United States Accidents Database Project 2

Yuzhi Yao, Linh Nguyen

DMS 5420

Spring 2020

April 14, 2020

1. Introduction

The dataset that is used for this application was found on kaggle. The data is car accident data in the United states (49 states) from 2016 - 2019. The dataset has over 3 million records with various fields relating to the accident (i.e. environmental conditions during the time of accident, location & location descriptions, etc). We decided to use this dataset since it met the desired specification of the project guidelines. We felt that making a dashboard app that displays U.S. accidents would be an interesting and useful application. Specifically, the dashboard would be good for visualizing trends of crashes in the United States. Individuals could make hypotheses about the data and conduct some exploratory data analysis with the dashboard to see if their hypothesis holds up.

2. Final Implementation

2.1 Use Case

Final use cases for the dashboard could be real time monitoring of accidents in a specific city or nation wide if the infrastructure was in place. This would be beneficial for helping deploy the necessary resources as an accident has occurred. Implementation of this platform as well as integration with traffic data would be useful to help route resources to that accident in the most efficient route. The platform would most benefit 2 stakeholders, first responders (EMT, Police, Firefighters) and hospitals / healthcare providers. Secondary benefits from the platform could be researchers monitoring trends and private individuals interested in monitoring accidents nationally. Researchers would be able to conduct trend analyses and possibly learn to understand why some areas are more likely to result in an accident. In addition, researchers could also begin to implement prediction algorithms to provide estimations of crashes in an area given current environmental conditions. The innovations from the dashboard could result in many lives saved due to more efficient use of resources and emergency preparedness by first responders.

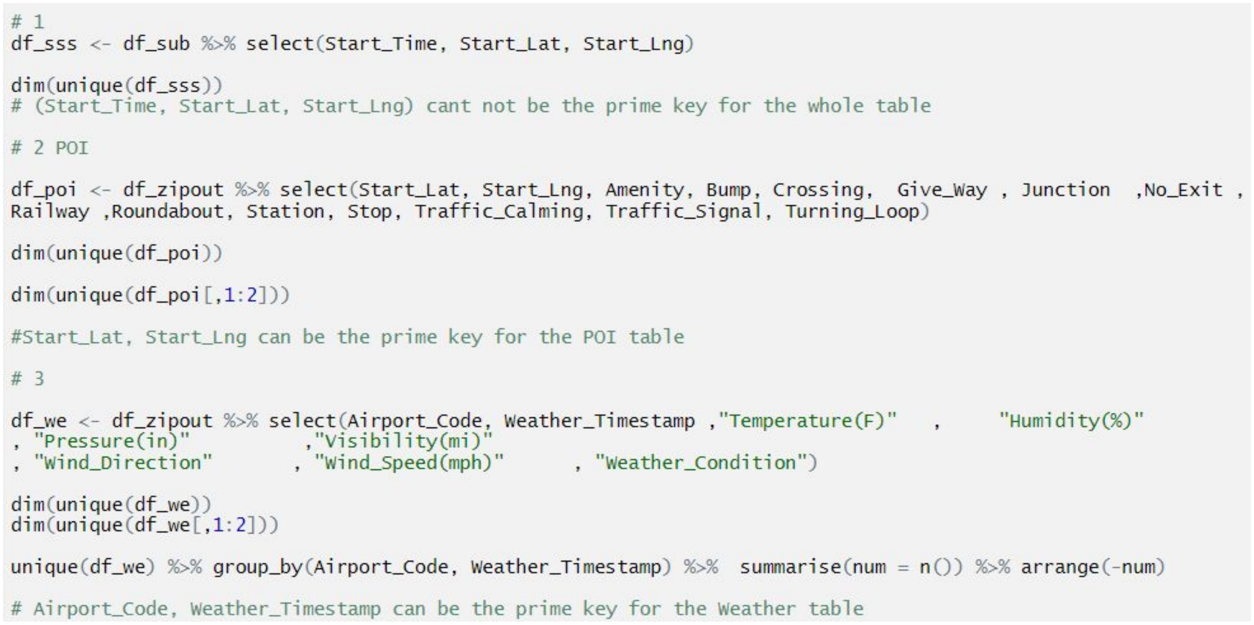
2.2 Database Design

A relational database was the best design for this data since each record of accident has interrelated data points. Decomposing the original dataset helps to improve efficiency of the database as well as provide an understanding to the relationship between data points. When creating a database for clients in a real scenario, there are business rules that the database should follow. We can think of the business rules as constraints that the database and relational tables should follow. Possible business rules that should constrain the database is that an accident should at least contain one individual and one vehicle. The accident record must have a date and location recorded.

