Final Project: Database for Car Dealership

By: Matthew Falkoski, Joon Jung, and Ryu Muthui.

Instructor: Dr. Erika Parsons

UW Bothell CSS 475

December 11, 2015

Contents

1 Introduction

This document provides an overview for the database developed by team Fast3 for the final project. The following sections outline the application, the entities that the database tracks, provide some examples of queries, and a list of milestones.

expand intro . . .

2 Database Application

The application developed is an inventory database for the Fast3 used car dealership. The database will allow the dealership to complete many of its required operational tasks. These tasks include (but are not limited to):

- Sales orders
- Vehicle inventory
- Service orders
- Part inventory
- Employee information
- Customer information

In addition to managing inventory, the database will allow the management to manage employees.

3 Entities

The database will hold all of the information related to the dealership. A list of entities and their attributes is provided below. All of the entities that will be in the database are listed. Figure 1 and 2 show the relations between them.

- Dealership
  - dealership id, address id, manager id, phone number
- Address
  - address id, unit number, street, city, state, zip code

- Vehicle Inventory
  - vin, dealership id, date acquired, cost, price, quantity
- Vehicle
  - vin, make, model, type, year manufactured, mileage, color
- Employee
  - employee id, ssn, f name, m initial, l name, address id, phone number, salary, birth date, hire date, dealership id
- Customer
  - customer id, f name, m initial, l name, address id, phome number, vin
- Vendor
  - vendor id, name, address id, phone number, type
- Part
  - part id, make, model, type, year manufactured, state, vendor id, dealership id
- Part Inventory
  - part id, dealership id, cost, price, quantity
- Service
  - service id, description, type, cost, time estimate, dealership id, part id
- Service Record
  - record, service id, customer id, vin, scheduled date, balance, amount paid, additional note
- Sale
  - refrence number, vin, sale price, sale date, dealership id, employee id, customer id

4 Design

The following sections outline the design of the database for the Fast3 dealership.

4.1 Assumptions

- Dealership provides car sales & services.
- Dealership sells both new and used cars.
- Dealership uses both new and used parts to fix cars.
- Employees only can work at one physical location of dealership.
- It is commission based job so only one salesperson associates on one vehicle sale.
- Vehicle only purchases by one customer.
- Customers can have vehicle even though they did not purchase vehicle from dealership.
- Customers may own multiple vehicles.
- Dealership may not get part that they ordered from vendor.
- Some of services do not require using part (e.g. balancing tire, diagnosis engine, etc).
- Database stores only WA state address in current design but set up for nation wide.

4.2 ER-Diagram

Figure 1 shows the ER diagram developed for the Fast3 database.

