

Milestone 1: Requirements Analysis & Conceptual Design

Car Rental Company

A **car rental company** (CRC) rents cars through their subsidiaries. The CRC has one or multiple **subsidiaries**. Each subsidiary has an address, a number of employees and a turnover (as well as other performance-related figures) and employs multiple employees which each have a name, an unique ID, a phone number and the total sum of all sales they accrued, as a measure for their performance. Each **employee** can only work at one subsidiary and has one or more boss/es.

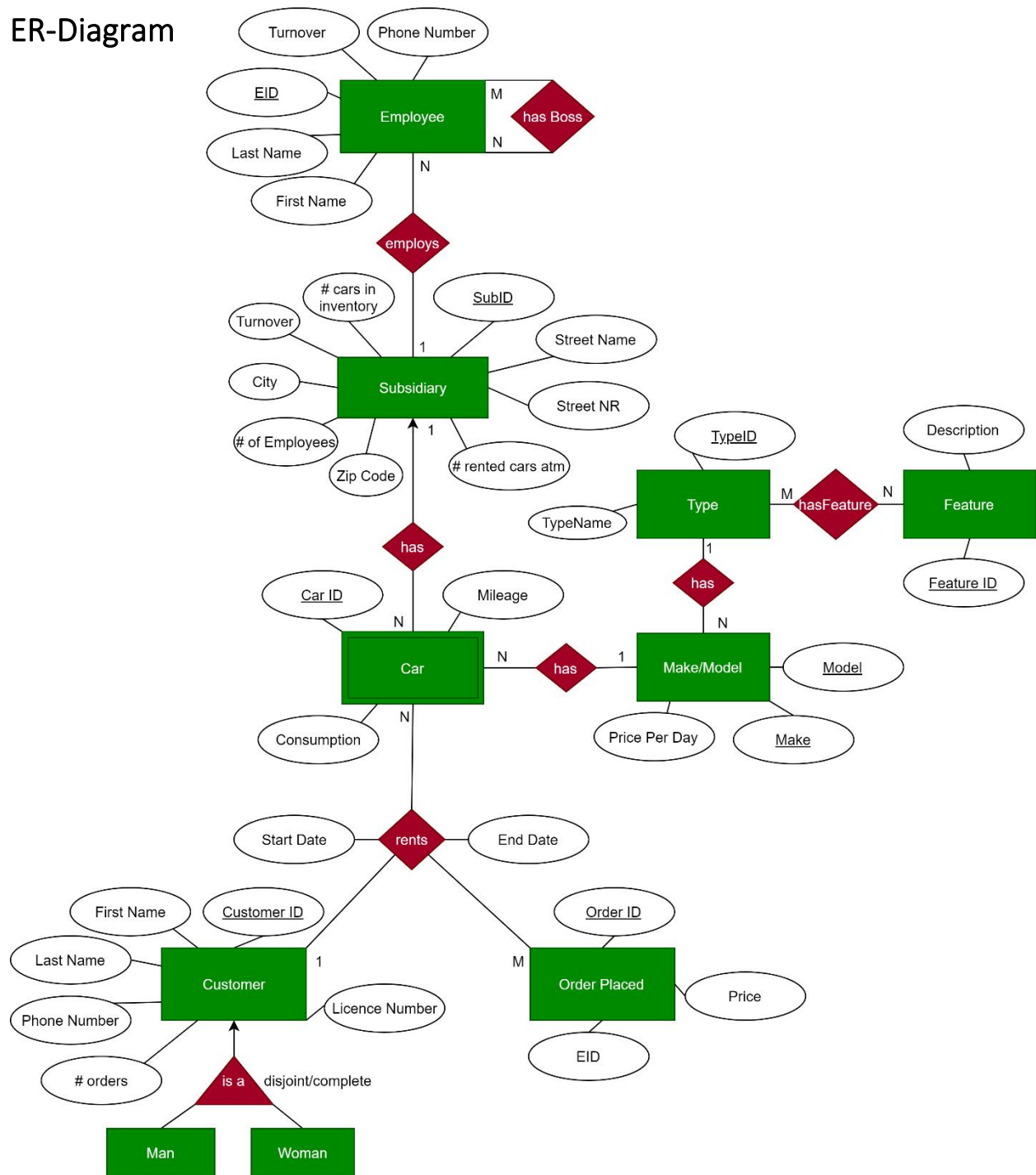
Each subsidiary has multiple cars in their inventory. Each **car** has an ID which is only unique in combination with the ID of the subsidiary, mileage, consumption and belongs to a **make** and **model** which together belong to a certain **type**. The particular combination of make and model have a price per day which is used to calculate the overall price of an order. Each type, such as SUVs or limousines, have their own set of features, such that every SUV has the same features, etc. Nonetheless, multiple types can share a common feature, e.g. every type has at least a navigation system and climate control.

