Final Project Report

CS 3265

Nov.30. 2020

Tianyu Han, Qibang Zhu

Introduction

This project is using the “US Accidents” dataset from Kaggle. The dataset was collected by Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, and Rajiv Ramnath in their paper “A Countrywide Traffic Accident Dataset.”, 2019. The accident dataset covers data from February 2016 to June 2020. There are 49 different attributes that described the accident from different angles. More than 3.5 million accidents happened during this period of time in 49 states of the USA. Car accidents are quite common in our daily lives, thousands of accidents are happening state-wide. Common as they are, large accidents can change people lives entirely. People’s physical health and property are often greatly hurt after those accidents. The more we know about the accidents, the less likely the accidents will happen to us. For this reason, we created this website and connected it with the US Accidents database. Users can use our website to search for detailed information about happened accidents. Learning from the past can help them avoid future accidents by telling them the blackspots on the road which need extra attention. Users can also contribute to our database by inserting, updating the data based on their information. They will also be able to delete the incorrect data from the database. In this way, the dataset can be more accurate and include more data about happened accidents in the US. By creating this platform, we hope that people can know more about car accidents and pay more attention on them.

Final Implementation

In our final implementation, we got our search, insert, update, delete and report functions ready. We used MySQL Workbench to write back-end SQL codes and used Visual Studio Code to write front-end PHP codes.

There are 49 columns in the dataset, which means that there are 49 attributes that we can use.
We found the following functional dependencies:

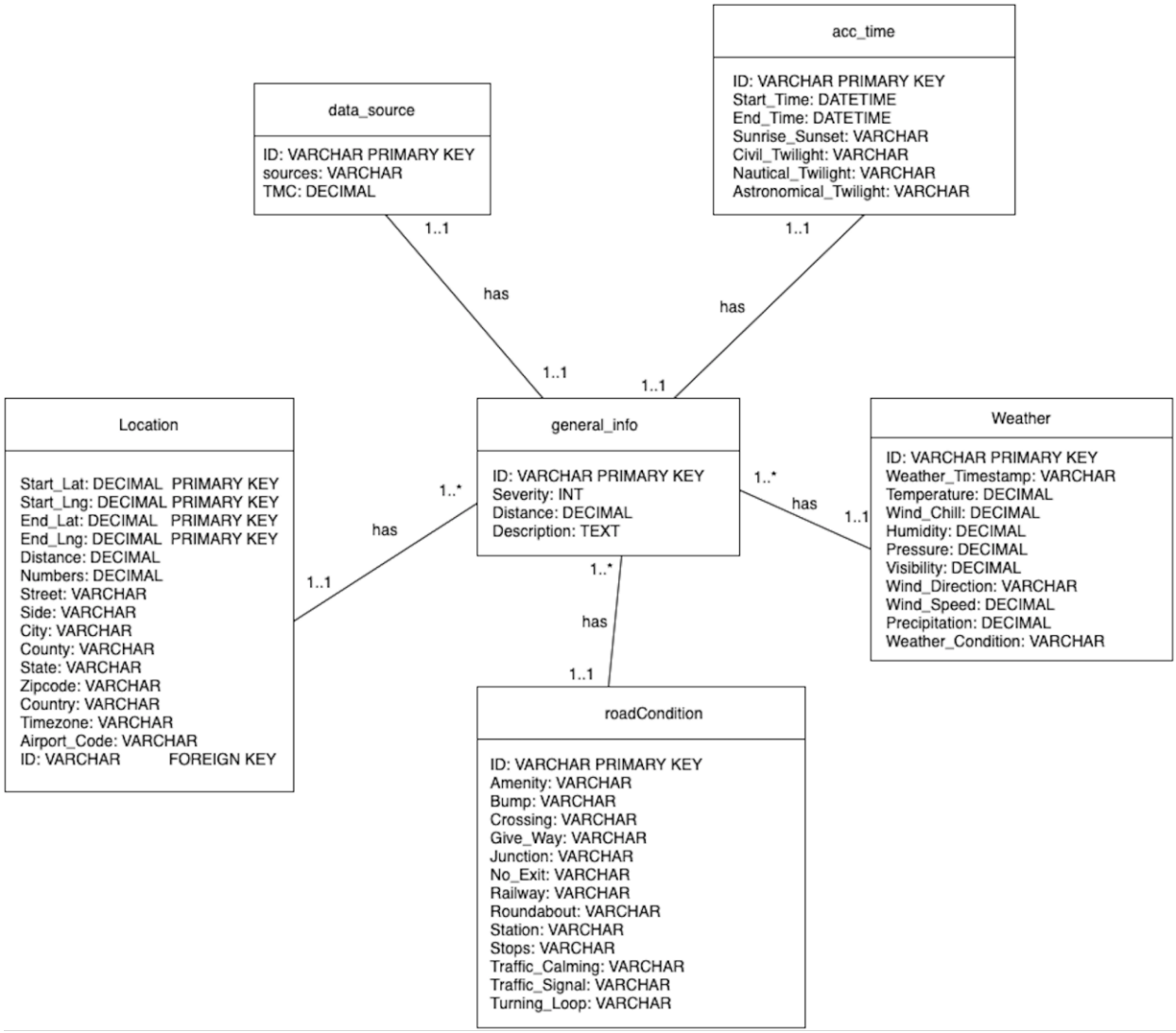
1. ID -> Sources, TMC,Severity, Start_Time, End_Time, Start_Lat, Start_Lng, End_Lat,End_Lng,Distance,Description,Numbers,Street,Side,City,County, State,Zipcode,Country,Timezone,Airport_Code,Weather_Timestamp, Temperature,Wind_Chill,Humidity,Pressure,Visibility,Wind_Direction, Wind_Speed,Precipitation,Weather_Condition,Amenity,Bump,Crossing, Give_Way,Junction,No_Exit,Railway,Roundabout,Station,Stops, Traffic_Calming,Traffic_Signal,Turning_Loop,Sunrise_Sunset,Civil_Twilight, Nautical_Twilight,Astronomical_Twilight

