Discussion

**advanced SQL features like Procedures, Triggers etc**

* **Fsgd**
* **Fds**
* **sdgsdg**

**Corrections**

* Separated Managers relation from Employees
* When we created the SQL tables, ServiceCenters and Employees, They depended on each other so, we created the Managers relation from Employees that was associated with three role types, manager, receptionist and mechanic.

Before

ServiceCenters(centerId, minWage, maxWage, address, phone, satOpen, managerId(fk Ref Employee))

Employees(employeeId, centerId(fk), roleType, username, password, firstName, lastName, email, phone, startDate, address, wage, salary)

After

ServiceCenters(centerId, minWage, maxWage, address, phone, satOpen, managerId(fk Ref Employee))

Employees(employeeId, centerId, roleType, username, password, firstName, lastName, email, phone, startDate, address, salary)

* Services
* We had the general approach of ISA hierarchies to maintenance and repair services, but changed to alternative (Object-oriented approach).
* Added the hours and attribute for labor to the relations.
* Added the serviceType attribute for distinguish service types into the relation.
* Added the Price tier table for efficiency to map to prices by the car models

Before

Services(serviceId, name)

RepairServices(serviceId, category)

MaintenanceServices(serviceId)

After

RepairServices(serviceId, category, name, hours)

MaintenanceServices(serviceId, scheduleType, hours, name, serviceType)

* Adopted Timeslot

**Application Design**

ServiceCenters(centerId, minWage, maxWage, address, phone, satOpen, managerId)

* Functional Dependencies
* centerId-> minWage, maxWage, address, phone, satOpen
* centerId -> managerId
* This is in BCNF.
* When we created the SQL tables, ServiceCenters and Employees, They depended on each other so, we created the Managers relation from Employees that is associated with three role types, manager, receptionist and mechanic.

Employees(employeeId, centerId, roleType, username, password, firstName, lastName, email, phone, startDate, address, wage, salary)

* Functional Dependencies
* employeeId -> roleType, username, password, firstName, lastName, email, phone, startDate, endDate, address
* centerId -> employeeId
* employeeId , roleType ->, wage, salary
* This relation is not in 3NF form so we created two relations that are both in BCNF form.
* Receptionists(employeeId, salary)
* Mechanics(employeeId, wage)

Receptionists(employeeId, salary)

* Functional Dependencies
* employeeId -> salary
* This relation is in BCNF

Mechanics (employeeId, wage)

* Functional Dependencies
* employeeId -> wage
* This relations is in BCNF form.

Vacations(vacationId, employeeId, fromDate , toDate)

* Functional Dependencies
* vacationId -> fromDate, toDate
* vacationId -> vacationId
* This relation is in BCNF form.

RepairServices(serviceId, category, name, hours)

* Functional Dependencies
* serviceId -> name, hours
* category -> serviceId, name
* This relation is in 3NF form.

MaintenanceServices(serviceId, scheduleType, hours, name, serviceType)

* Functional Dependencies
* serviceId -> scheduleType, hours
* serviceType -> name
* This relation is not in 3NF so we created the relation that is in BCNF.
* MaintHasServices(serviceId, name, serviceType)

MaintServicePriced(serviceId, centerId, model, priceTier, price)

* Functional dependencies
* serviceId, centerId, model -> priceTier
* priceTier -> price
* This is not 3NF or BCNF so, we create the relation that is in BCNF
* Prices(centerId, model, priceTier, dollar)

RepairServicePriced(centerId, serviceId, model, priceTier, price)

* Functional Dependencies
* centerId, serviceId, model -> priceTier
* priceTier -> price
* This is not 3NF or BCNF, so we create the relation that is in BCNF
* Prices(centerId, model, priceTier, dollar)

Customers(customerId, centerId, firstName, lastName ,address, status, active)

* Functional dependencies

customerId -> firstName, lastName, address, status, actice

customerId -> centerId

* This relation is in BCNF form.

