Introduction

The reason why we use this dataset is there are many data about pollutant such as their category, value, date, etc. These information can well help us extract the useful points for our goal. Our goal is to track the air quality in order to get a sense of economic activity in the country since everything is on lock down. The dataset include more than 50 attributes. Thus decomposing the dataset into many different meaningful tables are helpful when we design the database. According to the 3NF and the relationship among these attributes, we created five tables. At the same time, we drawed the UML graph to better show their relations.

Database Design

1.Explain the normalization process with functional dependencies.

We have a merged table which include each attribute in the mega_air CSV data file. The merged table is called "mega_air". The merge table is in "air". But if we contain all information in one table, it is hard to find whether the key attribute can play a decisive role in non-key attribute. Moreover, too many attributes are in a table that will make us hard to understand the database. And the merged table disobeyed BCNF principle. If we only treat Year,even_type,location and pullutant_measure_standard as key-attribute, my parameter_code can be dependent on parameter_name. Therefore I decided to transfer the data from the original table to the new table structures I created to keep data redundancy minimized and fast to be searched.

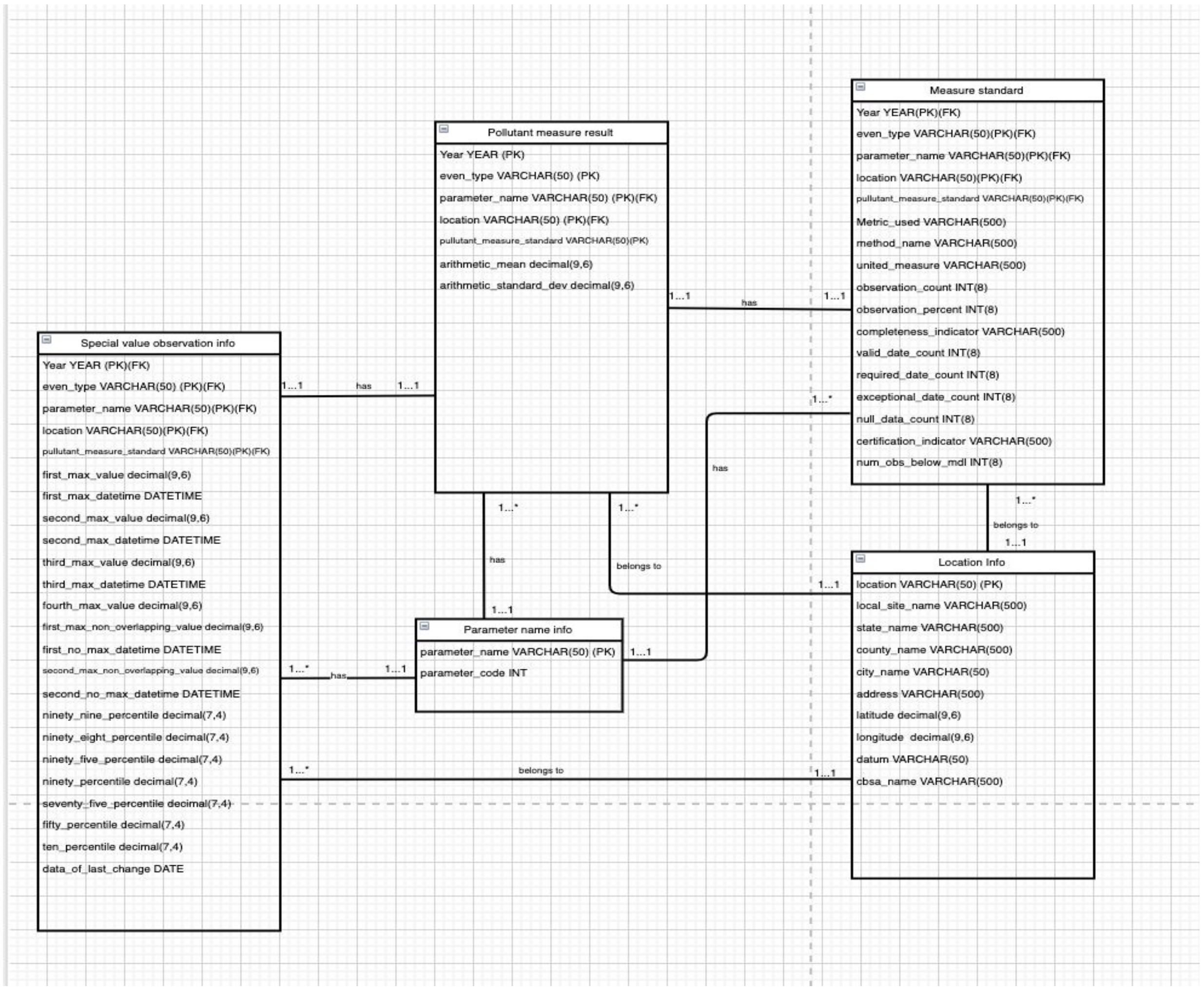
Because there exist some dependencies among the attributes just we mentioned above. We created several tables according to the decomposition we explained. First we created one big table where each record is unique. Thus the first step table belongs to 1 NF. Then we decompose the first step big table into four table. The four table are Pollutant_measure_result, Special_value_observation_info, Measure_standard and Location Info. Each table satisfies the 2 NF because all non-key column value fully and functionally dependent upon entire Primary Key. But if we want to achieve 3 NF condition, we must not let any dependency among Non-key attributes and also satisfy 2NF condition. The parameter_code can be dependent on parameter_name. Therefore, we have to select the parameter_code and parameter_name into a new table called Parameter_name _info to invoid the violation of 3 NF situation appearance. Eventually all tables satisfy 3 NF condition.