2. Start_Lat, Start_Lng, End_Lat, End_Lng -> Distance, Numbers, Street,Side, City, County, State,Zipcode, Country, Timezone, Airport_Code

Thus, we can divide the 49 attributes into 2 tables:

1. ID, Start_Lat, Start_Lng, End_Lat, End_Lng, Distance, Numbers, Street,Side, City, County, State,Zipcode, Country, Timezone, Airport_Code
2. ID, Sources, TMC,Severity, Start_Time, End_Time ,Description, Weather_Timestamp, Temperature,Wind_Chill, Humidity,Pressure,Visibility, Wind_Direction,Wind_Speed,Precipitation,Weather_Condition,Amenity,Bump, Crossing,Give_Way,Junction,No_Exit,Railway, Roundabout,Station,Stops, Traffic_Calming,Traffic_Signal,Turning_Loop, Sunrise_Sunset, Civil_Twilight, Nautical_Twilight,Astronomical_Twilight

There are too many attributes in the second table and all the attributes dependent on ID. We can divide them into small tables according to their categories. All of the tables are referencing ID from general_info table using foreign keys. We then got the following tables shown in the following UML:



The next step is to create small tables in MySQL Workbench and populate data into them from the megatable. Several attributes require data testing. For example, severity should be between 1 to 5. Latitude values and Longitude values should be between -180 to 180. Similar necessary data testing is done for attributes. We set the restrictions while creating tables and used test cases to test for correctness.

In our final version of website, users can use our platform when they want to search for data by City to get all accidents happened in that city or by ID to find a specific case. They can also manipulate the database by inserting, updating or even deleting the data based on the information they have about accidents. Using our summary by city and summary by state function, reports on which city or state has the most car accidents happened are also provided to the users to give them a more direct view.

Illustration of Functionality

After we have successfully created the decomposed tables and have imported data into these decomposed tables, we are now going to implement a few functionalities for our database in MySQL Workbench.

