# CONNOR'S AUTO SHOP

*CMPT308*

*DESIGNED BY*

*CONNOR LOUGHLIN*

# Table of Contents

# Executive Summary

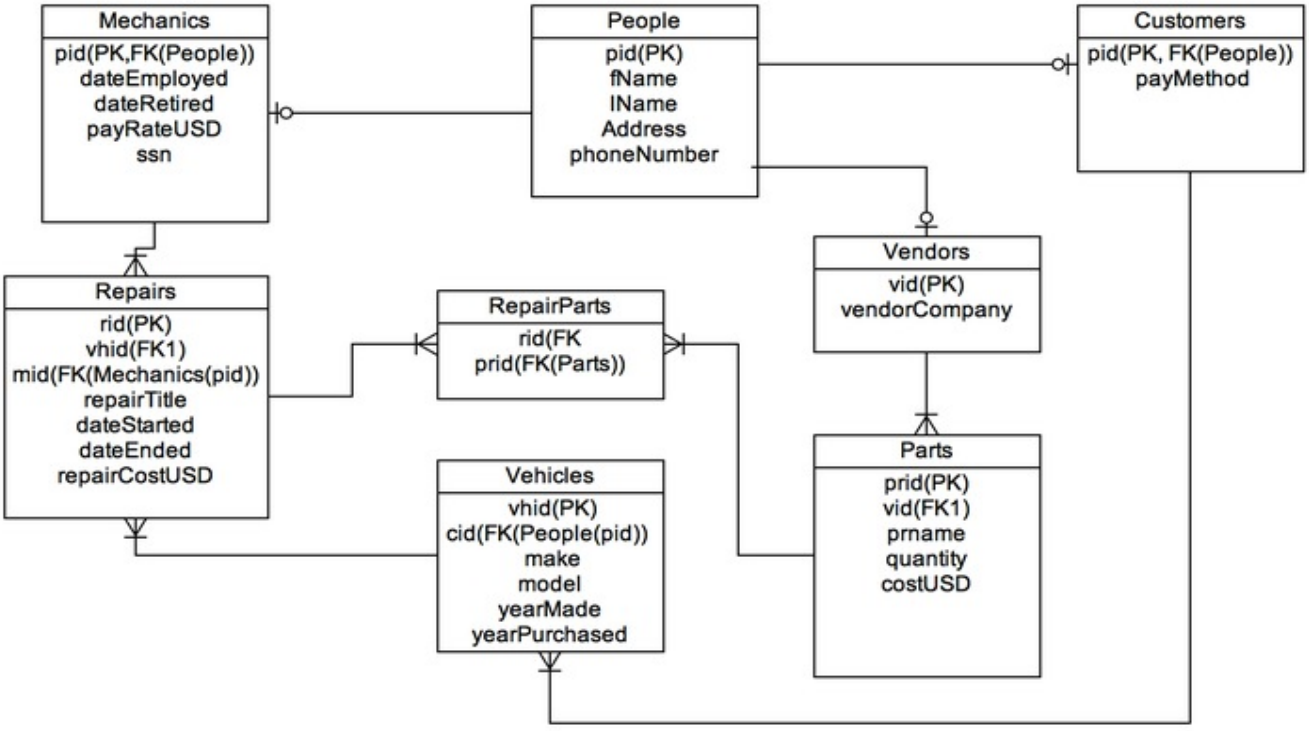
This document serves the purpose of recording the designing and implementing of the database for Connor's Auto Shop. It is essentially completely fictional, but a database nonetheless. IT begins with an ER Diagram, followed by detailed descriptions of each table, including the create statements of each, and sample data for each. This is so the reader can get a good grasp on how the database works. Next, several queries, views, reports, etc. were written to depict just what the database could be used for in a real life situation. Lastly, brief discussions of known errors, and ideas for the future of the database were touched upon.

# Overview and Objectives

This database is essentially going to be used to keep track of day to day procedures at the AutoShop, including the financial transactions. The database should be simple to use, and easy to understand. It should also be secure to prevent outside users from retrieving info on any mechanics, vendors, or customers – especially fianancials.

# Entity Relationship Diagram

This diagram demonstrates the structure of the Auto Shop Database. Below are the descriptions of the tables and all of their fields, as well as how this database will be implemented.