Environmental conditions can be recorded however the data is not always collected. Environmental conditions that are recorded should also have a date recorded as well. When data is inserted, deleted, or updated an audit table should record when the change has occurred.

2.3 Normalization Process

In order to understand the relationship between the data, we created a list of functional dependencies to test them in the R language.



```r
# 1
df_sss <- df_sub %>% select(Start_Time, Start_Lat, Start_Lng)

dim(unique(df_sss))
# (Start_Time, Start_Lat, Start_Lng) cant not be the prime key for the whole table

# 2 POI

df_poi <- df_zipout %>% select(Start_Lat, Start_Lng, Amenity, Bump, Crossing,  Give_Way , Junction  ,No_Exit ,
Railway ,Roundabout, Station, Stop, Traffic_Calming, Traffic_Signal, Turning_Loop)

dim(unique(df_poi))

dim(unique(df_poi[,1:2]))

#Start_Lat, Start_Lng can be the prime key for the POI table

# 3

df_we <- df_zipout %>% select(Airport_Code, Weather_Timestamp ,"Temperature(F)"    ,     "Humidity(%)"
, "Pressure(in)"          ,"Visibility(mi)"
, "Wind_Direction"       , "Wind_Speed(mph)"      , "Weather_Condition")

dim(unique(df_we))
dim(unique(df_we[,1:2]))

unique(df_we) %>% group_by(Airport_Code, Weather_Timestamp) %>%  summarise(num = n()) %>% arrange(-num)

# Airport_Code, Weather_Timestamp can be the prime key for the Weather table
```

Figure 1. Testing code for the functional dependency in R.

Airport Code, Weather Timestamp ~ Temperature, Wind chill, Humidity, Pressure, Visibility, Wind Direction, Wind Speed, Precipitation, Weather Condition
Zipcode ~ County, City, State, Country, Timezone, airport code
Start Lat, Start Lng ~ Number, Street, City, County, Timezone, Zipcode, Airport code
Start lat, start lng ~ Amenity, Bump, Crossing, Give_Way , Junction ,No_Exit , Railway , Station, Stop, Traffic_Calming, Traffic_Signal, Turning_Loop
The initial dataset contains an ID column which uniquely identifies the accidents within the dataset. The ID is the master key for the dataset. There are two clear relationships within this dataset. First is features related to the accident and second are environmental conditions related to the accident. We decomposed 2 tables which resulted in a 3NF table for environmental conditions with the airport and weather timestamp as the composite primary key for the table. When trying to create a 3NF table for the accident related data, we further discovered that there was data related to location description and information regarding the accident specifically. In order to create a 3NF accidents table, we decomposed 2 tables, with ID as the primary key for the accidents table and remaining information in the locations table. The locations table

was composed of location descriptions of the accident as well as city, state, zip code, etc. In order to achieve 3NF, we further decomposed into a zip code table which held zip code information (city, state, county, timezone) and a location table which has location descriptions for the accident. A visual UML design can be seen below in Figure 1.