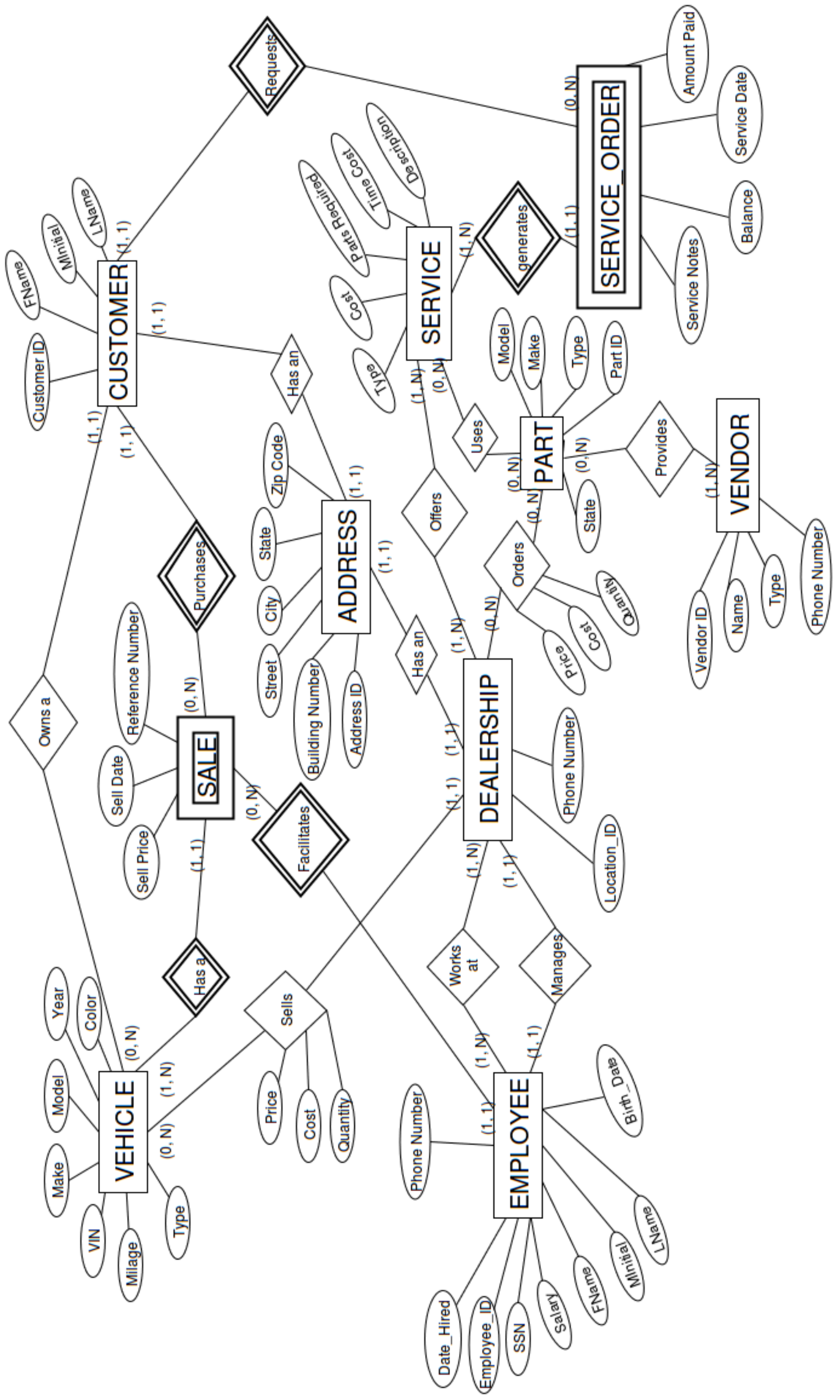

Figure 1: ER Diagram for the Fast3 database.

4.3 Relational Model

Figure 2 shows the relational model that was developed from the ER-diagram.