# Tables

**People:**

The People table exists to connect the three different types of persons involved in this database: Mechanics, Customers, and Vendors. Each person listed in this table is either a Mechanic, Customer, or Vendor. Each of these three aforementioned entities inherit all attributes of the People table.

**Functional Dependencies**: pid→ fName, lName, Address, phoneNumber

**Create Statement:**

CREATE TABLE People(

pid serial primary key,

fName text NOT NULL,

lName text NOT NULL,

Address text NOT NULL,

phoneNumber varchar(15) NOT NULL,

);

**Sample Data**:

| | pid integ | fname text | lname text | address text | phonenumber character varying(15) |
|---|---|---|---|---|---|
| 1 | 1 | Connor | Loughlin | 17 Nantucket Drive | 845-555-5555 |
| 2 | 2 | Tim | Rongo | 13 Free Lane | 914-555-5355 |
| 3 | 3 | Alex | Vanderbilt | 27 Bucket Drive | 867-534-2864 |
| 4 | 4 | Nick | Castano | 33 Brave Road | 655-234-0192 |
| 5 | 5 | Steve | Rockafellow | 1776 Freedom Avenue | 201-255-5325 |
| 6 | 6 | Matt | Jameson | 23 Jojo Drive | 203-397-3527 |
| 7 | 7 | Frank | Moody | 6 Delafield Street | 845-895-5505 |
| 8 | 8 | Joe | Shmoe | 58 Hallow Drive | 865-343-6534 |
| 9 | 9 | Andrew | Townes | 17 Utah Street | 369-234-4545 |
| 10 | 10 | Tony | Boyle | 49 Trouble Avenue | 845-390-7789 |
| 11 | 11 | Jomar | Benoit | 85 Wake Street | 823-340-7421 |
| 12 | 12 | Phil | Krupka | 49 Ghastly Lane | 715-250-5689 |

**Customers:**

The Customers table exists to show who from the "People" table are also customers. Customers have one extra attribute that is not in the People table: payMethod. This is required because the autoshop must know whether or not the customer is going to use cash, credit, debit, or insurance. Additionally, since cid = pid, the cids in the customers able are not necessarily uninterrupted numbers(i.e 1, 2, 4, 5,7).

**Functional Dependecies**: cid → payMethod

**Create Statement**:

CREATE TABLE Customers(

cid int REFERENCES People(pid) primary key,

payMethod text NOT NULL

);

**Sample Data**:

| | cid integer | paymethod text |
|---|---|---|
| 1 | 1 | Cash |
| 2 | 2 | Credit |
| 3 | 4 | Debit |
| 4 | 5 | Insurance |
| 5 | 7 | Insurance |

**Mechanics:**

The Mechanics table serves a similar purpose as the customers table: to demonstrate which people are mechanics. There are four additional attributes in the mechanics table: dateEmployed, dateRetired, payRateUSD, and ssn. Obviously, a business is going to know when its employees start and end working. Addtionally, mechanics will have an hourly salary(in US dollars), and a social security number used for financial purposes. Note that 'pid' in this table, is renamed 'mid.'

**Functional Dependencies:** mid → dateEmployed, dateRetired, payRateUSD, ssn

**Create Statement**:

CREATE TABLE Mechanics(

mid int REFERENCES People(pid) primary key,

dateEmployed date NOT NULL,

dateRetired date,

payRateUSD double NOT NULL,

ssn varchar(15)

);

**Sample Data**:

| | mid integer | dateemployed date | dateretired date | payrateusd real | ssn character varying(15) |
|---|---|---|---|---|---|
| 1 | 3 | 2010-01-23 | 2012-03-12 | 23 | 321-12-1234 |
| 2 | 6 | 2011-04-23 | 2013-07-14 | 19 | 323-12-1654 |
| 3 | 8 | 2010-07-23 | 2012-09-17 | 15 | 356-45-7849 |
| 4 | 9 | 2010-06-10 | 2013-08-25 | 23 | 311-72-8734 |
| 5 | 10 | 2010-01-01 | 2013-09-30 | 25 | 543-98-3264 |

**Vendors:**

The Vendors table is essentially the same as the mechanics and customers tables, only this table is for vendors. There is one additionally attribute in this table: vendorCompany. This exists to show what company the person supplying the parts works for. As is the same with the previous 3 tables, the vendors table is a subtable of people.

