CSC-540 Database Management Concepts and System

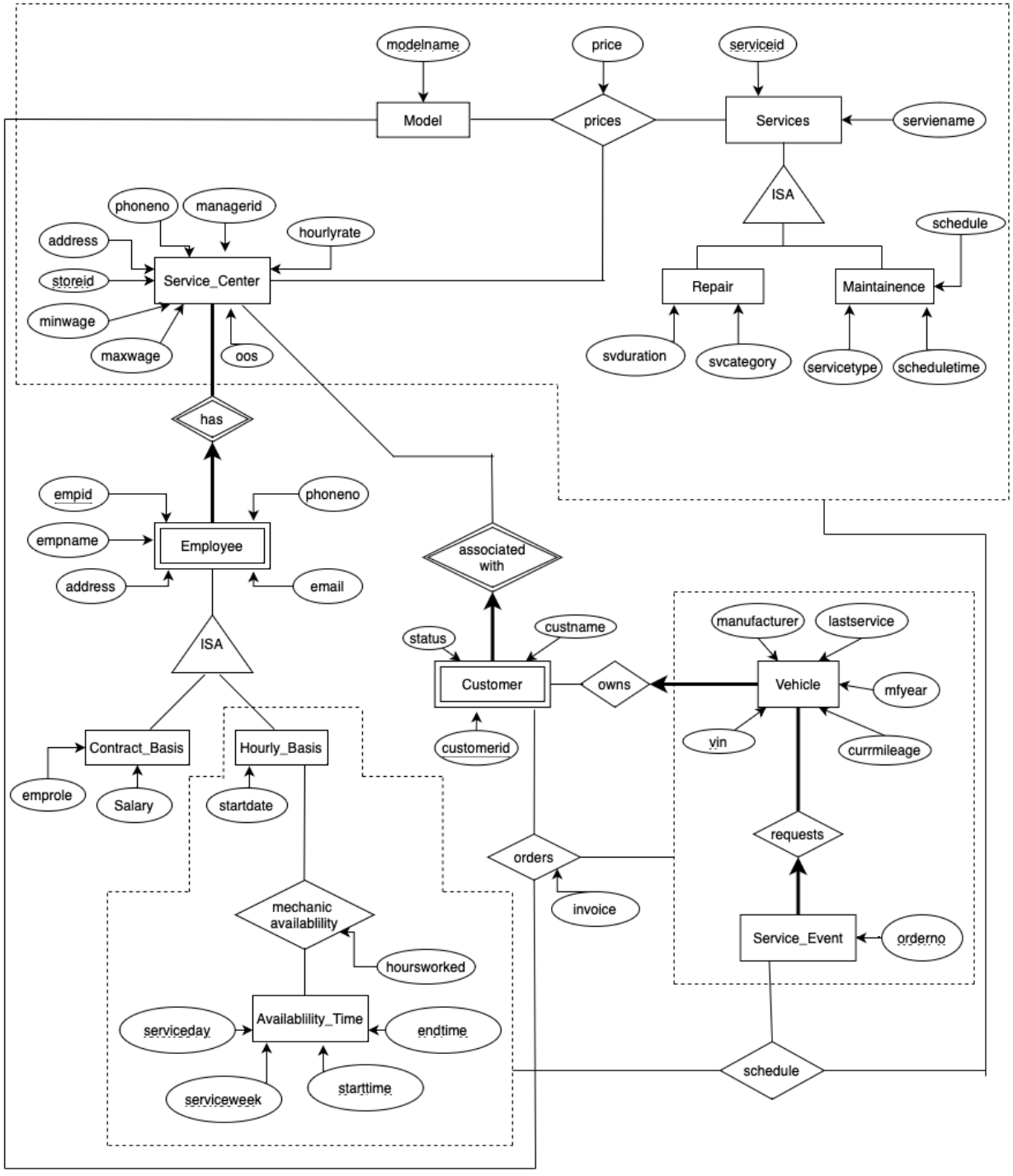
Final Report for Project-1

Team Members - T16

Aditi Salunkhe	Neel Shah	Purv Patel	Sourabh Wattamwar
acsalunk	npshah6	ppatel36	sswattam

