dskenlin
machaffe
dsmicken
cjhetzle

Entity Types

- PartSource
- ServiceCenter
- Employee
- Paycheck (weak entity)
- PartOrder
- Notification (weak entity)
- PartDatabase
- Customer
- Vehicle
- ServiceAppointment (weak entity)
- Service
- Fault
- ByMileage
- BillableUnit
- Warranty

Relationship Types

- Isa relationship between PartSource and subclass ServiceCenter
- Isa relationship between Service and subclass ByMileage/BillableUnit
- Isa relationship between BillableUnit and subclass Warranty
- “Delivers” is a binary relationship between PartSource and PartDatabase
- “ShipFrom” is a binary relationship between PartSource and PartOrder
- “HasStock” is a binary relationship between ServiceCenter and PartDatabase
- “ShipTo” is a binary relationship between PartOrder and ServiceCenter
- “Contains” is a binary relationship between PartOrder and Part
- “WorksIn“ is a binary relationship between Service Center and Employee
- “Creates” is a binary relationship between Notification and PartOrder
- “PaidTo” is a binary relationship between Paycheck and Employee
- “Schedules” is a ternary relationship between Employee, ServiceAppointment, and BillableUnit
- “Consumes” is a binary relationship between PartDatabase and BillableUnit
- “Prefers” is a binary relationship between ServiceAppointment and Employee
- “For” is a binary relationship between ServiceAppointment and Vehicle
- “Owns” is a binary relationship between Vehicle and Customer
- “Reports” is a binary relationship between ServiceAppointment and Fault.
- “Includes” is a binary relationship between Service and ByMileage
- “Recommends” is a binary relationship between BillableUnit and Fault

dskenlin
machaffe
dsmicken
cjhetzle

Relational Model

This is the relational model for the Acme Car Service database. We're targeting 3rd Normal Form since it gives a good balance of eliminating redundancy without being too restrictive.

PartSource

Lists a set of potential source for parts. This will include the unique ids of service centers and distributors. It has no functional dependencies

ServiceCenter

Service Center lists all Acme Car Service and Repair Center centers which are uniquely identified by their service center id, cid. cid references PartSource. ServiceCenter has the following functional dependencies:

- cid → name, street, city, state, zip, phone

Employee

Employee lists Acme Car Service and Repair Center employees and information about them. They are uniquely identified by their employee ID, eid. The service center that employs them, cid, references the ServiceCenter table. Employee has the following functional dependencies:

- eid → cid, name, street, city, state, zip, emailAddress, phone, role, startDate, pay, password

Customer

Represents a customer which includes their name, address details, email address, phone number, and password. customerID is the primary key, but emailAddress is also a candidate key since it's unique. Customer has the following function dependencies:

- customerID → cid, name, street, city, state, zip, emailAddress, phone, password
- emailAddress → customerID, cid, name, street, city, state, zip, phone, password

Vehicle

Represents a customer's vehicle including the license plate number, make, model, and year of the vehicle, date purchased and customerID. References Customer by customerID. We decided not to store the type, mileage, and date of the most recent service here because you can just query ServiceAppointment. Vehicle has the following functional dependencies:

- license → make, model, year, datePurchased, customerID

dskenlin
machaffe
dsmicken
cjhetzle

Service

The Service table encompasses all services. This is a superclass for both specific basic services and the larger scheduled maintenance services. It will contain a serviceID field, which is the primary key, and a name field (which includes the make and model). It has the following functional dependencies:

- serviceID → name

ByMileage

The ByMileage table contains scheduled maintenance services by mileage, make, and model. The serviceID will serve as the primary key and must reference the Service table. The combination of mileage, make, and model must also be unique and therefore make up a candidate key, and none of them may be null. ByMileage has the following functional dependencies:

- serviceId → mileage, make, model
- (mileage, make, model) → serviceId

BillableUnit

The BillableUnit represents basic services that will be presented as invoice items to the customer. The serviceID will serve as the primary key and must reference the Service table. The chargeRate and timeReq can not be null. It has the following functional dependencies:

- serviceID → chargeRate, timeReq

Includes

The Includes table specifies what services are included in a ByMileage service. It will have two fields, includer and included, which together serve as the primary key. Includer references the ByMileage table. Included references the Service table. It has no functional dependencies.

