Milestone 2 & Final Report

Team member

Wenting Zheng (wzheng8)

Yusi Qi (yqi3)

Tianming Liu (tliu18)

### Final Report

Main Tasks:

1. SQL file one: should contain SQL for triggers, tables, constraints, procedure.
2. SQL file two: should contain queries for populating the tables with the sample data. Sample data will be provided closer to demo date.
3. Executable file (e.g. - Executable JAR file) and source Java Code.

TODO list:

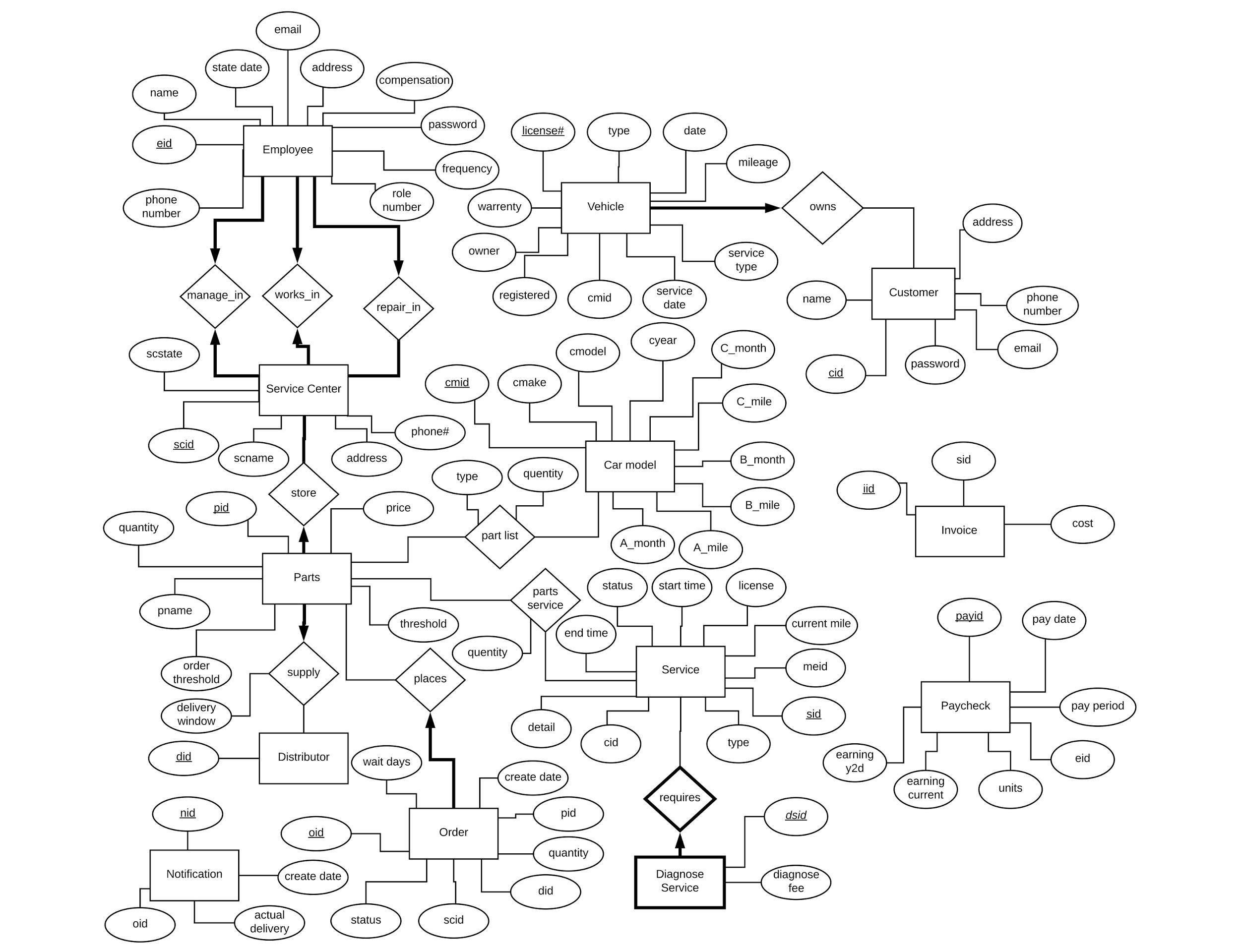
1. Triggers, procedures(based on the application flow)
2. 2nd SQL file
3. Java with SQL(only call the procedures)

(最后)README.txt - This should contain the names of the members and any additional instructions that you think will be necessary to compile and execute your code.

Tables:

### Milestone 2

1. ER diagram



2. Tables & Description

\* the primary keys are underlined

ServiceCenter(scid: integer, scname: string, address: string, phone: integer, scstate: string);

* Constraints: each service center has one manager, one receptionist, and at least five mechanics. This constraint will be represented by manages\_in, works\_in, and repairs\_in relation tables.