GitHub: https://github.ncsu.edu/sswattam/CSC_540_Project_T16

1. Complete ER Model:



2. Textual description of E-R model:

→ Entity Service_Centre:

- {storeid, address, phoneno, oos, managerid, minwage, maxwage, hourlyrate}
- The attribute storeid acts as primary key
- It is the parent entity for Employee entity and is linked to it by has relationship
- Each Service Center has 1 or more employees however each employee has one service center
- It is the parent entity for Customer entity and is linked to it by associated with relationship
- Each Service Center has 0 or more customers however each customer has one service center
- We have a check constraint which checks that the hourly rate is greater than equal to min wage and less than equal to max wage.
- Another check constraint is used to make sure that the value of oos(open on saturday) is binary where 1 signifies open and 0 closed.

→ Entity Contract_Basis:

- {empid, storeid, empname, address, phoneno, email, salary, emprole}
- It is a weak entity whose parent entity is Service Center and linked by has relation.
- The attribute empid is locally unique and together with storeid they form the primary key.
- As it's a weak entity all employees are associated with exactly one Service Center which is indicated by the arrow from Employee to has relationship.
- We have a check constraint which is used to make sure that the value of the role attribute is binary in nature where 1 signifies manager and 0 receptionist.
- Another check constraint is used to make sure that every store has exactly one receptionist and this is implemented in such a way that whenever the manager tries to add a new receptionist if that store already has one receptionist, it will discard this entry.

→ Entity Hourly_Basis:

- {empid, storeid, empname, address, phoneno, email, startdate}
- It is a weak entity whose parent entity is Service Center and linked by has relation.
- The attribute empid is locally unique and together with storeid they form the primary key.
- As it's a weak entity all employees are associated with exactly one Service Center which is indicated by the arrow from Employee to has relationship.
- We have implemented a trigger which makes sure that before entering a mechanic data, there should be a receptionist associated with that store.

→ Entity Vehicle

- {vin, storeid, customerid, manufacturer, lastservice, mfyear, currmileage}
- The attribute vin acts as primary key
- It is linked with Customer entity by the relation owns.
- Each Vehicle is owned by exactly one customer which is denoted by the bold arrow from vehicle to owns relationship.
- Check constraint is used to make sure that the value in lastservice is from the set{A, B, C} which denotes the last maintenance service done on that particular car.

→ Entity Customer

- {customerid, storeid, custname, status}
- It is a weak entity whose parent entity is Service Center and linked by associated with relation.
- The attribute customerid is locally unique and together with storeid they form the primary key.
- As it's a weak entity all customers are associated with exactly one Service Center which is represented by the arrow from Customer to associated with relationship.
- The customer is further linked with the Vehicle entity by own relationship.
- Each customer is linked with zero or more vehicles.
- Check constraint is used to make sure that the value of status is binary where 1 denotes active customers and 0 inactive customers i.e. customers who do not own any vehicles.

→ Entity Repair_Service

- {serviceid, servicename, svcategory, svduration}
- The serviceid attribute acts as the primary key
- All Repair services are further divided into 6 subcategories - engine, electrical, transmission, heating_AC, tire and exhaust.
- We make sure that if any new repair service is added, it's category is one of the above six.

→ Entity Maintenance_Service

- {serviceid, servicename, schedule, servicetype, scheduletime}
- The serviceid along with servicename attribute acts as the primary key
- All Maintenance Services are part of either Schedule A or B or C and they have a cumulatively increasing nature.
- We have a check constraint which makes sure that servicetype is binary in nature where 0 denotes only Maintenance service whereas 1 denotes both Maintenance and Repair service.