First, we have to implement triggers in order to guarantee a successful insertion even if an out of range value is entered. For example, currently, if a user tries to insert a value that is out of range, the system will return an error and the insertion will be unsuccessful. In order to reduce this kind of errors, we have implemented triggers that will check the severity value of the insertion so that it will be a positive value between 1 and 5. Any value less than 1 that is inserted will be inserted as 1, and any value greater than 5 that is inserted will be inserted as 5. Use of these triggers can not only improve user satisfaction, but can also secure the integrity of our database. Then, we go forth into developing the functionalities of our program. We decided to use PHP since Dr. Vikash has walked through a demo in class. We downloaded the demo file and decided to go from there and to add more functionalities to our database.

On the home page, we added two features that will allow user to access the report of the accident cases by city and by state. These reports included the count of the accident cases and their severity average of each individual state or city.

Home

Report by City

Submit

Results

City	Count	SeverityAverage
Houston	101240	2.2377
Los Angeles	79169	2.3718
Charlotte	78952	2.0697
Dallas	64790	2.3971
Austin	63889	2.1300
Raleigh	44675	2.1598
Atlanta	41126	2.6703
Oklahoma City	36340	2.1259
Miami	34549	2.4263
Baton Rouge	34090	2.1437
Nashville	32920	2.2439

Beside these two functionalities on the home page, we have also implemented several functionalities integrated into our navigation bar:

Home Search(ID) Search(City) Delete Update Insert

We have implemented two separate search functionalities; one is a search by ID that if successful will return only one matching result of the ID number that user has entered. The other is a search by city that will return all the accident cases that happen in the city that user has entered. We decided that it would be useful to include this search by city function because the users can sometimes have a hard time remembering the ID of the accident; therefore, a search by the name of the city would be a more convenient way to find the according accident.

Search Accident by City

City

Submit

Results

ID	City	State	Description	Severity	Start_Time
A-512396	Nashville	TN	Right lane blocked due to accident on I-40 Eastbound at Exits 215A 215B TN-155 Briley Pkwy Exits 6A 6A-B 6B.	3	2020-05-24 12:27:56
A-512400	Nashville	TN	Right lane blocked due to accident on TN-155 Briley Pkwy Westbound at Exits 18A 18B I-24 Exit 43.	2	2020-05-24 17:43:52
A-512401	Nashville	TN	Left lane blocked due to accident on I-65 Southbound at Exits 90A 90A-B 90B TN-155 Briley Pkwy Exits 15A 15B 16A 16B.	3	2020-05-24 17:53:50
A-513274	Nashville	TN	Accident on Bell Rd at Anderson Rd.	2	2020-05-25 09:57:36

Next, it is important to have DML functionalities for our database, so we integrated “insert”, “update”, and “delete” functionalities in our navigation bar as well. Having 49 different values for 1 insertion would be way too tedious, so we have designed that user only need to enter 6 key values for each insertion: ID, City, State, Description, Severity, and Start time. We will leave the other parameters as null when inserting this accident case into our database. With the update function, we are only updating

the severity level of the accident case identified by the ID number that user has entered since we thought that the other key attributes will less likely to be changed. The delete functionality that we have implemented for our database is also based on the ID number of the accident case.

Summary Discussion

The first challenge that we have met is loading the database into our decomposed tables. Since it is quite a large database with 49 attributes, any minor mistakes will cause an error in the workbench. We had to be extremely careful with the data types and the different attributes. We have encountered several errors during the process but they are all due to careless mistakes and they are not too hard to fix.
The greatest challenge for us is not the database aspects but making the frontend application. Since neither of us have prior knowledge of making frontend applications, we have met a few problems when making our database application. We have met with errors when designing the update functionality. Since the update function requires the processing of multiple input, we have met with some syntax issues while using php. However, after doing a quick search over the internet. We found the syntax error and corrected the mistake.

Division of Labor

Qibang and Tianyu together decomposed the dataset into the desired form. Then, Qibang created the mega table with all the attributes and populated the data into the decomposed table. Qibang is also responsible for creating the stored procedures, triggers, and views that will support the frontend functionality. Tianyu then would implement these frontend functionalities using php and then set up website for the database. After the website and frontend application is finished, Tianyu and Qibang met again to create the video that's uploaded onto Youtube and then we also divided up the work for the write-up report equally.