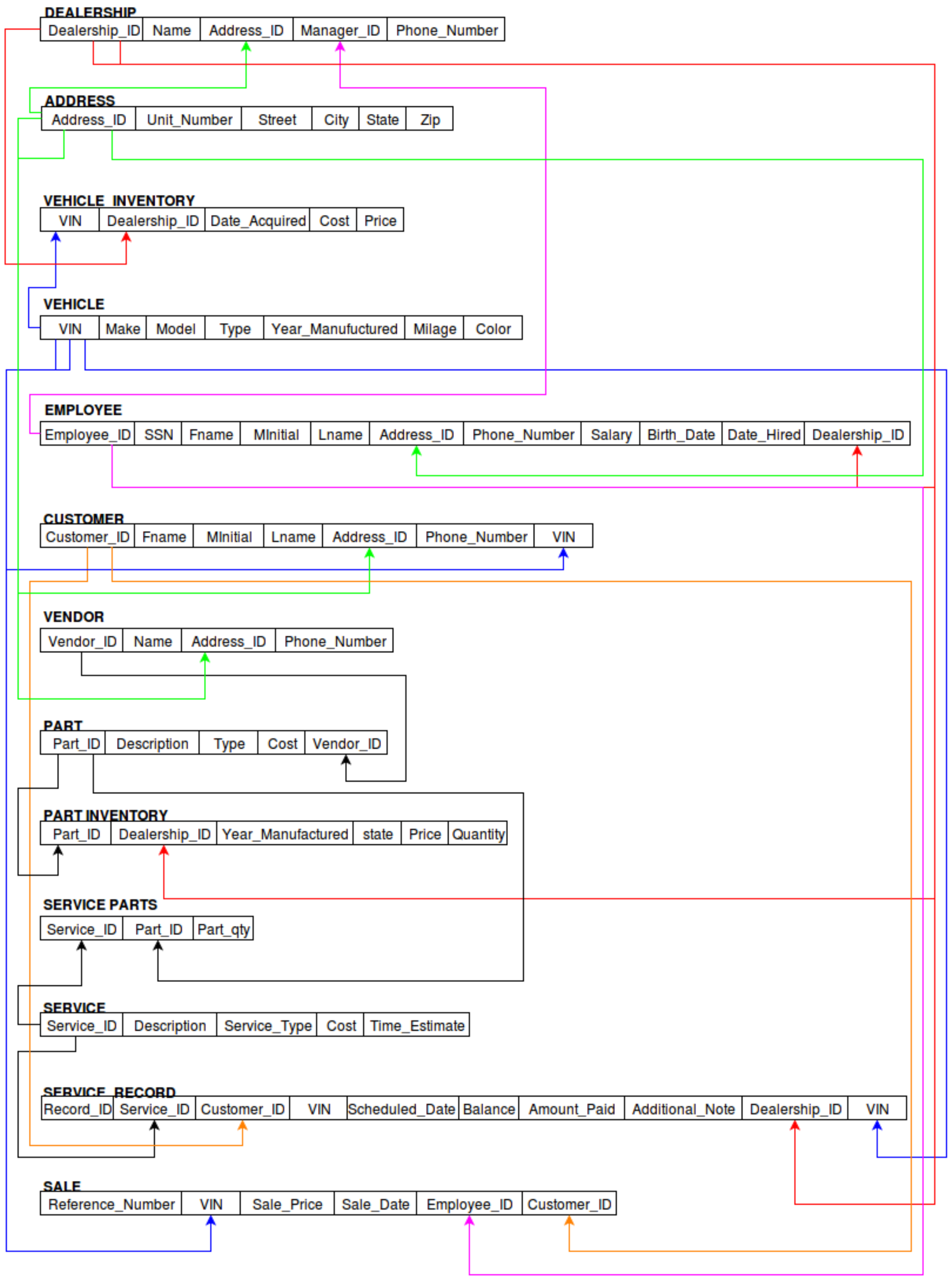
DEALERSHIP
Dealership_ID | Name | Address_ID | Manager_ID | Phone_Number
ADDRESS
Address_ID | Unit_Number | Street | City | State | Zip
VEHICLE_INVENTORY
VIN | Dealership_ID | Date_Acquired | Cost | Price
VEHICLE
VIN | Make | Model | Type | Year_Manufuctured | Milage | Color
EMPLOYEE
Employee_ID | SSN | Fname | MInitial | Lname | Address_ID | Phone_Number | Salary | Birth_Date | Date_Hired | Dealership_ID
CUSTOMER
Customer_ID | Fname | MInitial | Lname | Address_ID | Phone_Number | VIN
VENDOR
Vendor_ID | Name | Address_ID | Phone_Number
PART
Part_ID | Description | Type | Cost | Vendor_ID
PART INVENTORY
Part_ID | Dealership_ID | Year_Manufactured | state | Price | Quantity
SERVICE PARTS
Service_ID | Part_ID | Part_qty
SERVICE
Service_ID | Description | Service_Type | Cost | Time_Estimate
SERVICE_RECORD
Record_ID | Service_ID | Customer_ID | VIN | Scheduled_Date | Balance | Amount_Paid | Additional_Note | Dealership_ID | VIN
SALE
Reference_Number | VIN | Sale_Price | Sale_Date | Employee_ID | Customer_ID

Figure 2: RM Diagram for the Fast3 database.

4.4 Normalization

Once the ER diagram and the relational model were complete, the process of normalization allowed the derived tables to be analysed for deficiencies. The goal is to have all of the tabled in third normal form. Third normal form requires that there be no multi-valued attributes, every dependency must be fully functional, and there should be no transient dependencies. The following are all of the functional dependences for the final tables.

