Stationary Source Air Emissions Database

Andrew Toussaint

December 5, 2023

1. Introduction

A database application was created to manage sources of air pollution in the state of Kentucky. This report gives an overview of the application's development and resulting functionality. First, the motivation for the project is described. Next, A list of technologies used in development is included and the design choices are explained. The following section provides a detailed description of the application's functionality. The report concludes with a brief discussion of limitations and potential improvements.

2. Background

The Kentucky Division for Air Quality (DAQ) is responsible for tracking and conducting inspections of stationary sources of air pollution within the commonwealth. A single source refers to an individual or organization located in a continuous space. Most of these sources are industrial production facilities. To manage sources at a granular level stationary sources are divided into "Emission Units". Each point or process that generates emissions is its own unit. This includes everything from coal-fired boilers at power plants to venting systems that control ink at print shops. Emission units are subject to emission limits based on federal and state regulations and the source is responsible for managing its own units.

Currently, DAQ does not have a database system in place to keep track of the emission units within a facility. This data is stored in permits which are pdf documents. Parsing the information about a single unit from a permit is a common task and can be time-consuming depending on the size of the permit. The goal of this project is to create a database system that stores information about sources, emission units, and emission limits in an easily accessible manner.

3. Technology

The technologies used to create this application were a MySQL database to store the data, an Apache server for hosting, and PHP for web page design and processing. jQuery was also implemented into some web pages to facilitate adding JavaScript functions that make the page responsive to user input. The development environment was comprised of Visual Studio Code and the XAMPP control panel which allowed for fast and simple testing of the application. A git repository was created to manage revisions and a remote repository was hosted on GitHub to share the project online.

4. Design

The first step in designing a database to store the desired information was to create an entity relation diagram (ER diagram) for the data. The diagram can be seen in Figure 1. The entities included were facilities, emission units, emission limits, and regulations. Each emission unit belongs to a facility and may have any number of emission limits assigned to it. Each emission limit is derived from a regulation. Each entity is also given a primary key. Facilities are defined by their agency interest number which is an arbitrary number assigned to the facility. Emission units and limits are defined by their ID. Regulations are defined by their location in the code of federal or state regulations which can be described as a citation. The various attributes that are related to each entity are also included in the diagram.

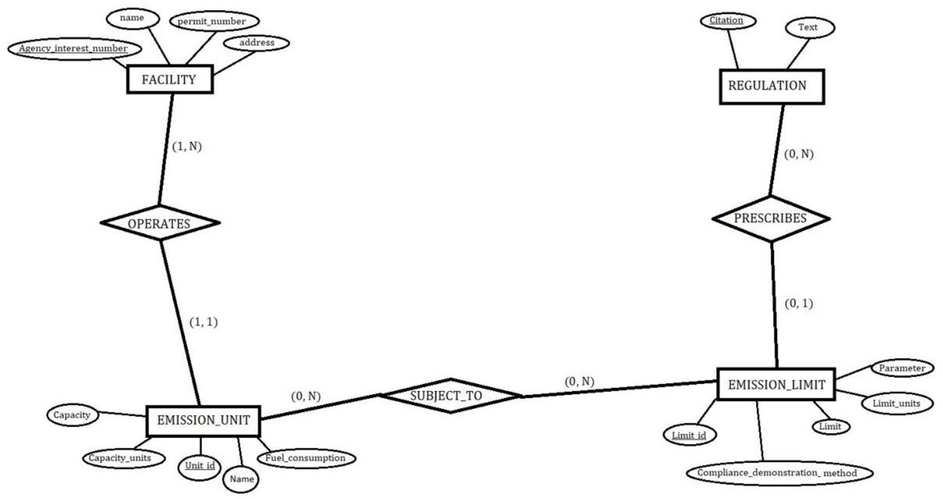

Figure 1: Entity Relation Diagram

The ER diagram is further refined into a relational schema which can be more directly related to a set of SQL relations (see figure 2). A few notable changes were made in translating the diagram to a schema. The EMISSION_UNIT relation is given a primary key of two attributes, the Unit_id and the Facility_AI_Number. This was implemented because units located at different facilities may share the same unit ID. Facility_AI_Number is also a foreign key to the Agency_interest_number attribute of the facility relation. The “Fuel_consumption” attribute was moved to a separate table with the same primary key because the Fuel_consumption attribute is NULL for most units, so providing a separate relation will reduce the occurences of NULL values within the database. The UNIT_LIMITS relation also had to be added to manage the many-to-many relationship between emission units and emission limits. This relation is an all key relation. Figures 3 and 4 show the foreign keys between relations and functional dependencies within each relation respecitvely.

