Making a Vehicle Parts Database

CSS 475 Spring 2020

Website Interface URL: https://brandonhu.com/css475/

Brandon Hu
Applied Computing
University of Washington Bothell
Bellevue, WA USA
bhu413@uw.edu

Mahamadou Diallo
Computer Science and Software Engineering
University of Washington Bothell
Seattle, WA USA
mahamd@uw.edu

ABSTRACT

The vehicle database will allow users like vehicle mechanics or enthusiasts to query information about the type of vehicle, specifications, and parts that make up that vehicle. For example, enthusiasts or mechanics can use this database to search for information on a vehicle, things like year, make, model, parts information, availability, and quantity.

Other vehicle parts databases on the internet require a hefty fee to access and are also difficult to use because of their poor design. This is why

CCS CONCEPTS

• information systems ~ Data management systems ~ Database design and models ~ Relational Database Model

KEYWORDS

Database, Relational Model, Vehicle parts

DESIGN

1 Attributes for each table in the database

VEHICLE: Year, Make, Model, Engine, Style, Lights (headlights), Wheels

ENGINE: Oil, Fuel, Number of Cylinders, Displacement, Fuel Injectors, Spark Plugs

LIGHT: Socket, Voltage, Wattage

SPARK PLUG: Seat style, Thread Size, Drive Size

WHEEL: Rim Size, Tire

TIRE: Season, Pressure, Width, construction and diameter

All parts also have their own unique part number

1.1 Assumptions

The database will only contain data of vehicles that are from 1990 to 2020.

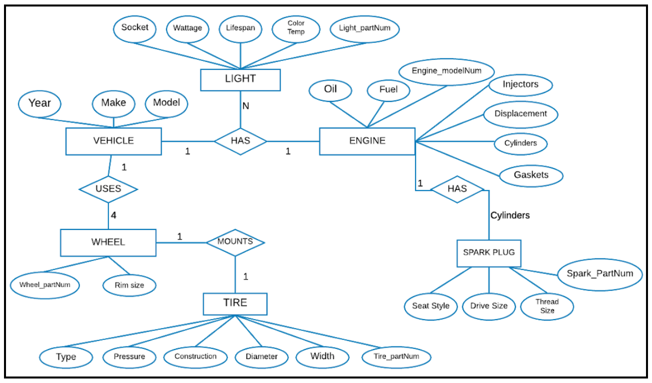
The database will include vehicle information of discontinued cars and parts up until the year it is no longer available from its manufacturer.

Similar parts used by manufacturers across different vehicle makes, models, and years will be indicated by the database.

The database will not include any electric or hybrid vehicles.

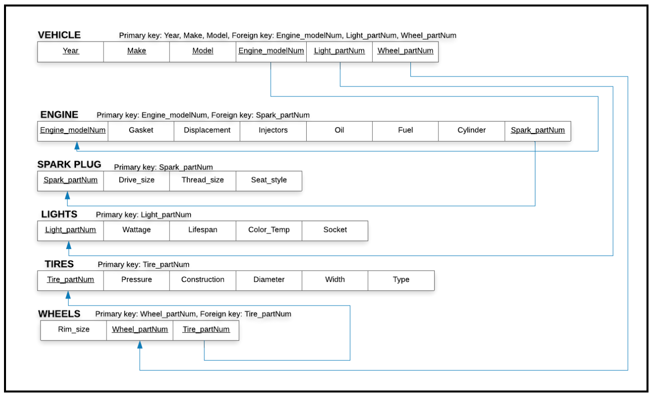
The database will not include any vehicles with rotary engines.

1.2 **Entity-Relationship (ER) Diagram**



This Entity Relationship Diagram provides an overall layout of the relationships between the entities and their attributes. The entities include the following: VEHICLE, ENGINE, SPARK PLUG, WHEEL, TIRE, and LIGHT. The relationship of the entities is clearly indicated with one another along with their respected attribute

1.4 **Relational Model**



This diagram explains the overall layout of the diagram and how each entity is related. The primary and foreign keys provide a way to specify which type of part the vehicle uses or has. The arrows highlight the references between entities.

1.5 **Constraints**

- All part and model numbers should be unique within the entity
  - These primary keys should also not be null
- In VEHICLE, Year should be between 1990 and 2020
- The foreign keys of VEHICLE must not be null since a vehicle needs all of those to run.

1.6 **Useful Queries**

- Get spark plugs compatible with a 2003 Acura TL.
- Show engines with 4 cylinders and a 2.4-liter displacement
- Get the cars made by BMW in 1995
- List cars that have a 9005 bulb before the year 2007 alphabetically
  - Useful for if they want to pull a part from an old car

2 **Schedule for Deliverables**

Week	Goals
Week 1	The team should be formed with a few project ideas in mind. Since this is the first week of the quarter, we will get everything settled first before actually starting on the project.
Week 2	There should be a set final idea with the project iteration 0 (this document) completed along with the team contract filled out. A presentation will be made to be presented on Thursday or next week depending on the class schedule.
Week 3-4	The team should have created an Entity Relationship diagram along with the relational schema model. Members should also have chosen a web hosting service (AWS or Azure) and database programming language.
Week 5-6	Successfully established a connection web server. Begin writing and populating the database
Week 7-8	Nearing completion, we expect to have completed all documents and successful website access for the database
Week 9	Final touches Complete presentation

2.1 **Division of Workload**

Since each of us want experience working with backend database creation and front end appearance, we have decided to each have a part in building this application. Right now, it seems difficult to do since we don’t know how collaboration works when it comes to coding but we are confident we can learn to communicate and work together effectively. Work will be divided 50/50.