Employee(eid: integer, password:string, e\_name: string, address: string, email: string, phone: string, role\_number: integer, start\_date:date, compensation: float, frequency:string);

* password: cannot be NULL. The password is initially set to “12345678” when a new employee is created.
* e\_name: cannot be NULL.
* address: cannot be NULL.
* email: cannot be NULL.
* phone: cannot be NULL.
* start\_date: cannot be NULL.
* compensation: cannot be NULL.
* Manager, Receptionist, and Mechanic has a ISA relationship with Employee with covering constraints.
* role\_number: cannot be NULL, 0 represents mechanic, 1 represents receptionist, 2 represents manager.
* The compensation of the manager and receptionist will be based on the monthly salary while the compensation of the mechanic will be based on the hourly salary and the total number of hours.
* Frequency: the Employees get paid 1st and 15st

manages\_in(scid: integer, eid: integer);

* scid: cannot be NULL.
* eid: cannot be NULL.
* Constraint: scid and eid are foreign keys from ServiceCenter and Employee tables respectively.
* The eid should only be the employee ID of the manager whose role\_number is 2.

works\_in(eid: integer, scid: integer);

* scid: cannot be NULL.
* eid: cannot be NULL.
* Constraint: scid and eid are foreign keys from ServiceCenter and Employee tables respectively.
* The eid should only be the employee ID of the receptionist whose role\_number is 1.

repairs\_in(scid: integer, eid: integer);

* Constraint: scid and eid are foreign keys from ServiceCenter and Employee tables respectively. Every service center must have at least five mechanics, which means at least five different eid have the same scid in the table.
* The eid should only be the employee ID of the mechanic whose role\_number is 0.

Paycheck(pay\_id: string, pay\_date: date, pay\_period: string, eid: integer, units: float, earning\_current: float, earning\_y2d: float);

* eid is the foreign key from Employee table. Each eid has only one employee.
* Units is used to calculate the compensation of the employees. This presents how many hours/days the employee worked.
* Earning\_current represents the current earning and the earning\_y2d represents the year-to-date earning.

stores(scid: integer, pid: integer)

* Constraint: scid and pid are foreign keys from ServiceCenter and Part tables respectively

Part(pid: integer, name: string, qty: integer, qty\_tsh: integer, order\_tsh: integer, price: float):

* Constraint: if the qty(quantity) is less than qty\_tsh(quantity threshold), an order must be created. The quantity of the part in each order has to be at least order\_tsh(order threshold).

supplies(pid: integer, did: integer, delivery: int)

* Constraint: pid and did are foreign keys from Part and Distributor tables respectively
* The delivery represents the days required for the Part to be delivered

Distributor(did: integer);

Order(oid: integer, create\_date: date, pid: integer, qty: integer, s\_scid: integer, s\_did: integer, p\_scid: integer, status: string, wait\_days: integer);

* pid and scid are the foreign keys from the Part, ServiceCenter tables respectively
* s\_scid is the foreign key from either ServiceCenter table. If all ServiceCenter don’t have the demanded amount of parts, s\_scid is set to NULL
* s\_did is the foreign key from Distributor table. If there is no enough Part to be ordered from the other ServiceCenter, the Part will be ordered from the Distributor. If s\_scid is not NULL, s\_did is NULL. If s\_scid is NULL, s\_did is not NULL.
* p\_scid is the scid of the ServiceCenter which placed the order.
* The status will start with “Pending” and change to “Complete” when the order is received by an employee.
* Note: this new Order table is the combination of the places and Order tables. The places relation table is eliminated.

Notification(nid: integer, create\_date: date, oid: integer, actual\_delivery: date);

* Constraint: oid is the foreign key from Order table.
* The Notification is generated if the an Order doesn’t arrive on time. The expected delivery date can be calculated by adding the create\_date and the wait\_days in the Order table.
* The manager can view all the Notifications

Customer(cid: integer, name: string, email: string, address: string, phone: integer);

* cid: cannot be NULL.
* name: cannot be NULL.
* address: cannot be NULL.
* email: cannot be NULL. The Customer will use the email to login, so the email address must be unique.
* password: cannot be NULL.
* phone: cannot be NULL.

Car\_Model(cmid: string, cmake:string, cmodel:string, cyear:string, A\_mile: integer, A\_month: integer, B\_mile: integer, B\_month: integer, C\_mile: integer, C\_month: integer);

* Car type = Honda/Nissan/Toyota
* New car type can be added by a manager.
* Each Car\_Model has different service A/B/C. The parts required can be found in the part\_list relation table.

part\_list(cmid: string, pid: integer, type: int, qty: integer);

* The cmid and pid are the foreign keys from Car\_Model and Part tables.
* The value of the type will be 0/1/2. 0 is the parts for service A. 1 is the additional parts for service B. 2 is the additional parts for service C. Therefore, <= 1 represents all the parts for service B, and <= 2 represents all the parts for service C.

Vehicle (license#: string, Pdate: date, cmid: string, mileage: integer, serviceDate: date, warranty: integer, owner: integer, serviceType: string, registered: bit);

* owner is the foreign key from the Customer table. Each customer can have 0…\* vehicles, but vehicle only has one owner.
* cmid is the foreign key from the Car\_Model table. Each vehicle could only one type.
* Note: the new Vehicle table is the combination of the owns and previous Vehicle tables. The owns relation table is eliminated.

Service(sid: integer, service\_type: string, start\_time: date, end\_time: date, status: string, detail: string, license: string, currentMile: int, meid: integer, cid: integer);

* Constraint: cid and license are the foreign keys from Customer and table respectively.
* Constraint: meid is the foreign key from Employee table. If the customer doesn’t select specific mechanic, it is set to NULL.
* The service\_type can be “M” for Maintenance or “R” for Repair.

parts\_service(sid: integer, pid: integer, qty: integer);

* Constraint: pid and sid are foreign keys from Part and Service tables respectively
* This table is a relation table represents how many each part is required for a service.

DiagnosticService(dsid: integer, rsid: integer, fee: float);

* rsid is the foreign key from the Service table.
* The DianosticService is a weak entity of the Service table. The primary key is the combination of the dsid and foreign key rsid. If one Service is deleted, the related DiagnosticServices are deleted.
* Note: this table is the combination of the requires relation table and previous DiagnosticService table.

Invoice(iid: integer, sid: integer,cost: float);

* sid is the foreign key from the Service table. ~~If the rsid in Service table, the rsid in Invoice table will also be deleted.~~