WOW operates as a car rental company with multiple branches housing a diverse fleet of rental vehicles categorized by Make, Model, Year, VIN (Vehicle Identification Number), and License Plate number.

Each vehicle class is associated with a specific daily rental rate and charges for exceeding mileage limits.

WOW serves two customer types: Individuals and Corporations.

Discount coupons are available for both individual and corporate customers, with the constraint that only one type of coupon can be applied per transaction.

WOW has the capability to generate invoices for rental services, and customers have the flexibility to make payments using various methods.

So first we need to create a view of entities in this system to specify their relationships. There are mainly nine entities that exist in the system: location, customer, service, vehicle, vehicle class, invoice, payment, and so on. Customers can be identified by the customer id, and its attributes contain first name, last name, email, phone number, and address that can be divided into street, city, state, and zip code. The customers should also be divided into two subtypes: individual and corporate. For individual ones, we need to store their driver license number, insurance company name and insurance policy number, while for corporate customers, their corporate name and registration number are stored as well as their employee id. Office locations can be identified by the location id, with specific addresses and phone number stored. For vehicle class, it can be identified by the vehicle id and includes vehicle type, daily rate of service, over mileage rate and odometer limit. In each class, there are multiple vehicles available. They can be identified by their VIN (Vehicle Identification Number), and we store the make, model, year, and license plate number. In the service entity, it can be specified by the service id, and they include the number of rental days, start and end odometer. In the discount entity identified by discount id, there are two subtypes of corporate and individual ones. Both include discount percentage, but the difference is that corporate discounts are differentiated through companies, so the corporate name needs to be stored. For individual ones, they have validity dates, so we need to have the storage of valid start date and valid end date. For the invoice identified by the invoice id, the invoice amount can be generated from the service with the invoice date included. Then the payment entity is identified by the payment id, and it has attributes of payment date, card number and CVV. Since customers can pay an invoice using multiple methods, so we also need the card type entity consisting of card type id and type name.

In these relationships

- One location can have many vehicles and services.
- Vehicle Class has a one-to-many relationship with Vehicle since one class can have many vehicles.
- Customer has a one-to-many relationship with Service because one customer can have many rental services.

- And Service has a one-to-many relationship with Invoice.
- Invoice has a one-to-many relationship with Payment and each card type can be used in different payments.
- Service also has a one-to-one relationship with Discount. That means one rental service can only have one discount coupon used and it’s optional.

Here’s the relational model generated from the logical model, and we modified some foreign keys to be correctly name. And here’s the list of tables and we have populated some sample data into the tables, and it looks well in the database. Next, Yunling Bai will talk about the backend section.