**Functional Dependencies**: vid→ vendorCompany

**Create Statement**:

CREATE TABLE Vendors(

vid int REFERENCES People(pid) primary key,

vendorCompany text NOT NULL

);

**Sample Data**:

| | vid integer | vendorcompany text |
|---|---|---|
| 1 | 11 | Ford |
| 2 | 12 | Dodge |

**Parts:**

The parts table exists to show which parts the autoshop currently has in stock. Obviously, these parts are needed for repairs. This table has a foreign key of vid, because vendors are going to be in charge of these particular parts. Other attributes include prname, quantity, and costUSD. This gives the name of the part, how many of those parts are in stock, and the cost of each part. As for the relationship, vendors can have many parts, and a part can only have one vendor.

**Functional Dependencies**: prid → vid, prname, quantity, costUSD

**Create Statement:**

CREATE TABLE Parts(

prid serial primary key,

vid int REFERENCES Vendors(vid),

prname text NOT NULL,

quantity int NOT NULL,

costUSD real NOT NULL

);

**Sample Data:**

| | prid integer | vid integer | prname text | quantity integer | costusd real |
|---|---|---|---|---|---|
| 1 | 11 | 11 | Tire | 500 | 45 |
| 2 | 12 | 11 | Brakes | 200 | 100 |
| 3 | 13 | 11 | Exhaust Pipe | 157 | 60 |
| 4 | 14 | 11 | Tail Light | 1000 | 15 |
| 5 | 15 | 11 | Windshield | 125 | 350 |
| 6 | 16 | 12 | Door | 432 | 300 |
| 7 | 17 | 12 | Wipers | 549 | 20 |
| 8 | 18 | 12 | Transmission | 321 | 300 |
| 9 | 19 | 12 | Emergency Brake | 763 | 25 |

**Vehicles:**

The vehicles table is representative of the vehicles that have been to the autoshop for repairs. It contains attributes of make, model, yearMade, and yearPurchased – to get some basic information about the vehicle, and also know how long the customer has owned the vehicle. Vhid is the primary key, with cid as a foreign key.

**Functional Dependencies:** vhid→ cid, make, model, yearMade, yearPurchased.

**Create Statement:**

```
CREATE TABLE Vehicles(
vhid serial primary key,
cid int REFERENCES Customers(cid),
make text NOT NULL,
model text NOT NULL,
yearMade varchar(4),
yearPurchased varchar(4)
);
```

**Sample Data:**

| | vhid integer | cid integer | make text | model text | yearmade character varying(4) | yearpurchased character varying(4) |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | Ford | Fusion | 2007 | 2008 |
| 2 | 2 | 2 | Ford | Taurus | 2006 | 2006 |
| 3 | 3 | 4 | Dodge | Durango | 2008 | 2009 |
| 4 | 4 | 5 | Dodge | Caravan | 2003 | 2010 |
| 5 | 5 | 7 | Ford | Explorer | 2006 | 2008 |

**Repairs:**

The repairs serves to record the repairs that occur in the autoshop, on what vehicle the repair was performed on, and what mechanic worked on said repair. Other attributes include repairTitle, dateStarted, dateEnded, and repairCostUSD; which are all pretty self explanatory.

**Functional Dependencies**: rid → vhid, mid, repairTitle, dateStarted, dateEnded, repaircostUSD

**Create Statement**:

```
CREATE TABLE Repairs(
rid serial,
vhid int REFERENCES Vehicles(vhid),
mid int REFERENCES Mechanics(mid),
repairTitle text NOT NULL,
dateStarted date NOT NULL,
dateEnded date NOT NULL,
repairCostUSD double NOT NULL,
);
```