- Dealership
  - dealership id → name
  - dealership id → address id
  - dealership id → manager id
  - dealership id → phone number

  The dealership table is in third normal form. All of the attributes are fully dependent on the dealership id and there are no transient dependencies.

- Address
  - address id → unit number
  - address id → street
  - address id → city
  - address id → state
  - address id → zip code

  The address table is technically in second normal form. State and city are transiently dependent on zip code. This decision was made because it seems like a common practice to format an address table in this manor, and the information is easy to verify on the client side.

- Vehicle Inventory
  - vin → dealership id
  - vin → date acquired
  - vin → cost
  - vin → price

The vehicle inventory table is in third normal form. All of the attributes are fully dependent on the vin and there are no transient dependencies.

- Vehicle
  - vin → make
  - vin → model
  - vin → type
  - vin → year manufactured
  - vin → mileage
  - vin → color

  The vehicle table is in third normal form. All of the attributes are fully dependent on the vin and there are no transient dependencies.

- Employee
  - employee id → ssn
  - employee id → first name
  - employee id → middle initial
  - employee id → last name
  - employee id → address id
  - employee id → phone number
  - employee id → salary
  - employee id → birth date
  - employee id → hire date
  - employee id → dealership id

  The employee table is in third normal form. All of the attributes are fully dependent on the employee id and there are no transient dependencies.

- Customer
  - customer id → first name
  - customer id → middle initial
  - customer id → last name

  - customer id → address id
  - customer id → phone number
  - customer id → vin

  The customer table is in third normal form. All of the attributes are fully dependent on the customer id and there are no transient dependencies.

- Vendor
  - vendor id → name
  - vendor id → address id
  - vendor id → phone number

  The vendor table is in third normal form. All of the attributes are fully dependent on the vendor id and there are no transient dependencies.

- Part
  - part id → description
  - part id → type
  - part id → cost
  - part id → vendor id

  The part table is in third normal form. All of the attributes are fully dependent on the part id and there are no transient dependencies.

- Part Inventory
  - (part id, dealership id) → year manufactured
  - (part id, dealership id) → state
  - (part id, dealership id) → price
  - (part id, dealership id) → quantity

  The part inventory table is in third normal form. All of the attributes are fully dependent on the part id and the dealership id. There are no transient dependencies.

- Service
  - service id → description

  - service id → stype
  - service id → cost
  - service id → time estimate

  The service table is in third normal form. All of the attributes are fully dependent on the service id and there are no transient dependencies.

- Service parts
  - service id → part id
  - service id → quantity

  The service parts table is in third normal form. All of the attributes are fully dependent on the service id and there are no transient dependencies

- Service Record
  - record → service id
  - record → customer id
  - record → vin
  - record → scheduled date
  - record → balance
  - record → amount paid
  - record → additional note
  - record → dealership id

  The service record table is in third normal form. All of the attributes are fully dependent on the record number and there are no transient dependencies

- Sale
  - reference number → vin
  - reference number → sale price
  - reference number → sale date
  - reference number → employee id
  - reference number → customer id

  The sale table is in third normal form. All of the attributes are fully dependent on the reference number and there are no transient dependencies

5 Queries

The following list gives some examples of queries that that will be ran against the database.

- How many employees work at a specific location.
- How many cars have been sold by each employee and the price of the average car sold by the employee.
- Who sold the most vehicles for a sales period.
- What car has the largest number of sales.
- What supplies are low in stock.
- How many customers bought a specific type of vehicle that was sold by a specific employee.
- What vendor has the cheapest parts for a specific model of car.
- What location has the highest average profit margin.

The queries listed are examples of ones that management would use to monitor the operations of the business. The information returned by the queries would allow them to know what vehicles are popular, which salesperson performs best, what supplies need to be ordered, and more.

5.1 Implemented Queries

This section details the queries that were implemented for this project. The code is given in a source listing and it's followed by a brief description of the purpose and view that it relates to.