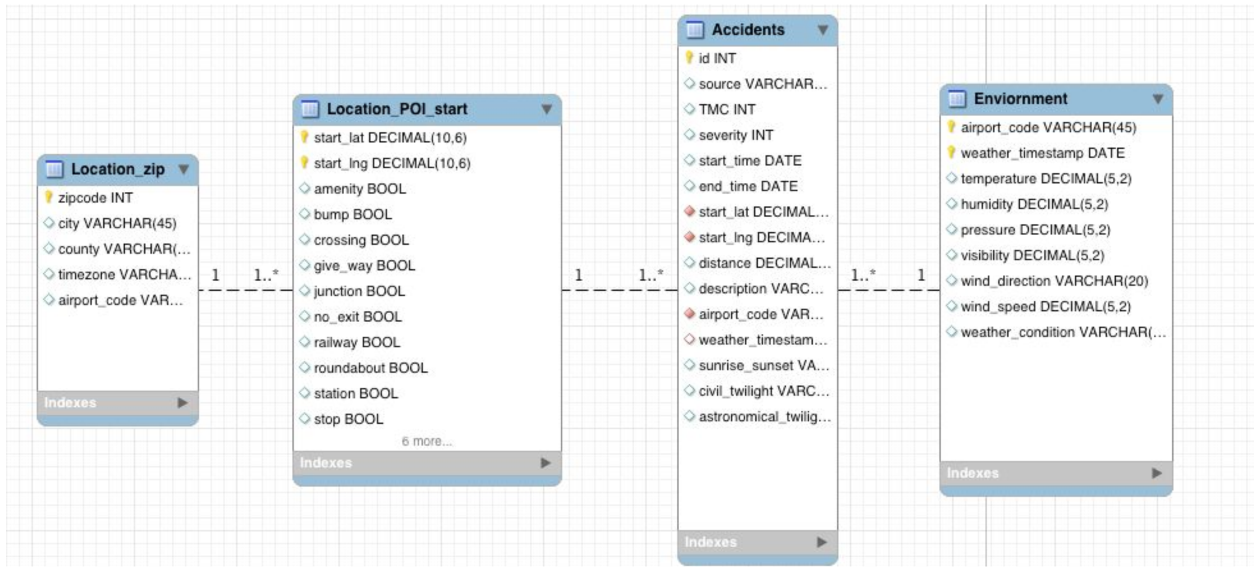


Figure 2. Final UML for Accidents Database Design

3. Illustration Of Functionality

3.1 Front-End Dashboard

The main audience we focused our project to would be first responders as well as researchers. Therefore the front end should allow easy and intuitive buttons and visualizations to allow users to quickly query the database. In order to start the front end please follow these steps:

Step 1: Download the folder which contains the us_accidents project.
Step 2: In your terminal make sure to download Streamlit (pip install streamlit). For more information, Streamlit documentation can be found here https://docs.streamlit.io/
Step 3: Start up your MAMP.
Step 4: In terminal navigate to the us_accidents folder and make sure the “accidents.py” file is there.
Step 5: In the terminal type, “streamlit run accidents.py”
Step 6: Navigate to localhost:8501
The front end should now be set up and running. You should be able to interact with many of the buttons in order to query the database and visualize the data.

3.2 Website for inserting/deleting data.

As the interaction with the database is one of the requirements for this project, we created a separate frontend that allows users with the correct permissions to insert, and delete the data. In order to get this frontend portion ready please follow the steps as follows:
Step 1: Download the folder<PHP_DBMS> and move it to the “wamp64 > www” folder.
Step 2: Visit localhost:80/PHP_DBMS/insertzip to insert data into location_zip table.
Step 3: Visit localhost:80/PHP_DBMS/getaudit to view the audit table for the location_zip table. This table will provide a running audit of when records have been inserted or deleted.
Step 4: Visit localhost:80/PHP_DBMS/delectzip for deleting data from location_zip table.
Step 5: Visit localhost:80/PHP_DBMS/getaudit to view the audit table for the location_zip table.

4. Summary & Discussion

4.1 Status of the Project

For our database project, we were able to finish the database portion (importing, cleaning, and creation of tables). The front end of the project has also been completed (one front end of users, another front end for users with insert and delete privileges). Several triggers, procedures, and views were created such as views that hide the complexity of the database, audit tables, and triggers for updating/ deleting. The whole project meets specification for the project however if we had more time and expertise in the areas of front end development we would have liked to create a more cleanly designed front end. A cleaner look for inserting and deleting of data would have been great however the functionality stands as is. The functionality clearly shows individuals are able to insert and delete data from the database as well as see a running record of which items were deleted in the audit table. If we had more time, we wanted our front end to do more visualizations and provide users the ability to pick and choose different data and graph types to create on their own. However with the front end right now, the functionality provided to users is limited.