The steps above described my normalization steps taken to revise the database schema.

2.Explain the final database design.

After viewing all attributes in the dataset, we decide to divide the big table into five subject-based tables to reduce redundant data. The five table are Pollutant_measure_result, Special_value_observation_info, Measure_standard ,location Info and parameter name table.Then we turn the relative attributes into these table,specify table's primary key and set up the table relationship . For Pollutant_measure_result, we specify year, even_type,parameter_name,location and pollutant_measure_standard as the primary key. At the same time, in this table, we treat paramter_name as foreign key from paramter_name table and location as foreign key from location info table. For Special_value_observation_info, I specify year, even_type,parameter_name,location and pollutant_measure_standard as the primary key. And in this table we treat paramter_name as foreign key from paramter_name table and location as foreign key from location info table and year, even_type,parameter_name as foreign key from Pollutant_measure_result. For Measure standard table, we do the same operation on primary key and foreign key like Special_value_observation_info table. For location table, I treat location column as primary key. Eventually, for parameter name table, we specify paramter_name as primary key and also put the parameter code column into this table because they has dependency relation.

3.Include UML with cardinality.(No hand sketched UML)



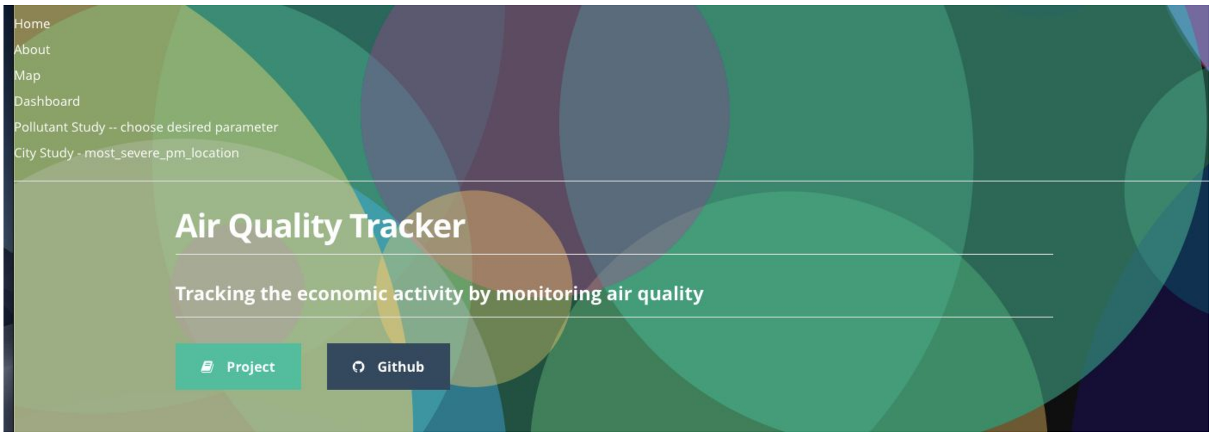
Pollutant measure result
Year YEAR (PK)
even_type VARCHAR(50) (PK)
parameter_name VARCHAR(50) (PK)(FK)
location VARCHAR(50) (PK)(FK)
pullutant_measure_standard VARCHAR(50)(PK)
arithmetic_mean decimal(9,6)
arithmetic_standard_dev decimal(9,6)
Measure standard
Year YEAR(PK)(FK)
even_type VARCHAR(50)(PK)(FK)
parameter_name VARCHAR(50)(PK)(FK)
location VARCHAR(50)(PK)(FK)
pullutant_measure_standard VARCHAR(50)(PK)(FK)
Metric_used VARCHAR(500)
method_name VARCHAR(500)
united_measure VARCHAR(500)
observation_count INT(8)
observation_percent INT(8)
completeness_indicator VARCHAR(500)
valid_date_count INT(8)
required_date_count INT(8)
exceptional_date_count INT(8)
null_data_count INT(8)
certification_indicator VARCHAR(500)
num_obs_below_mdl INT(8)
Special value observation info
Year YEAR (PK)(FK)
even_type VARCHAR(50) (PK)(FK)
parameter_name VARCHAR(50)(PK)(FK)
location VARCHAR(50)(PK)(FK)
pullutant_measure_standard VARCHAR(50)(PK)(FK)
first_max_value decimal(9,6)
first_max_datetime DATETIME
second_max_value decimal(9,6)
second_max_datetime DATETIME
third_max_value decimal(9,6)
third_max_datetime DATETIME
fourth_max_value decimal(9,6)
first_max_non_overlapping_value decimal(9,6)
first_no_max_datetime DATETIME
second_max_non_overlapping_value decimal(9,6)
second_no_max_datetime DATETIME
ninety_nine_percentile decimal(7,4)
ninety_eight_percentile decimal(7,4)
ninety_five_percentile decimal(7,4)
ninety_percentile decimal(7,4)
seventy_five_percentile decimal(7,4)
fifty_percentile decimal(7,4)
ten_percentile decimal(7,4)
data_of_last_change DATE
Parameter name info
parameter_name VARCHAR(50) (PK)
parameter_code INT
Location Info
location VARCHAR(50) (PK)
local_site_name VARCHAR(500)
state_name VARCHAR(500)
county_name VARCHAR(500)
city_name VARCHAR(50)
address VARCHAR(500)
latitude decimal(9,6)
longitude decimal(9,6)
datum VARCHAR(50)
cbsa_name VARCHAR(500)
1...1
has
1...1
1...1
has
1...1
1...*
has
1...*
belongs to
1...1
1...*
belongs to
1...1
1...*
has
1...1
1...*
has
1...1
1...*
belongs to
1...1