Each **order** is placed by one **customer** and can contain up to three cars. This customer can order the same car again in a new order if they wish to. Every customer has an unique ID, a name, a phone number, a driving licence number and the number of orders they already placed as well as their gender which can be male or female. Both male and female customers share the same attributes, however.

The **order** has an ID, a start- and end date, the ID of the employee and a price, which is calculated from the duration of the order as well as the cars that are part of the order.

This implementation is meant as a **database access point** for the employees of the subsidiaries. With it they can update, insert, delete or search for existing data.

ER-Diagram



Milestone 2: Logical Design

----- Entities -----

Subsidiary(SubID, Street_Name, Street_Nr, ZIP_Code, City, # Employess, # Cars_in_Inventory, # Rented_Rars_atm, Turnover)
PK: SubID

Employee(EID, First_Name, Last_Name, Phone_Number, Turnover, *SubID*)
PK: EID
FK: Employee. SubID \diamond Subsidiary. SubID

Car(CarID, *SubID*, Make, Model, Mileage, Consumption)
CK: (CarID, SubID)
FK: Car. SubID \diamond Subsidiary. SubID, Car.Make \diamond Make_Model.Make, Car.Model \diamond Make_Model.Model

Make_Model(Make, Model, Price/Day, *TypeID*)
CK: (Make, Model)
FK: Make_Model.TypeID \diamond Type.TypeID

Type(TypeID, TypeName)
PK: TypeID

Feature(FeatureID, Description)
PK: FeatureID

OrderPlaced(OrderID, *EID*, Price)
PK: OrderID
FK: OrderPlaced.EID \diamond Employee.EID

Customer(CustomerID, Licence_Nr, FirstName, LastName, Phone_Number, # Orders, Gender)
PK: CustomerID

----- Relations -----

hasBoss (*EID1*, *EID2*)
CK: (EID1, EID2)
FK: hasBoss.EID1 \diamond Employee.EID, hasBoss.EID2 \diamond Employee.EID

hasfeature(*TypeID*, *FeatureID*)
CK: (TypeID, FeatureID)
FK: hasfeature.TypeID \diamond Type.TypeID, hasfeature.FeatureID \diamond Feature.FeatureID

rents(OrderID, CarID, SubID, *CustomerID*, Start_Date, End_Date)

CK: (OrderID, CarID, SubID)

FK: rents.CarID \diamond Car.CarID, rents.SubID \diamond Subsidiary.SubID, rents.OrderID \diamond OrderPlaced.OrderID,
rents.CustomerID \diamond Customer.CustomerID

Milestone 4: Implementation

Documentation of the normalization process:

- **FD: SubID → Zip_Code → City (≠ 3NF)**

Therefore **Subsidiary** was split into **Subsidiary** and **Location** via a 1-N relation, where **Subsidiary** inherited the **Zip_Code** as a foreign key:

Location(Zip_Code, City)

PK: Zip_Code

Subsidiary(SubID, Street_Name, Street_Nr, *Zip_Code*, # Employess, # total_cars, # rented_cars_atm, turnover)

PK: SubID

FK: Subsidiary.Zip_Code ◇ Location.Zip_Code

- **FD: OrderID → Start_date, End_date. Not 2NF-conform!**

Therefore, **rents** was split into two separate tables for orderIDs and start & end dates

- **FD: OrderID → CustomerID. Not 2NF-conform!**

Therefore, separate tables for orderIDs and customerIDs from **rents**:

Both tables could be combined into the table **OrderDetails** because no NF-conditions were violated and to reduce unnecessary redundancy.

rents(OrderID, CarID, SubID)

CK: (OrderID, CarID, SubID)

FK: rents.CarID ◇ Car.CarID, rents.SubID ◇ Subsidiary.SubID, rents.OrderID ◇ OrderPlaced.OrderID

OrderDeatils(OrderID, *CustomerID*, Start_date, End_date)

PK: OrderID

FK: OrderDetails.CustomerID ◇ Customer.CustomerID, OrderDetails.OrderID ◇ OrderPlaced.OrderID