4.2 Project Challenges

There were several challenges for this assignment. One of the main challenges we faced was working on different operating systems as well as different versions of MySQL workbench. There were a few instances where writing code to import, load, or clean the data was successful for one person however was not successful for the other person. The way we solved this was just by trial and error to see what worked for both people so we could work on 1 script instead of 2. Another challenge for us in this project was development of the front end. Since we both do not have any skills or background in this area, it was a difficult aspect of the project that we felt data scientists rarely encounter (unless they work at a startup). We created two front ends, one for users to interact with the data by visualizing it and another for users with privileges to delete and insert data. This in itself was difficult since we decided to divide and tackle these tasks independently. Another challenge with this project was the sheer size of the original dataset. The dataset was 1GB which is small in comparison to many other datasets however for us, this was one of the first time we designed and implemented database design with data with significantly larger size. After several iterations of trying to load the whole dataset, we decided to take only a sample of the dataset for our project. The main reason for this was because it would take 300+ seconds for queries to complete if we forgot to limit the number of rows to be returned. In addition, when preprocessing the data, it would also take 300+ seconds to finish executing. In addition, if we made a mistake in one of the inserting statements at initial loading of the database, either that query time limit would be reached and the query would be incomplete or it would hang for more than 10 minutes. The long execution of code significantly impacted productivity. For this reason we decided using ~ 300 mb (instead of 1 GB) of data would be reasonable to work with and meet specifications of the project. Another challenging part of the project was getting each part of the project to sync up in order to have a working product. For example, if we forgot to add a column from a table but we had already created a trigger for it, we need to remember that the trigger will fail if it was not updated with the table.

4.3 The Division Of Labor

Backend

Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
General idea	Data cleaning	Importing data	Building UML model	Creating/ Inserting Tables	Advanced features (Triggers, procedures, views)
Yuzhi/Linh	Yuzhi	Linh	Yuzhi	Yuzhi/Linh	Linh

Step 1: For the first step of the project, we needed to brainstorm ideas for who we wanted to create our platform for. In addition, we needed to search for datasets that would meet the specifications for this project. This led us to the US Accidents dataset.
Step 2: As the US Accidents dataset is relatively large (more than 1GB) and it causes our machines to become significantly slow whenever we interact with the data. Thus we decide to clean & downsize the data for this project (~300MB). Yuzhi worked on this part of the project.

Step 3: After cleaning the data, Linh was responsible for importing the data into Mysql and creating a megatable. We have many problems in this stage and sometimes we need to re-cleaning the dataset again. Data cleaning was an important task for getting data in correctly and as we expected it. With a large volume of data, there were a lot of deviations from what we expected data types to be. In addition working with null values also played a role in challenges for importing data.
Step 4: While Linh was importing the data, Yuzhi works on the UML model and normalization. As step 3 and step 4 can be processed in parallel, we decided to split this work into individual tasks and reached out to one another to solve problems we encountered.
Step 5: After the UML model is created, we will need to create the tables and insert the data from the megatable. We had many errors in this step and we debugged & worked together in order to get our data into tables correctly.
Step 6: After the database was created, we decided that one of us started to work on the frontend and the other one worked on the advanced features. Linh created advanced features such as views, triggers, and procedures.

Frontend

Step 1	Step 2 - part 1	Step 2 -part 2
General idea	Stremlit dashboard	PHP ~ Mysql
Yuzhi/Linh	Linh	Yuzhi

Step 1: As both of us did not have frontend experience before, we explored many different methods for developing the frontend with a Mysql database. Including Tableau, Javascript and Streamlit. After careful evaluation and clarification from the professor, we decided to split the frontend application into 2 parts.
Step 2 - Part 1: As our potential clients will be researchers or who are interested in the US accident data, it made no sense for them to have the privilege to delete/ update/ insert data in the database from the dashboard. Thus, part 1 of our frontend was to create an analytical live dashboard which allows the users to query/ analyze the data in real time. Linh studied the streamlit python package and built a dashboard based on that.
Step 2 - Part 2: As this project is a database class project, we are required to build a frontend that interacts with the database by inserting, deleting, and updating data. After we talked to the professor, we were allowed to finish this part separately from our analytical dashboard. We created a separate front end to allow for this functionality by using PHP. Yuzhi worked on this part and modified the framework the professor showed during the class to meet the requirements.This front end allows individuals to insert and

delete data as well as view an audit table to show when the data was inserted or deleted.