→ Entity Auth

- {username, userpassword, storeid, emprole}
- The username along with userpassword attribute acts as the primary key.
- This entity is used to identify the user based on their role and display the appropriate dashboard to them.
- There are in total 5 roles - Admin, Manager, Receptionist, Mechanic, Customer and each have a different dashboard.

→ Entity Prices

- {storeid, serviceid, modelname, price}
- The storeid along with serviceid and model attributes acts as the primary key.
- We know that different service centres have different prices for the same serviceid and the same model.
- Also for the same service center there are different prices for the same service but different models.

→ Relation Orders

- {orderno, vin, storeid, customerid, modelname, invoicestatus}

- The orderno(invoice number) attribute acts as the primary key.
- This table is used to store all orders that have been placed by customers across all service centers and all cars of that particular customer.
- It is an aggregation relationship between Customer, Vehicle and Prices tables which helps us provide details about a service on a particular car of a customer.
- We have a check constraint which makes sure that the value of invoicestatus is binary in nature such that by default the value is 0 which denotes unpaid and when customer pays for that order the value is set to 1 which signifies paid.

→ Relation Schedule
- {orderno, servicename, serviceday, serviceweek, startslot, endslot, empid, storeid}
- The orderno(invoice number) and servicename attribute acts as the primary key.
- This table is used to store all the schedule related information like which mechanic is working on which orderno and on what day of the week.
- It is an aggregation relationship between Services, Orders and Mechnaic_Availability which helps us provide details about which mechanic is available to work on which order and the service duration from Services table.
- Based on details from this table a mechanic can view his upcoming shifts.

→ Relation Mechanic_Availability
- {empid, storeid, serviceday, serviceweek, starttime, endtime, hoursworked}
- The {empid, storeid, serviceday, serviceweek} attribute acts as the primary key.
- This table is used to store all information about the availability of the mechanics and the number of hours they have worked as of now.
- Based on details from this table mechanics can decide to swap their shifts or ask for time off.

→ Relationship has
- It links the weak entity employee and its parent entity Service_Center
- This relationship ensures that each employee is associated with exactly one service center and each service center has 1 or more employees.

→ Relationship associated with
- It links the weak entity Customer and its parent entity Service_Center
- This relationship ensures that each customer is associated with exactly one service center.

→ Relationship owns
- It links the Vehicle entity with the weak entity Customer.
- This relationship ensures that each vehicle is associated with exactly one customer and there may be some customers without any vehicle.

→ Relationship provides
- It is a relationship between the Service Center and Service entity.
- It has its own attribute price which determines the price of a particular service of a particular model at a particular store.

3. Application Design and Implementation:

1. **Start Pages -**

→ Home: At the start of the program, the user has two options, either to Login into the system or else Exit the program. When the user selects option (1) he/she is taken to the login page, if the user selects (2) the program is terminated.

→ Login: Once the user selects to Login, he/she has to enter the User ID and Password, which will be authenticated by the system and only then the user will be allowed to enter the system. If the entered credentials are invalid, a message saying Invalid Credentials is shown and the user is not allowed to login. Once the user is logged in, he/she is taken to the Landing page according to their assigned role. If the user selects option (2), he/she is taken back to the Home page.

2. **Admin Pages -**

→ Landing: If the user's assigned role is Admin, he/she is directed to this page once logged in. Here the user is given 4 options, System Set Up, Add New Store, Add New Service, and Logout. The user is then redirected to the respective page when he/she selects the particular option. If the user selects option (4), he/she is logged out successfully.

→ System Set Up: Here the admin has two options. The first one allows the admin to upload a file which has general information about the services. Another option allows the admin to upload a file that has information about the store. A message is displayed whether the file has been uploaded successfully or not. If the admin chooses option (3), he/she is taken to the Landing page.

→ Add New Store: The admin can add Store details manually by using this option. If the admin chooses option (3), he/she is taken to the Landing page.

→ Add New Service: The admin can add Service details manually by using this option. If the admin chooses option (3), he/she is taken to the Landing page.