PartDatabase

The PartDatabase table will hold the distributor (referencing PartSource), name, the cost per unit, and the delivery time for each part. Every part is identified uniquely by the partID, which can be used in other tables to find information like part orders that are pending for that item. PartDatabase has the following functional dependencies:

- partID → distID, name, unitPrice, distDelay

dskenlin
machaffe
dsmicken
cjhetzle

Warranty

The Warranty table specifies when a basic maintenance service has a warranty and what repair services that warranty covers. The service makes up the primary key and references BillableUnit. ISA is used because warrantied services are a subset of BillableUnits with extra fields. There will also be a field for the duration of the warranty. It has the following functional dependencies:

- service → duration

Consumes

The Consumes table captures what parts are consumed by each service. The serviceID field references BillableUnit and is the primary key. The partID field references PartDatabase. There is also a quantity field that cannot be null. It has the following functional dependencies:

- serviceId → partID, qty

Fault

The Fault table captures the list of reportable faults. It has the name of the fault, which is the primary key, and how much it costs to diagnose. It has the following functional dependencies:

- name → diagnosticFee

Recommends

The Recommends table captures the relationship between faults and associated repair services. The serviceID field references the BillableUnit table, the fault field references the Fault table, and together they comprise the primary key. It has the following functional dependencies:

- (Fault, serviceID) → make, model

ServiceAppointment

The ServiceAppointment table captures the information about an appointment. License references the Vehicle table, fault references the Fault table (and can be null if this is a maintenance appointment), and preferredMechanic references the Employee table. License and apptDate are the primary key, and the mileage field must not be null. It has the following functional dependencies:

- (license, apptDate) → fault, preferredMechanic, mileage

Schedules

Schedules captures the timeslots. The mechanic field references the Employee table, the service field references the BillableUnit table and the license and dateScheduled fields reference the ServiceAppointment table. There is also an attribute for timeSlot, which is the time of day in 15-minute increments. A mechanic can only perform one service on one car at one point in time, so mechanic, dateScheduled, and timeSlot are part of a candidate key. Schedules has the following functional dependencies:

- (license, dateScheduled, timeSlot, mechanic) → service
- (mechanic, dateScheduled, timeSlot) → licenseNum, service.

PartInventory

The PartInventory table will hold the partID (referencing PartDatabase), the service center id, cid, (referencing ServiceCenter), the quantity of parts for that center, the minimum required quantity to be in inventory, and the minimum quantity that can be ordered. The key for this table will be the combination of partID and cid. This table will be used primarily for service centers to coordinate materials between them. PartInventory has the following functional dependencies:

- (partID, cid) → quantity, minQnty, minOrderQnty

PartOrders

PartOrders has a unique orderID as the primary key. This table will hold information for quantity, where the parts are coming from (referencing PartSource) and where the parts are going (referencing ServiceCenter) as well as details about the dates and the status of the order. PartOrders has the following functional dependencies:

- orderID → dateOrdered, dateExpected, dateArrived, partID, quantity, sourceFrom, centerTo, status

Paycheck

The paycheck table lists a history of the paychecks given out. They are uniquely identified by the employee id, eid, which is a reference to the Employee table, and the date that the paycheck was issued which combine to form the primary key. Paycheck also keeps track of the pay period information and the amount earned. Paycheck has the following functional dependencies:

- (eid, datePaid) → periodStart, periodEnd, rate, payableTime, earning, ytdEarning

Notifications

The Notifications table contains alerts that are generated when PartOrders are delayed. The primary key is notificationID. It also stores the date it was generated and the orderID (referencing PartOrders) that was delayed. It has the following functional dependencies:

- notificationID → dateGenerated, orderID

dskenlin
machaffe
dsmicken
cjhetzle

Application Constraints

These are the constraints enforced by the application code (Java) instead of the database.

Employees

- We don't store compensation frequency because it functionally depends on the employee's role. Instead, we have a function that returns "monthly" for receptionists/managers and "hourly" for employees.
- The application prevents adding more than one receptionist to a service center. This way, we can immediately warn a user in the UI as soon as they type in the role.

Parts and Orders

- The application calculates the number of parts due to arrive in time for an appointment (because it involves calculating business days).
- The application decides whether to order a part from another service center or from a distributor. The queries to do this are complex (30+ lines), so we used application code to make it simpler.
- The application checks if a part is below the minimum quantity and places an order. We decided it was more efficient to run this check once per day (during the receptionist's daily tasks) rather than once per UPDATE on the PartInventory table.

Payroll

- The application generates payrolls when an employee views the payroll page, then saves them so they can be accessed faster later.
- The application calculates year-to-date earnings from the list of payrolls instead of storing this value.

Vehicles

- The application prevents customers from registering unsupported vehicles. This way, we can immediately warn a user in the UI.

Other

- Session data (such as which user is logged in) is handled in the application because it does not need to persist between restarts of the application.

Constraints that couldn't be implemented in the database at all

- In MySQL, `CHECK` constraints are ignored. As a result, values for `Employee.role`, `Schedules.timeSlot`, `PartOrders.status`, and other values are checked in the application instead of the database.
- MySQL's date arithmetic doesn't have a concept of "business days", so any time we need to measure business days, it's done in the application.