**Sample Data:**

| | rid integer | vhid integer | mid integer | repairtitle text | datestarted date | dateended date | repaircostusd real |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 3 | Tire Change | 2010-01-29 | 2010-01-29 | 90 |
| 2 | 2 | 1 | 3 | Window Replacement | 2010-03-14 | 2010-03-14 | 30 |
| 3 | 3 | 2 | 3 | Transmission Change | 2010-05-30 | 2010-06-01 | 300 |
| 4 | 4 | 1 | 6 | Wiper Replacement | 2011-06-30 | 2011-06-30 | 90 |
| 5 | 5 | 2 | 6 | Windshield Replacement | 2011-07-29 | 2011-07-29 | 350 |
| 6 | 6 | 3 | 8 | Tail Light Change | 2010-08-27 | 2010-08-27 | 15 |
| 7 | 7 | 4 | 8 | Brake Replacement | 2010-09-17 | 2010-09-18 | 100 |
| 8 | 8 | 5 | 9 | Emergency Break Replacement | 2010-10-29 | 2010-10-29 | 25 |
| 9 | 9 | 2 | 9 | Door Replacement | 2010-11-11 | 2010-11-12 | 300 |
| 10 | 10 | 3 | 10 | Exhaust Pipe Change | 2010-02-14 | 2010-02-14 | 60 |
| 11 | 11 | 4 | 10 | Windshield Replacement | 2010-01-15 | 2010-01-15 | 350 |

**RepairParts:**

RepairParts exists solely to avoid the use of a many to many relationship between repairs and parts. Therefore the table consists of an ‘rid’ and a ‘prid.’ only.

**Functional Dependencies:** There are none, both of these attributes are foreign keys.

**Create Statement:**

```
CREATE TABLE RepairParts(
rid int REFERENCES Repairs(rid) ,
prid int REFERENCES Parts(prid),
primary key(rid, prid)
);
```

**Sample Data:**

| | rid integer | prid integer |
|---|---|---|
| 1 | 1 | 11 |
| 2 | 2 | 20 |
| 3 | 3 | 18 |
| 4 | 4 | 17 |
| 5 | 5 | 15 |
| 6 | 6 | 14 |
| 7 | 7 | 12 |
| 8 | 8 | 19 |
| 9 | 9 | 16 |
| 10 | 10 | 13 |
| 11 | 11 | 15 |

# Views

The following sections demonstrates the use of views within the autoshop database. This includes the name of the view, sql statements that created the query for the view, and the output of the view.

**VIEW SQL:**

```
CREATE VIEW MechanicRepairs AS
SELECT P.fName, P.lName, R.repairTitle
FROM People AS P, Repairs AS R, Mechanics AS M
 WHERE P.pid = M.mid
AND M.mid = R.mid
order by fName asc;
```

**OUTPUT: Select * from MechanicRepairs**

| | fname text | lname text | repairtitle text |
|---|---|---|---|
| 1 | Alex | Vanderbilt | Tire Change |
| 2 | Alex | Vanderbilt | Window Replacement |
| 3 | Alex | Vanderbilt | Transmission Change |
| 4 | Andrew | Townes | Door Replacement |
| 5 | Andrew | Townes | Emergency Break Replacement |
| 6 | Joe | Shmoe | Tail Light Change |
| 7 | Joe | Shmoe | Brake Replacement |
| 8 | Matt | Jameson | Windshield Replacement |
| 9 | Matt | Jameson | Wiper Replacement |
| 10 | Tony | Boyle | Exhaust Pipe Change |
| 11 | Tony | Boyle | Windshield Replacement |

**VIEW SQL:**

```
CREATE VIEW PartsOnVehicles AS

SELECT P.prname, V.make, V.Model

FROM Parts AS P, Vehicles AS V, Repairs AS R, RepairParts AS RP

WHERE P.prid = RP.prid

AND RP.rid = R.rid

 AND R.vhid = V.vhid

Order by prname asc;
```

**OUTPUT:(Shows the parts used on which make and model)**

| | prname text | make text | model text |
|---|---|---|---|
| 1 | Brakes | Dodge | Caravan |
| 2 | Door | Ford | Taurus |
| 3 | Emergency Brak | Ford | Explorer |
| 4 | Exhaust Pipe | Dodge | Durango |
| 5 | Tail Light | Dodge | Durango |
| 6 | Tire | Ford | Fusion |
| 7 | Transmission | Ford | Taurus |
| 8 | Window | Ford | Fusion |
| 9 | Windshield | Ford | Taurus |
| 10 | Windshield | Dodge | Caravan |
| 11 | Wipers | Ford | Fusion |

# Reports and Queries

The following are random reports with their queries shown.