3. **Customer Pages -**

→ Landing: When the user's assigned role is Customer, he/she is directed to this page once logged in successfully. Here the customer is given 4 choices, he/she can view and update their profile, view and schedule the services, check their invoices and Logout once done.

→ View and Update Profile: The customers can view their profiles which were set by the Receptionist, here they can add/delete a particular vehicle. When a customer selects the Go Back option, they are redirected to the customer landing page.

→ View and Schedule Service: Here the customer can view and schedule services. By using the View Service History option, customers can get the details about past services. The customers can schedule service appointments by selecting the Schedule Service option. The customer can separately add Repair and Maintenance services. Once the services are added to the cart, the customer can select the available slots for servicing. When a customer selects the Go Back option, they are redirected to the customer landing page.

→ Invoices: By selecting the invoices option, customers can view all the invoices and also have the option to pay the pending invoices. When a customer selects the Go Back option, they are redirected to the customer landing page.

4. **Receptionist Pages -**

→ Landing: When the user's assigned role is Receptionist, he/she is directed to this page once logged in successfully. Receptionist has been given 3 options to choose from, he/she can add a new customer profile, view all the customers having pending invoices and Logout.

→ Add New Customer Profile: Only the Receptionist of a store can add a new customer profile through this option. He/she has to fill all the customer details and save the information to add customers to that particular service store. The receptionist can select the Go Back option which will take him/her to the Landing page.

→ Find Customers with Pending Invoices: From this option the receptionist can get a list of customers who haven't paid their invoices, along with the details of pending invoices. The receptionist can select the Go Back option which will take him/her to the Landing page.

5. **Manager Pages -**

→ Landing: When the user's assigned role is Manager, he/she is directed to this page once logged in successfully. The Manager can set up the store by adding additional details, he/she can add new employees as required. And can Logout of the system once the work is done.

→ Setup Store: When an Admin adds a new store, he/she adds general information regarding that store. A manager, by using this option, can add additional details. Like, new employees, store operational hours and service prices for different models. The manager can select the Go Back option which will take him/her to the Landing page.

→ Add New Employee: The Manager can directly add new employees using this option. He/she has to manually enter the employee details and select save. The manager can select the Go Back option which will take him/her to the Landing page.

6. **Mechanic Pages -**

→ Landing: When the user's assigned role is Mechanic, he/she is directed to this page once logged in successfully. Mechanic is given 5 options to choose from, he/she can view their work schedule, they can request for time-off or swap their schedules, they can also accept/reject a colleagues swap request and Logout once work is done.

→ View Schedule: This page will provide the schedule details of that particular mechanic, by analyzing the schedule the mechanic can then request for time off or swap if required. The mechanic can select the Go Back option which will take him/her to the Landing page.

→ Request Time-off: The mechanic can send a request for time-off, it will be approved automatically if at least 3 employees are working during that shift. The mechanic can select the Go Back option which will take him/her to the Landing page.

→ Request Swap: The mechanic can send a shift swap request to another mechanic. The mechanic can select the Go Back option which will take him/her to the Landing page.

→ Accept/Reject Swap: When a mechanic sends a swap request to another, they can either accept or reject the request using this option. The mechanic can select the Go Back option which will take him/her to the Landing page.

4. Functional Dependencies:

1. In Service_Center Table -
   1. address → {storeid, managerid, phoneno, minwage, maxwage, hourlywage, oos}
   2. phoneno → {storeid, managerid, address, minwage, maxwage, hourlywage, oos}
   3. managerid → {storeid, phoneno, minwage, maxwage, hourlywage, oos, address}
   4. Storeid → {address, phoneno, oos, manergerid, minwage, maxwage, hourlywage}

2. In Contract_Basis Table -
   1. {empid, storeid} → {empname, address, phoneno, email, salary, emprole}
   2. address → {empid, storeid, empname, phoneno, email, salary, emprole}
   3. phoneno → {empid, storeid, empname, address, email, salary, emprole}
   4. email→ {empid, storeid, empname, address, phoneno, salary, emprole}