Database Creation

3.1 DBMS

- Azure SQL Database
    - Connecting directly using Azure Data Studio
    - Cloud database allows two people to access anytime

3.2 Supported Data Types

- Integers: part numbers, model numbers
- Float
- Strings - char, varchar
- Date: manufacturer dates

3.3 Import/Export Options

- Not really applicable to our situation since our data would already be in the cloud
- Option to export as BACPAC file
    - Stores it in Azure blob storage
    - Needs an Azure storage account
    - Can be transferred back into Azure SQL Database
    - No write activity while exporting
- BACPAC file stores schema, not just data
    - Max size of 200 GB

3.4 Data Creation

- We are going to be manually entering data for vehicles. Currently, there is a Google Sheet we plan to use to enter in data first instead of directly using queries to enter. This way we can keep track of what vehicles have and have not been entered.
    - We may switch to using the only SQL, later on, to simplify things once we get the hang of it
- The Google Sheets for the data can be found using this link:
    - https://docs.google.com/spreadsheets/d/12iE4j5dq3XwZ9UA3Mk2IXnI7rC_kS7QiocZyivUnI6I/edit?usp=sharing

4 Complications

- While trying to alter Spark_partNum and have it as a VARCHAR instead of an INT, we ran into a problem because it was referenced to other tables. Drop the column wasn't working either. "ALTER TABLE ENGINE DROP CONSTRAINT FK__ENGINESpark_pa__59063A47;" is the command that worked for us.
- When trying to create the web UI, we could not figure out how to connect to the Azure database. The server we were running the website on did not work when trying to download PHP extensions that would enable us to connect. Therefore, we used the web server itself since it had a database application already on it with phpMyAdmin. Since we stored our queries for creating the tables and populating data, it was easy to just run those queries on the new server.

Discussion

5.1 Design & Results

- For the database design, all the tables were created to be related to one another. Primary keys of tables were used as foreign keys in other tables making them dependent on each other. Any changes that needed to be made means it would have to be done in a sequence and one by one or else it would not work as it conflicts with other tables.
- For the results, we went with a website switching From Azure to MySQL to connect with PHP. The website consists of HTML, CSS, and JavaScript.

5.2 Evaluation

- We spent more time and effort than anticipated in the beginning. We had a lot of issues when we first created the database to take in "INT" for part numbers not realizing there are also letters involved in the part numbers causing us to go back and change everything to "VARCHAR" which was difficult because of conflicting primary and foreign keys. If we had to do this again, we would have all the data as "VARCHAR" from the beginning.
- If we had another week, we could easily reach 500 data points.
- Our UI code is quite messy since we do not have experience with Javascript and PHP. We would try to clean it up and make things simpler and more efficient if we had more time.
    - If we were to do this again, we would use some javascript libraries to make things easier.

5.3 Feedback

- The only feedback we received on iteration 2 was that we did not have enough data in our database. We only provided one insert query as an example of what we were planning. Since then, we have inserted 23 cars with all the corresponding parts, giving us a total of about 138 data points.

5.4 **Data Generation**

- Our data was based on actual real-life data and was researched and entered manually. We were unsuccessful in creating a website scraper because the data was inconsistent, so we had to generate the data by researching each individual vehicle we wanted to add in the database and then populate manually.

5.5 **Testing**

- Testing for our database consists of website access, queries, and results. For example, using one vehicle to search for its respected spark plug.
    - Website Access: Ensure others have access to the information by entering the provided URL.
    - Queries: Entering specific queries and verifying its output and the data is correct.
    - Results: The results of each query match the inserted data.

Normalization

- Since our tables and relations are relatively simple, we believe our database is already in BCNF.
- The only table that has a composite primary key is the VEHICLE table. However, there is no table that references VEHICLE as a foreign key.
- No table has a non-functional dependency. and each column has one value

6 **SQL Query Statements**

SQL Statement	Purpose
SELECT Engine_modelNum FROM ENGINE WHERE Engine_modelNum LIKE ‘value%’ AND Oil LIKE ‘value%’ AND Fuel LIKE ‘value%’ AND Cylinder LIKE ‘value%’	The keyword LIKE and the percent signs allow us to search for attributes that start with that value. If there was no value, it would act like “any” which is particularly useful for our website if the user does not want to input every single column
SELECT * FROM ENGINE WHERE Engine_modelNum IN (SELECT Engine_modelNum FROM VEHICLE WHERE Model_year LIKE ‘value%’ AND Make LIKE ‘value%’ AND Model LIKE ‘value%’)	This query is useful for getting the engine that a certain vehicle uses. Again, the LIKE keyword and the percent signs allow the user to choose what vehicle they want to find the engine for.
SELECT * FROM SPARK_PLUG WHERE Spark_partNum IN (SELECT Spark_partNum IN ENGINE WHERE Engine_modelNum IN (SELECT Engine_modelNum IN VEHICLE WHERE Model_year LIKE ‘value%’ AND Make LIKE ‘value%’ AND Model LIKE ‘value%’))	This query finds the compatible spark plug for the specified vehicle. Since the spark plug table is not directly connected to the vehicle table, we need to first find the compatible engine which is why there are two nested queries.

ACKNOWLEDGMENTS

Professor Dr. Dong Si
Quan Trung Nghiem
James Zhang