CustomerVehicles(vin, customerId, model, mileage, year, lastMClass)

* Functional dependencies

vin -> mileage,year, lastMclass

vin -> model

customerId -> vin

* This is in BCNF form.

ServiceEvents(eventId, vin, mechanicId, scheduledService, week, day, startDate, startTimeSlot, endTimeSlot, totalPrice, totalPaid, completed, invoiceStatus)

* Functional dependencies
* EventId -> scheduledServices, startDate, week, day, startTimeSlot, endTimeSlot, totalPrice, totalPaid, completed, invoiceStatus
* Eventid -> vin
* eventid -> mechanicId
* This is in BCNF form.

Invoices( invoiceId,eventid, createDate, totalPrice, totalPaid,status)

* Functional dependencies
* invoiceId -> createDate, totalPrice, totalPaid, status
* invoiceId -> eventid
* This relation is in BCNF form.

**Constraints**

Employees(employeeId, centerId, roleType, username, password, firstName, lastName, email, phone, startDate, address)

* Keys
  + employeeId: Unique identifier
* NOT NULL
* centerId
* roleType
* Referential Integrity
* centerId

Receptionists(employeeId, salary)

* Key
* employeeId
* Referential Integrity
* employeeId Employees (employeeId) on Delete cascade

Mechanics (employeeId, wage)

* Key
* employeeId
* Referential Integrity
* employeeId Employees (employeeId) on Delete cascade

Vacations(vacationId, employeeId, fromDate , toDate)

* Key
* vacationId
* NOT NULL
* fromDate
* Referential Integrity
* employeeId Mechanics (employeeId) on Delete cascade

ServiceCenters(centerId, minWage, maxWage, address, phone, satOpen, managerId)

* Keys
  + centerId: Unique identifier
* NOT NULL
* minWage , maxWage
* Referential Integrity
* centerId

RepairServices(serviceId, category, name, hours)

* Keys
  + serviceId: Unique identifier
* NOT NULL
* name , hours, category

MaintenanceServices(serviceId, scheduleType, hours, name, serviceType)

* Keys
* serviceId : Unique identifier
* NOT NULL
* scheduleType, hours, name

MaintHasServices(serviceId, name, serviceType)

* Keys
* serviceId, name
* NOT NULL
* serviceType
* Referential Integrity
* serviceId references MaintenanceServices(serviceId) on Delete cascade

MaintServicePriced(serviceId, centerId, model, priceTier, price)

* Keys
* serviceId, centerId, model
* NOT NULL
* priceTier, price
* Referential Integrity
* serviceId references MaintenanceServices(serviceId) on Delete cascade
* model references CarModels(model)
* centerId references ServiceCenters(centerId) on Delete cascade

RepairServicePriced(centerId, serviceId, model, priceTier, price)

* Keys
* serviceId, centerId, model
* NOT NULL
* priceTier, price
* Referential Integrity
* serviceId references MaintenanceServices(serviceId) on Delete cascade
* model references CarModels(model)
* centerId references ServiceCenters(centerId) on Delete cascade

Customers(customerId, centerId, firstName, lastName ,address, status, active)

* Keys
* customerId : Unique identifier
* Referential Integrity
* centerId references ServiceCenters(centerId) on Delete cascade

CustomerVehicles(vin, customerId, model, mileage, year, lastMClass)

* Keys
* Vin: Unique identifier
* Referential Integrity
* model references CarModels(model)
* customerId references Customers

ServiceEvents(eventId, vin, mechanicId, scheduledService, week, day, startDate, startTimeSlot, endTimeSlot, totalPrice, totalPaid, completed, invoiceStatus)

* Keys
* eventId: Unique identifier
* NOT NULL
* scheduledService
* Referential Integrity
* vin references CustomerVehicles (vin)
* mechanicId references Mechanics(employeeId)

Invoices( invoiceId,eventid, createDate, totalPrice, totalPaid,status)

* Keys
* invoiceId: Unique identifier
* NOT NULL
* createDate
* Referential Integrity
* eventId references ServiceEvents (eventId)