3. In Hourly_Basis Table -
   1. {empid, storeid} → {empname, address, phoneno, email, startdate}
   2. address → {empid, storeid, empname, phoneno, email, startdate}
   3. phoneno→ {empid, storeid, empname, address, email, startdate}
   4. email→ {empid, storeid, empname, phoneno, address, startdate}

4. In Repair_Service Table -
   1. serviceid → {servicename, svcategory, svduartion}
   2. servicename → {serviceid, svcategory, svduartion}

5. In Maintenance_Service Table -
   1. {serviceid, servicename} → {schedule, servicetype, scheduletime}
   2. schedule → schdeuletime

6. In Prices Table -
   {storeid, serviceid, modelname} → {price}

7. In Auth Table -
   {username, userpaswword} → {emprole, storeid}

8. In Customer Table -
   {customerid, storeid} → {custname, status}

9. In Vehicle Table -
   vin → {storeid, customerid, manufacturer, lastservice, mfyear, currmileage}

10. In Orders Table -
    1. orderno → {vin, storeid, customerid, modelname, invoicestatus}
    2. vin → {customerid, modelname, storeid}

11. In Schedule Table -
    1. {orderno, servicename} → {serviceday, serviceweek, startslot, endslot, empid, storeid}
    2. {serviceday, serviceweek, startslot, endslot, empid, storeid} → {orderno, servicename}

12. In Mechanic_Availability Table -
    {storeid, empid, serviceday, serviceweek} → {starttime, endtime, hoursworked}

- For FD's in the Service_Centre table, none of them are trivial FD and for all of them the left hand side contains a candidate key. Hence all FD's of Service_Centre are in BCNF form.
- For FD's in the Contract_Basis table, none of them are trivial FD and for all of them the left hand side contains a candidate key. Hence all FD's of Contract_Basis are in BCNF form.
- For FD's in the Hourly_Basis table, none of them are trivial FD and for all of them the left hand side contains a candidate key. Hence all FD's of Hourly_Basis are in BCNF form.
- For FD's in the Repair_Service table, none of them are trivial FD and for all of them the left hand side contains a candidate key. Hence all FD's of Repair_Service are in BCNF form.
- For FD's in the Maintenance_Service table, none of them are trivial FD and for the 1st FD left hand side contains a candidate key. But for 2nd FD the left hand side does not contain a candidate key but the right hand side is part of a candidate key. Hence Repair_Service is in 3NF form.
- For FD in the Prices table, the left hand side contains a candidate key. Hence the given FD of Price is in BCNF form.
- For FD in the Auth table, the left hand side contains a candidate key. Hence the given FD of Auth is in BCNF form.
- For FD in the Customer table, the left hand side contains a candidate key. Hence the given FD of the Customer table is in BCNF form.
- For FD in the Vehicle table, the left hand side contains a candidate key. Hence the given FD of Vehicle is in BCNF form.
- For FD's in the Orders table, none of them are trivial FD and for the 1st FD left hand side contains a candidate key. But for 2nd FD the left hand side does not contain a candidate key but the right hand side is part of a candidate key. Hence Orders is in 3NF form.
- For FD's in the Schedule table, none of them are trivial FD and for the 1st FD left hand side contains a candidate key. But for 2nd FD the left hand side does not contain a candidate key but the right hand side is part of a candidate key. Hence Schedule is in 3NF form.