Listing 1: Function used to generate an Address.

```
SELECT f_name AS First_Name, m_initial AS M_I, l_name AS
   Last_Name,
       salary AS Salary,hire_date AS Date_Hired
FROM employee
WHERE (salary > 100000) AND (hire_date < '2000-01-01')
order by hire_date;
```

The query in listing 1 can be used by a manager for salary and budget calculation purposes. It returns employee name in all the company who earn 100,000 or more who were hired before the year 2000, in the case of promotions and or letting people go type of scenario.

Listing 2: Function used to generate an Address.

```sql
SELECT dealership.name AS Dealership_Name,
COUNT(service_record.service_id) AS
   Total_Number_Of_Services_Per_Dealership
FROM service_record
LEFT JOIN dealership
ON dealership.dealership_id = service_record.dealership_id
GROUP BY dealership.name;
```

The query in listing 2 can be used by a manager or a senior mechanic to find out how many services are done at each dealership.

Listing 3: Function used to generate an Address.

```sql
SELECT dealership.name AS Dealership_Name,
COUNT(service_record.service_id) AS
   Total_Number_Of_Services_Per_Dealership,
SUM(service_record.balance) AS Balance_Due,
SUM(service_record.amount_paid) AS Amount_Paid,
SUM(service_record.balance - service_record.amount_paid) AS
   Difference
FROM service_record
LEFT JOIN dealership
ON dealership.dealership_id = service_record.dealership_id
GROUP BY dealership.name;
```

The query in listing 3 can be used by a manager or a senior mechanic to find out how many services are done at each dealership: the balances owed and have been paid for budgeting and finance purposes.

Listing 4: Function used to generate an Address.

```sql
SELECT dealership.name AS 'Dealership',
COUNT(employee.employee_id) AS '# of Employee'
FROM employee
LEFT JOIN dealership
```

```sql
ON dealership.dealership_id = employee.dealerShip_id
GROUP BY dealership.name;
```

The query in listing 4 is useful to most users of the database. It returns the count of all the employees in each of the dealerships

Listing 5: Function used to generate an Address.

```sql
SELECT DISTINCT (em.employee_id),
        em.f_name AS 'First Name',
        em.l_name AS 'Last Name',
        COUNT(sa.employee_id) AS '# of Car Sold',
                 AVG (sa.sale_price) AS 'Average sold car
                    price'
FROM SALE AS sa, EMPLOYEE AS em
WHERE sa.employee_id = em.employee_id
GROUP BY em.employee_id;
```

The query in listing 5 returns the employees id and name the number of cars sold and the average price of the cars they sold.

Listing 6: Function used to generate an Address.

```sql
SELECT de.name AS 'Available DealerShip',
       ve.year_manufactured AS 'Year',
              ve.color AS 'Color',
       COUNT(ve.vin) AS 'Available Stock'
FROM DEALERSHIP AS de, VEHICLE AS ve,
     VEHICLE_INVENTORY AS vein
WHERE ve.vin = vein.vin  AND ve.make = 'Mini'
      AND de.dealership_id = vein.dealership_id
GROUP BY de.dealership_id;
```

The query in listing 6 is useful for the sales representative for when the customer requests a specific model, it searches through all dealerships, returns the available number of that car, year, and manufactured year.

Listing 7: Function used to generate an Address.

```sql
SELECT pa.description AS 'Part name',
       pain.quantity AS'Qty',
       ve.name AS 'Vendor Name',
       ve.phone_number AS 'Vendor Phone #'
FROM PART AS pa, PART_INVENTORY AS pain,
     DEALERSHIP AS de, VENDOR AS ve
WHERE pa.part_id = pain.part_id
      AND pain.quantity < 4
      AND de.dealership_id = pain.dealership_id
      AND de.name = 'Lynnwood Ford'
      AND ve.vendor_id = pa.vendor_id
GROUP BY pa.description;
```

The query in listing 7 is useful for mechanics: it returns the Part name with quantities below 4 so as to when they need to restock items it also returns the vendor name and vendor phone # for fast access.

