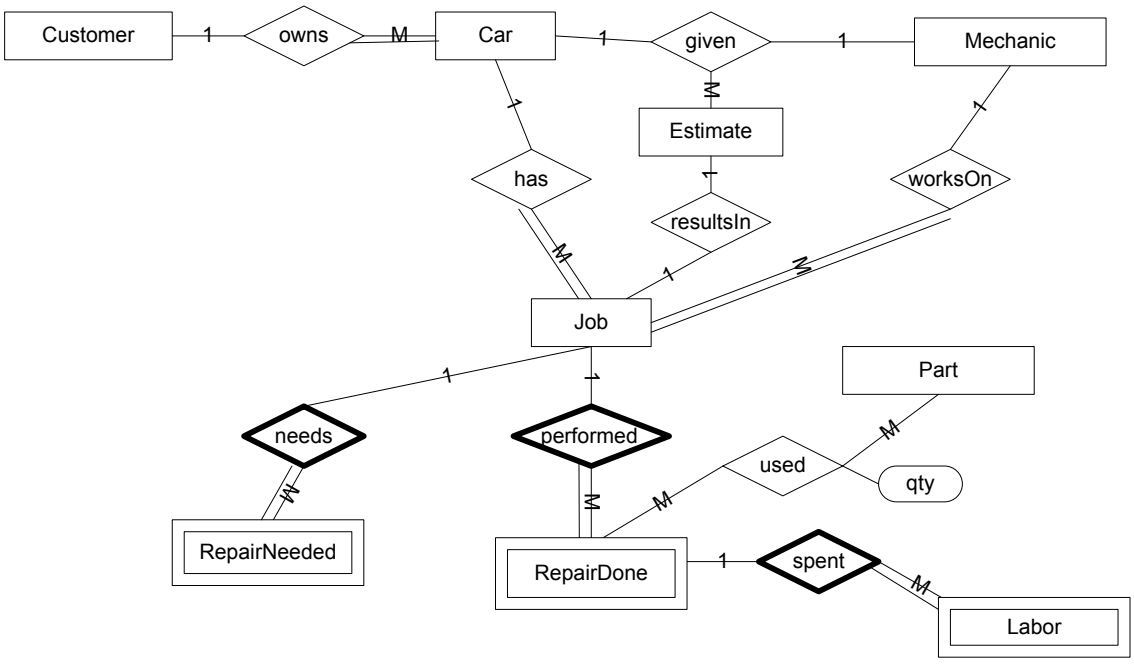
CS431: Normalization Study v.20170426

This study should not be turned in as homework. You can use this to practice doing normalization on a designed database. The following sections begin with (1) a description, (2) followed by a possible E-R diagram, (3) a defined schema based on E-R diagram, and (4) a sample normalization analysis. Try to follow the normalization of each relation and understand the logic behind it.

3.7 An automobile body repair shop needs to keep information about its operations. Customers initially bring their cars to the shop for an estimate of repairs. A mechanic looks at the car and estimates the cost and time required for the entire job. If the customer accepts the estimate, a job number is assigned and the customer's name and contact information; the car's license plate number, make, model, and year; and a list of the repairs needed are recorded. The customer then makes an appointment to bring in the car on a specified date. When the car is brought in for repairs, the work begins. The shop keeps track of the charges for parts and labor as they accumulate. Only one mechanic works on the car for the entire job. A job might include several repairs (e.g., replacing the left fender, painting the passenger door). The time actually spent for each repair is recorded and used to calculate the cost of labor, using a fixed hourly rate. Draw a complete E-R diagram for this example.

Sample E-R Diagram



Sample Database Schema:

For the E-R diagram shown above

Customer(driverLicState,driverLicNumber, firstName, lastName, street, city, state, zip, areaCode, phoneNumber, referredBy)

Car(licPlateState, licPlateNumber, VIN, make, model, year, color, mileage, cylinders, driverLicState, driverLicNo)

Mechanic(mechLicNo, firstName, lastName, street, city, state, zip, areacode, phoneNumber, specialty)

Estimate(estNo, dateGiven, timeNeeded, estCost, licPlateState, licPlateNumber, mechLicNo)

Job(jobNo, description, dateStarted, totalHoursSpent, totalCostOfParts, dateCompleted, totalCost, licPlateState, licPlateNo, mechLicNo)

RepairNeeded(jobNo, type, description, timeNeeded)

RepairDone(jobNo, type, description, dateStarted, timeSpent, dateCompleted)

Part(partNo, partName, manufacturer, supplier, unitCost)

PartUsed(jobNo, repairType, partNo, qty)

Labor(jobNo, repairType, numberHours)

Sample Normalization:

For Customer, we have
driverLicState, driverLicNumber → all attributes
If phoneNumber is unique, we may have
phoneNumber → all attributes
If that is the case, the determinant is a superkey, so we can leave it in the relation.

We also have the transitive dependency
zip → city, state

We therefore use projection to give us the following relations, both in BCNF
Customer(driverLicState,driverLicNumber, firstName, lastName, street, *zip*, areaCode, phoneNumber, referredBy)

Zips (zip, city, state)

In Car, in addition to the FD on the key, we have
VIN → all attributes
Since this is a superkey, we can leave it in the relation, so the relation is already BCNF

Car(licPlateState, licPlateNumber, VIN, make, model, year, color, mileage, cylinders, *driverLicState, driverLicNo*)

In Mechanic, we remove the city and state, leaving the zip. We will use the Zips table from Customer. We assume phoneNumber is a superkey, so we leave it in the relation.
Mechanic(mechLicNo, firstName, lastName, street, *zip*, areacode, phoneNumber, specialty)

Estimate is already BCNF, since the primary key is the only determinant.
Estimate(estNo, dateGiven, timeNeeded, estCost, *licPlateState, licPlateNumber, mechLicNo)*

In JobNo, in addition to the primary key FD, since we assume the total cost of a job is the sum of the cost of parts and the cost of labor, which is a fixed rate per hour, we have

{totalHoursSpent, totalCostOfParts} → totalCost
Job(jobNo, description, dateStarted, totalHoursSpent, totalCostOfParts, dateCompleted, totalCost, *licPlateState, licPlateNo, mechLicNo*)

We change the relation to
Job(jobNo, description, dateStarted, totalHoursSpent, totalCostOfParts, dateCompleted, *licPlateState, licPlateNo, mechLicNo*)
Since the total cost is easily computed, we do not store a table for it.

For Repair Needed, we have the usual FD for the primary key. Depending on our assumptions, we may also have
Type → description, time Needed

If this FD were true, we would take the dependent attributes out and create the two tables
RepairNeeded1(*jobNo*, type)
RepairMenu(type, description, timeNeeded)

However, we will assume the type of repair is generic enough so that different descriptions and time needed might exist for the same repair type, so we will keep the original relation.
RepairNeeded(*jobNo*, type, description, timeNeeded)

For the RepairDone table, we assume that type is generic, so the only FD is the one involving the primary key, leaving us

RepairDone(*jobNo*, type, description, dateStarted, timeSpent, dateCompleted)

For Part, we assume
partNo → all attributes
partName → all attributes
Since both determinants are superkeys, we leave the relation intact.

Part(partNo, partName, manufacturer, supplier, unitCost)

In both PartUsed and Labor, the only determinants are the primary keys, so they are already BCNF.
PartUsed(*jobNo, repairType*, *partNo*, qty)
Labor(*jobNo*, *repairType,* numberHours)

The final normalized schema consists of the relations in boldface.