Description of data used for testing (how created / acquired, numbers of items entered / needed for testing)

The data source is from https://www.kaggle.com/epa/air-quality/version/1.We have 55 attributes items and five tables. Then our advanced features are view, stored procedures and triggers. Each feature have two instances for showing how we use our database to achieve some function. For view test, we choose to select all columns from the view table. We can get top 10 location has max PM 2.5 or Ozone value in five year. For stored procedures test, we should input the parameter_name as variable for stored procedures function. For example, if we input the 'Ozone' as a variable, we can get the output of each year pollutant average value. For Trigger test, we design the Trigger test for checking if people input the right longitude, latitude and year. When people input the wrong value which is out of the suitable range, the Trigger test will raise the message to let people know that they input the wrong values.

Description of final implementation

Use Cases


Home
About
Map
Dashboard
Pollutant Study -- choose desired parameter
City Study - most_severe_pm_location
Air Quality Tracker
Tracking the economic activity by monitoring air quality
Project
Github

Our Application has three main functions. First of all, users could go to dashboard page to access each table. Users could also change the layout and length of the table display, performing sorting on the table column. Since our data is so large, all tables are responsive and adaptive designed.

Secondly, we have two features for our users, pollutant study and city study. For pollutant study, user could choose their desired parameter to get the average value for certain pollutant in the past five year. Moreover, user could choose City Study to get the top 10 polluted location by PM2.5.

The third function of our application is mapping. We are tracking the air quality data across united states to get a sense of the economic activity. Since the world is under shutdown, we could use this to get a sense of economic activity.

Application Architecture

Our application use Java Spring boot to set up our back end and connect to mySql. The required dependency includes mysql, jpa, spring-web. The java application would provide api methods for our front end to call, which is implemented under react framework. Our main package in react is material-ui.

Walkthroughs

To use our applicatio, first the user has to install required dependcies : Web, JPA, MySQL, DevTools. Starting the java application, then run “npm start” the react application. One thing to be noted here if the application running on same localhost for testing, there would be access-control-allow-origin ERROR. To resolve this for testing, one could install plugins on chrome or use proxy in the react request code.

Summary Discussion

We have completed the project with good results. The challenging part is how to decompose the table appropriately and integrate the JAVA spring, React and mySql. It takes a long time to get everything in place to work. For database decomposition, the first thing is we should put the attributes we need into a big table. Since we cannot decompose the tables into 3NF immediately thus we need to decompose the database step by step. First we should make the tables satisfy 1 NF and then satisfy. Finally we should make table achieve at least 3 NF principle. Before decomposition, we also need to do some data cleaning to ensure further process success. For the backend and frontend, the first challenging part is use Java Spring to access mysql and router that as api. There are also lots of bugs when integrate that with react front end. Overall, the most challenging part is data flow. The coding is not hard but managing data across all of these applications pose a great challenging and very deep understanding of networking. We have created division of work. Mingli will be be responsible for front end, back end and also help out some sql. Qianyi will be responsible for database design and decomposition. Moreover, we all work together to finish the report and video. It is fair because the tasks are assigned according to our own strength and request. And we assume we will spend relative same time on our own tasks.