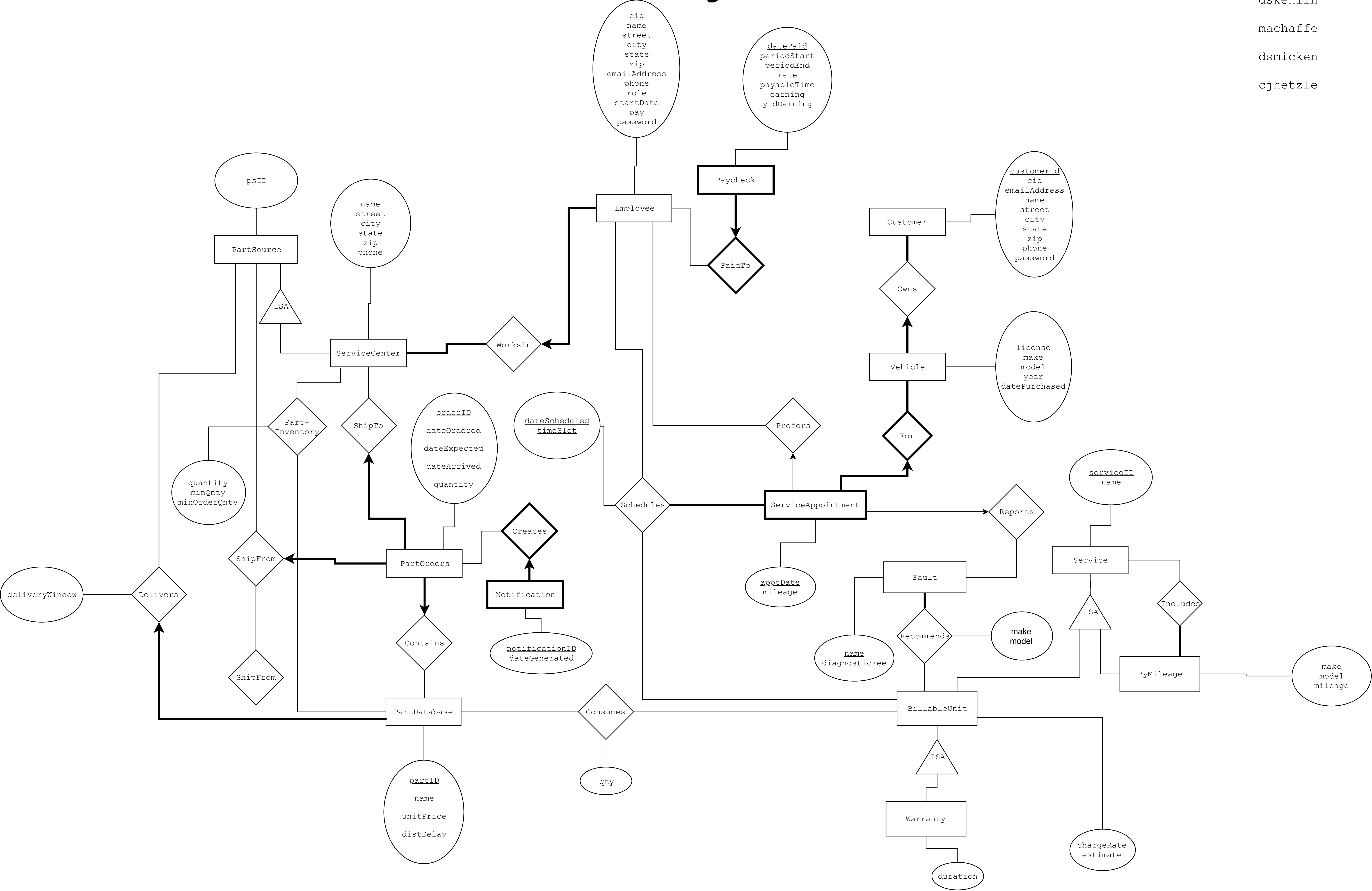
ER Diagram

dskenlin

machaffe

dsmicken

cjhetzle


eid
name
street
city
state
zip
emailAddress
phone
role
startDate
pay
password
datePaid
periodStart
periodEnd
rate
payableTime
earning
ytdEarning
psID
name
street
city
state
zip
phone
Paycheck
Employee
PaidTo
Customer
customerId
cid
emailAddress
name
street
city
state
zip
phone
password
PartSource
ISA
Owns
WorksIn
ServiceCenter
Vehicle
license
make
model
year
datePurchased
Part-Inventory
ShipTo
orderID
dateOrdered
dateExpected
dateArrived
quantity
dateScheduled
timeSlot
Prefers
For
quantity
minQnty
minOrderQnty
Schedules
ServiceAppointment
Reports
serviceID
name
Creates
ShipFrom
PartOrders
Service
apptDate
mileage
Fault
Notification
Includes
ISA
deliveryWindow
Delivers
Contains
Recommends
make
model
name
diagnosticFee
notificationID
dateGenerated
ByMileage
make
model
mileage
ShipFrom
PartDatabase
Consumes
BillableUnit
ISA
qty
partID
name
unitPrice
distDelay
Warranty
chargeRate
estimate
duration