in the Vehicle table. 'Amenity' has the vehicle's primary key as a primary foreign key along with the type of the amenity as a primary key.
3. For those with a many-to-one relationship (ex: Parking Lot and Employee), the primary key field of the 'one' table gets put into the 'many' table as foreign key field. And for those with total participation, we note the foreign key as N.N. (non-null), such as the EID in the Parking Lot table.

Queries Explanation:

1) Money spent, find the total money gained from each customer ordered from least to most.

*Explanation***:** For each customer tuple joined with rent and vehicle we summed the total price spent on renting cars in our database over that customer and then ordered the rows by ascending price. By grouping by cid we were able to get one total price for each customer.

---

2) Most used vehicle, find the vehicle (or vehicles) rented the most times (assuming a vehicle has been rented at least once)

*Explanation***:** The X query retrieves the total number of times each vehicle was rented. Then the following query utilizes a witness to select only the maximum valued tuple from the resulting X table. By using a witness here we are able to get every vehicle where the number of times rented is the maximum.

---

3) Best employee, list all employees and their total sales in decreasing order

Explanation: The X query lists all the employees and their respective total sales, grouping by employee id to get only each employee and their respective total once. Then by using a union join and a NOT IN we set the total sales value to 0 for all the employees that made no sales.

---

4) Faulty Vehicle, list the number of times the vehicle has been repaired from most to least

Explanation: For each vehicle we join it to its respective RepairedAt entity using vid, and count the total number of entities each vehicle is joined with to count the number of times it has been repaired. We then order by descending count to list the vehicles that were repaired the most to find the vehicles that break down the most.

---

5) Dealership Log, return the customer, employee and rent data from a specific dealership

Explanation: Here we want to return a “log” or a list of all the rental transactions for a given dealership. For this query we use dealership 1. Then we list the customer id, customer name, employee id, vehicle id and rental date for each time any vehicle was rented through dealership 1. This returns a complete list of every time a vehicle was rented from dealership id 1, and all the people involved.

---

6) Parking lot counts, List each parking lot and the count of company vehicles currently there

Explanation: For each parking lot, we selected the parking lot id, address, and the number of cars that belong to that parking lot. We selected the parking lots where there was at least a count of two cars that belong there.

---

7) Specific amenity, list all the vehicles with a given amenity that is currently available

Explanation: Using a left join, we identified the vehicles that had the specific amenity of bluetooth available in the car.

---

8) Customer preferences, list the number of times each customer rented a vehicle with a specific amenity from most to least

Explanation: For this query, we counted the number of times each customer rented a vehicle with a specific amenity to understand a customer's preference and perhaps the demand of certain amenities.

---

9) Repair shop value, from each repair shop, list the address, rating and number of repairs.

Explanation: In this query, the work statistics of each repair shop was listed by getting each repair shop id and its corresponding address, rating, and the number of times a vehicle from the database was repaired there. In a real-world setting this would be beneficial for the company to gauge where to send future broken vehicles by having important repair shop information and their interaction count with the company in one list. The query works by using the X query to first return all the requested information from each repair shop for each repair shop, grouping by rsid to remove repeating data. Then, the main query uses a union and not in to apply a 0 value to each repair shop in the database that has not yet repaired any vehicles. The final list is ordered by numRepairs descending to return the repair shops the company has used the most first.

Insert/Modification Explanation:

1. The first transaction simulates a customer renting out a car, where information was inserted into the Rent relationship. This includes information about the vehicle rented out, customer renting, the employee that facilitated the transaction, and the renting dates.
2. The second transaction simulates sending a car to the repair shop, where information was inserted into the RepairedAt relationship. This includes information about the vehicle being repaired, the repair shop, and the repair dates.

What We Learned:

From this project, our group learned the end-to-end progress of creating a database from a business and customer perspective. Using these perspectives, we had to brainstorm the specific elements and attributes that would be most useful for both parties. For example, keeping track of the sales an employee has taken part in is helpful from a company's perspective since it allows them to monitor the company's progress/status. Similarly, keeping track of a car's amenities is useful for when a customer requires specific elements of a vehicle. Ultimately, the project required creating a database and queries through a real-world lens, as well as using our own experiences with car rental companies, such that we include components that are useful and practical. This overall gives us a better understanding of the application of the topics we've learned in class throughout the semester.

We also learned the importance of working as a group throughout this project. A key takeaway has been discovering the optimal approach to the project: rather than dividing the work, we collaborated together on each component so that we can share ideas, which allowed us to see each member's approach. This especially applied as we were writing our queries, in which we were able to see our different thought processes (such as using a witness query vs left join) and ultimately learn from each other. In addition, adaptability and openness to change were important, specifically after we received feedback from Phase 2 of the project. This allowed us to understand that revision is a standard part of the process to improve our design.