**REPORT SQL:**

```
SELECT P.fName, P.lName, R.repairCostUSD

FROM People AS P, Customers AS C, Repairs as R, Vehicles AS V

WHERE P.pid = C.cid

AND C.cid = V.cid

AND V.vhid = R.vhid o

rder by fName asc;
```

**OUTPUT:**

| | fname text | lname text | repaircostusd real |
|---|---|---|---|
| 1 | Connor | Loughlin | 90 |
| 2 | Connor | Loughlin | 30 |
| 3 | Connor | Loughlin | 90 |
| 4 | Frank | Moody | 25 |
| 5 | Nick | Castano | 15 |
| 6 | Nick | Castano | 60 |
| 7 | Steve | Rockafellow | 100 |
| 8 | Steve | Rockafellow | 350 |
| 9 | Tim | Rongo | 300 |
| 10 | Tim | Rongo | 300 |
| 11 | Tim | Rongo | 350 |

This report shows the money that each customer spent on the repairs that their vehicles underwent

# Stored Procedures

Stored Procedures can be used to query the database for info based on a specified parameter, and they will return the information specified by the user.

**Procedure: List all repairs done on a specified customer's vehicle.**

```
CREATE FUNCTION listRep(f_Name text, l_Name text)
RETURNS TABLE (repairTitle text) AS $$
BEGIN
RETURN QUERY SELECT R.repairTitle
 FROM Repairs AS R
WHERE R.vhid IN (   SELECT V.vhid
                        FROM Vehicles AS V
                        WHERE V.cid IN (     SELECT P.pid
                                              FROM People AS P
                                              WHERE P.fName = f_name
                                              AND P.lName = l_name));
 END;
$$ LANGUAGE PLPGSQL
```

**OUTPUT : SELECT listRep('Connor', 'Loughlin')**

| | listrep<br>text |
|---|---|
| 1 | Tire Change |
| 2 | Window Replacement |
| 3 | Wiper Replacement |

# Triggers

Triggers are used to update the database or keep the database from being updated inccorectly when a specified action takes place.

**Trigger:**

```
CREATE FUNCTION decrement_quantity()
RETURNS trigger AS $$
DECLARE
        prid STRING := “”;
BEGIN
        UPDATE Parts SET quantity


CREATE TRIGGER decrementQuantity AFTER INSERT
ON RepairParts FOR EACH ROW EXECUTE decrement_quantity (prid)
```

I cannot figure this trigger out. Essentially, this trigger is supposed to activate after a repair occurs. The quantity attribute in the parts table should decrement by 1 after every time a repair involving that part is finished.

# Security

Part of security of databases in general deals with granting permission to use certain parts of the database to certain users. This could be done in a number of ways. For maintenance of the database, a Statement might look like:

Create ROLE Maintenance

GRANT SELECT, INSERT, UPDATE

ON ALL TABLES IN SCHEMA PUBLIC

TO Maintenance

Essentially, security is what you want it to be. If you want a certain person to only be able to see certain tables, views, reports, etc. Simply make a user, and only grant either select, insert, update, or combinations of all three to that particular user.

# Implementation Notes

Implementing this database was not incredibly difficult, but there were still some complications getting certain functions to work correctly. For a long while, listRep returned the names of all repairs that were done, rather than those that were just done on that particular customer's vehicle. Other than that, the entirety of the trigger could not be figured out, but that was pretty much the only problem with implementing this database and everything that goes along with it.

# Known Problems

- Currently everything in the prid column starts at 11 instead of 1. The simplest solution to this problem would probably be to just drop that table, and then recreate it so as to undo any previous re-inserts into that table.
- Currently there is no way to prevent the ending dates of any kind(dateRetired, dateEnded, etc) from being inserted as a date EARLIER than the start date. Of course this can be fixed with a sompple check, however seeing as I did not realize this until the very end of the project until just now, this seems to be the best way to address that I understand that part of those tables should be fixed.

# Future Enhancements

Future updates to this database could include but are not limited to the following options:

- Make a function to return the customer who is in the shop most, for maybe special discount purposes (or just a good thrashing for being a terrible driver).
- Insert more attributes into the vehicle table: license plate number, color, age of the car.
- Query the sum of the money spent by each customer on repairs, not just simply the repairs and the money that each cost listed in a query.