Listing 8: Function used to generate an Address.

```
SELECT  em.f_name AS 'SalePerson First Name',
    em.l_name AS 'Last Name',
            MAX(sa.sale_price) AS 'Price Sold',
            sa.sale_date AS 'Purchased Date',
cu.f_name AS 'Customer First Name',
            cu.m_initial AS 'M.I.',
            cu.l_name AS 'Last Name'
FROM SALE AS sa,
    EMPLOYEE AS em,
        DEALERSHIP AS de,
        CUSTOMER AS cu
WHERE sa.sale_date > '2013-01-01'
    AND de.dealership_id = em.dealership_id
    AND em.employee_id = sa.employee_id
          AND cu.customer_id = sa.customer_id;
```

The query in listing 8 is used by sales manager and or sales person. It returns the name of the employee that sold the highest priced car, the date it was sold, and the customer name that bought it.

Listing 9: Function used to generate an Address.

```
SELECT MIN(vein.price) AS 'Cheapest Price Car',
       de.name AS 'Dealership',
       de.phone_number AS 'Phone #',
       ad.street,
       ad.unit_number,
       ad.city,
       ad.state,
       ad.zip_code AS 'ZIP CODE'
FROM DEALERSHIP AS de,
     ADDRESS AS ad,
     VEHICLE_INVENTORY AS vein
WHERE vein.dealership_id = de.dealership_id
      AND de.address_id = ad.address_id;
```

The query in listing 9 can be used by customers or sales reps where it returns the cheapest car available in the inventory of vehicles, and also returns the address and phone number at which dealership is it located at.

Listing 10: Function used to generate an Address.

```
SELECT * FROM service
WHERE stype = 'Tune up'
ORDER BY cost;
```

The query in listing 10 returns all of the Tune Up services offered by the company and is ordered by the cost from low to high.

Listing 11: Function used to generate an Address.

```
SELECT de.name AS 'Dealership',
       COUNT(sere.service_id) AS '# of Serviced car'
FROM SERVICE_RECORD AS sere,
     DEALERSHIP AS de
WHERE sere.scheduled_date > '2015-06-01'
      AND de.dealership_id = sere.dealership_id
GROUP BY de.name;
```

The query in listing 11 returns number of service performed at each dealership later than 2015, June.

6 Data Generation

To populate the database, a combination of data generation scripts and manual data generation was used. Some of the tables will only need a to contain a small number (less than ten) entities to test the database; however, there is a benefit from having many more in others.

To generate the data, a combination of bash and awk scripts have been written. All of the text processing is done with awk; names, addresses, and phone numbers are generated with rig; and bash is used to glue it all together. Listing 12 shows the function used to generate the SQL statements to insert an address. The complete script was submitted as a septate file.

The script generates data for the following tables: address, vehicle inventory, vehicle, employee, customer, part inventory, service record, and sale. The remaining tables have been generated by hand.

Manual data generation was necessary for some of the tables due to the inability to generate correct data automatically. To generate everything automatically would require the data generation script interacting with the already populated tables of the database. In addition to needing knowledge of the information in the database, a flat file of additional data would still need to be generated by hand to have usable information.

Listing 12: Function used to generate an Address.

```
function printAddress {
  DATA=$(rig | awk '
    BEGIN{
         aidx=1;
         srand();
       }
       {
         if (NR==2) {
           fieldCount=split($0,streetAddr," ");
           ADDRESS[aidx++]=streetAddr[1];
           for (i=2; i<=fieldCount; i++) {
             street=street" "streetAddr[i];
           }
           sub(/^[ ]+/, "", street);
           ADDRESS[aidx++]="\x27"street"\x27";
         } else if (NR==3) {
           split($0,addr," ");
           sub(/,/, "", addr[1]);
           ADDRESS[aidx++]="\x27"addr[1]"\x27";
```

```
                ADDRESS[aidx++]="\x27"addr[2]"\x27";
                ADDRESS[aidx++]="\x27"addr[3]"\x27";
            }
        }
    END{
            print ADDRESS[1]", "ADDRESS[2]", "ADDRESS[3]", "
                ADDRESS[4]", "ADDRESS[5];
        }')

  echo -e "INSERT INTO ADDRESS VALUE(NULL, "$DATA");"
}
```

7 Testing

Testing was conducted at each stage of the implementation of the data base. Once the initial script was written to create the database, 0xDBE from Jetbrains was used to generate an ER model of the schema. The generated diagram was checked against the initial ER diagram (figure 1) to identify any problems with the initial implementation.