Thus in total we have some tables that follow BCNF form while some are in 3NF form. **Hence, collectively our database design is in 3NF form.**

5. Constraints:

Description	Constraint
Service_Centre	
Each service center is identified by a globally unique ID(storeid).	Primary Key
Open on Saturdays (oos) attribute has to be a boolean value	CHECK constraint - CHECK (oos in (1,0))
The general employee structure in each center has a manager	Not Null in Service Center Table (managerid)
Hourly rate for each mechanic of the store should lie in between the minwage and maxwage amount	CHECK (minwage <= hourlyrate AND hourlyrate <= maxwage)
Contract_Basis	
Each employee hired on a contract basis has a locally unique ID(empid) and each employee is associated with only one Service Centre	{storeid + empid} as Primary Key
Contract_Basis is a weak entity	Foreign Key
Role for each employee should be either ‘ 1 - Manager or 0 - Receptionist	check (emprole in (1,0))
There should be only one receptionist per store.	
Hourly_Basis	
Each employee hired on a hourly basis has a locally unique ID(empid) and each employee is associated with only one Service Centre	{storeid + empid} as Primary Key
Hourly_Basis is a weak entity	Foreign Key
Auth	
Each user logging into the system is validated for his/her credentials in the Auth table. Each tuple is uniquely identified by the username and user password	{username + userpassword} as Primary Key
Repair_Service	
Each repair service is uniquely identified by its service id.	Primary Key

A new Repair Service entered should not belong to any category other than the pre-existing categories.	CHECK Constraint: CHECK svcategory IN (‘Engine Services’, ‘Exhaust Services’, ‘Electrical Services’, ‘Transmission Services’, ‘Tire Services’, ‘Heating and A/C Services’)
Maintenance_Service	
Each maintenance service is uniquely identified by the service id and the service name as there are multiple service names under one Maintenance Schedule	{serviceid + servicename} as Primary Key
Service type is either m (denoted by 0) or m,r (denoted by 1)	check (servicetype in (1,0))
We cannot add a new Maintenance Schedule. We can just add new services to the pre existing schedules.	check (schedule in (‘A’, ‘B’, ‘C’))
Prices	
Each tuple is uniquely identified by storeid, service id and model name	{storeid + serviceid + modelname} as PRIMARY KEY
Customer	
Each customer is identified locally by a 9-digit customer id. So, Customer is a weak entity supported by Service Centre	{storeid + customerid} as PRIMARY KEY
As Customer is a weak entity. We have store id as a foreign key which references the Service Centre table.	Foreign Key
Customer status is either active(1) or inactive(0)	CHECK(status = 1 OR status = 0)
Vehicle	
Each vehicle is uniquely identified by a globally unique VIN number	Primary Key
Customer id and store id is referenced from Customer table and Service Centre table respectively	Foreign Key
The last maintenance service has to be either ‘A’ or ‘B’ or ‘C’	CHECK(lastservice IN ('A', 'B', 'C'))
Mechanic Availability	

Each row is uniquely identified by the combination store id, employee id, serviceday, serviceweek, starttime, endtime	{storeid, empid, serviceday, serviceweek, starttime, endtime} AS PRIMARY KEY
Employee id and store id is referenced from the Hourly_Basis table.	{storeid, empid} AS FOREIGN KEY
Hours worked per week of a mechanic should not exceed 50	CHECK Constraint- CHECK(hoursworked <= 50)
Orders	
Each tuple in Orders table is uniquely identified by order id, storeid, customerid, vin, modelname	Primary Key
Customer id and Store id is referenced from Customer table and vin is referenced from Vehicle table	Foreign Key
Invoice Status is either 0 (unpaid) or 1 (Paid)	CHECK Constraint- CHECK (invoicestatus in (0,1))
Schedule	
Each tuple is identified uniquely by the combination of order no, servicename, serviceday, serviceweek, startslot, endslot, storeid, empid	{storeid, empid, serviceday, serviceweek, starttime, endtime} AS PRIMARY KEY
Order no is referenced from Service Event	Foreign Key
Service Day, Service Week, Start time and End time is referenced from AvailabilityTime	Foreign Key
Service name, store id is referenced from Services	Foreign Key
AvailabilityTime	
Each tuple is uniquely identified by the service day, service week, start time and end time	Primary Key
Model	
Each tuple is uniquely identified by the model name	Primary Key
Service_Event	
Each service event is uniquely identified by order no	Primary Key