Once the create database script was creating the database as intended triggers were implemented to guard against bad data. To test the databases ability to handle incorrect input, several invalid insert statements were entered into the database by hand. After all of those passed. The data generation script was modified to produce invalid insert statements.

8 Schedule

Milestones are defined for the four different core deliverable discussed in the iteration one document. The following sections detail the deliverable and their respective milestone dates.

8.1 Documentation Milestone

The documentation includes:

- Entity-Relationship diagram
- Relational data model

The tentative deadline for this milestone is 11-16-2015.

8.2 Database Milestone

The second milestone includes:

- Creating a SQL script to create the database
- Create plan for populating the database
- Document the creation and population of the database

The tentative deadline for this milestone is 11-25-2015.

8.3 SQL Milestone

The third milestone includes:

- Populate the database with sample data
- Creating SQL statements

The queries developed for this milestone will include all of those outlined in section 5. There will also be others that are developed as the project progresses.

The tentative deadline for this milestone is 11-31-2015.

8.4 Presentation Milestone

The fourth milestone includes:

- Creating a PowerPoint presentation
- Assigning each member different talking points
- Practising the presentation

The tentative deadline for this milestone is 12-07-2015.

9 Work Distribution

The work distribution varied from week to week; however, we all worked on every aspect of the project. During the weekly meetings, we used extreme programming techniques where one person would be at the computer (or white board) and the rest would be observing and contributing ideas. After the meeting, we would split up the remaining amount of work that needed to be done.

Working this way allowed us to all stay on the same page and removed the chance that group members would spend time working on the same thing independently.

10 Project Evaluation

The following sections provide a retrospective of the creation of the database for the final project. During the compleation of the project, each team member spent between four and five hours a week working on the database.

10.1 Successes

Several parts of this project worked out better that expected. All of the team members were easily available and able to meet in person once a week for two hours. The time spent working together was extremely productive and allowed all of the remaining tasks to be split up in person.

Aside from the gains made by working together in person, the choice to use scripts to generate data was a big win. The database is populated with several thousand records. Having a large amount of data means that there was no time spent on trying to craft the final queries around what was in the database. It also led to a more realistic scenario where there was know beforehand knowledge of what was in the database.

In addition to making the queries easier to write and saving time, the auto-generation allowed us to generate large amount of invalid statements and use those to test the database. This again saved a large amount of time that would have been otherwise spent manually testing queries.

Lastly, the choice to use git saved a large amount of time and effort. We were able to all work on the same SQL scripts and not have to worry about merging our work together or being out of date. It removed the weekly meet up and merge routine that many have experienced before.

10.2 Issues

During the development of the database a few issues were encountered. The original plan was to host the database on Microsoft's Azure Cloud, but after fighting permission errors for several weeks, the plan was changed to just use a local database of each computer. The databases would remain synchronised by rebuilding the database when a change was made to the test data.
The second issue that was encountered was that more data than initially thought had to be generated by hand. This cause a setback of several days while the necessary tables were generated by hand.
The final issue that we encountered was a

10.3 Lessens Learned

10.4 Improvements

If there was more time to work on the project, several parts could be improved. Mostly, more testing could be preformed to ensure that the database is functioning properly. While the test that have been run are though, that does not guarentee that there are no bugs, or that there are no design oversights.

In addition to testing, more time would allow for more verification of the tables and relationships between them. We would like to gather more domain knowledge to see which assumptions are valid. With a better understanding of how an actual dealership works, we could refine our database.

The last improvement, would be the inclusion of a simple front end. There were initially plans to create a bash script that would use the mysql command line client to connect to the database. The script would act as a command line front end to the database. Due to time constraints, we were unable to complete that portion of the assignment.