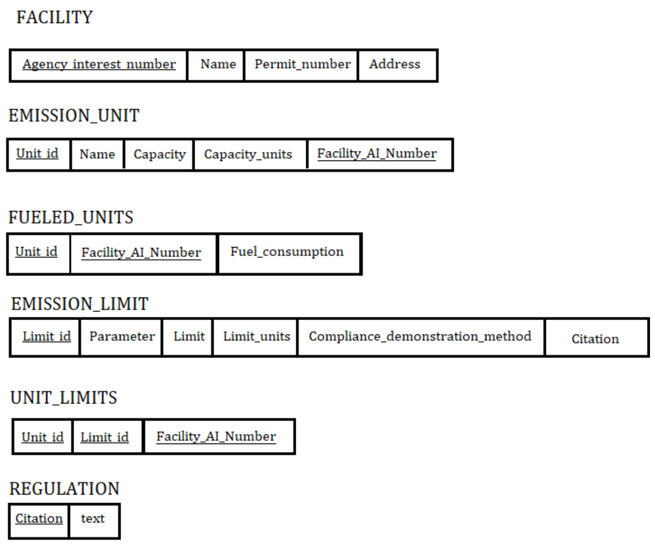

Figure 2: Relational Schema

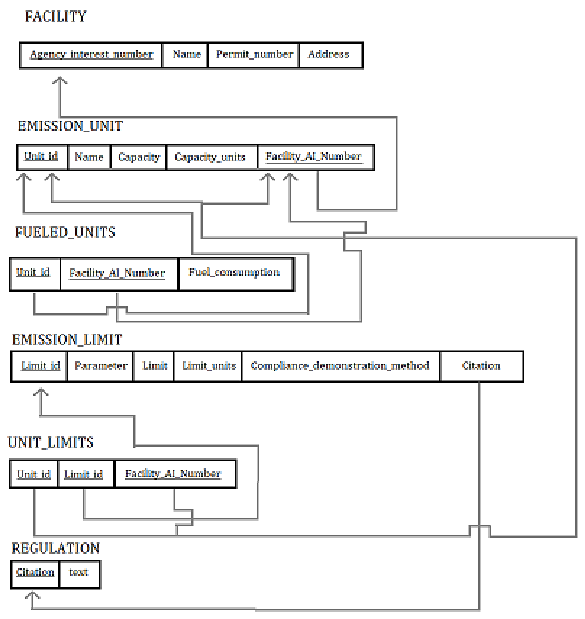

Figure 3: Foreign key relations

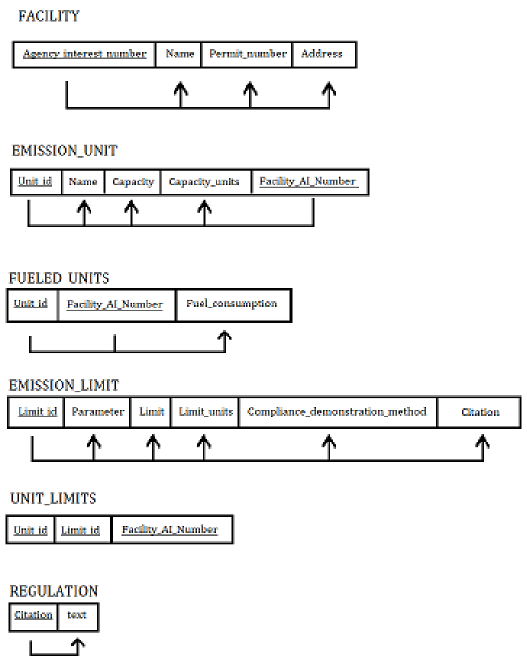

Figure 4: Functional dependencies

5. Implementation

A MySQL database was generated based on the final relational schema. Figure 5 on the next page shows a diagram of the resulting database generated by the PHPMyAdmin page. The SQL queries used to create the database were input using the phpMyAdmin page included with XAMPP. First a table was created for each relation with the attributes shown in the relational schema. Primary keys were assigned to the specified attributes. Once all the tables were complete, foreign keys were assigned between the tables. One of the challenges when developing an SQL database is selecting the correct data type for each attribute. Several of the attributers in this database were text based and used the VARCHAR datatype which requires the database administrator to specify a maximum number of characters that the attribute can hold. The number of characters for each attribute was determined by finding an example of one of the longest possible required inputs and adding a conservative buffer in case there are larger inputs required in the future. This method is impercise and could lead to problems if a larger input is required. For example, there is no set limit on the number of characters that can be included in a single section of a federal regulation. 2500 was chosen as the limit for the database because this allowed many of the larger examples available, but it is not impossible that a regulation exists which exceeds the limit.

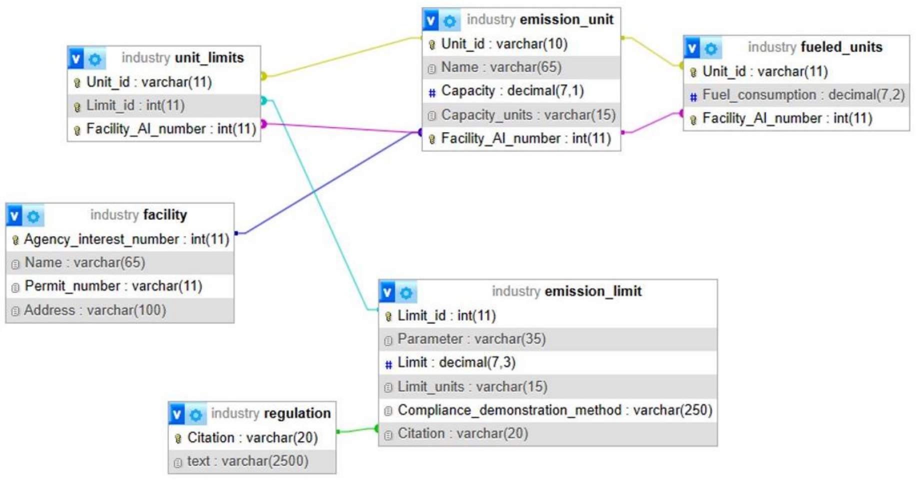

Figure 5: Database diagram

A web application was also created to facilitate and manage user access to the database. The web application consists of a series of php web pages hosted on an Apache web server. A home page was created which allows users to navigate between the other main pages. A regulations page presents users with a list of all regulations stored that allows them to create, edit, and delete regulations from the database. The Facilities page provides similar functionality for entries in the facility table. This table is also able to be ordered in ascending or descending order by either AI number or facility name. Clicking on a given facility will take a user to the "Facility/ID" route from which they can interact with the units in the facility and their emission limits. The user can create new units, edit or delete existing units, and create or add existing emission limits to these units. If a user deletes the last unit in the entire database using an emission limit, then the limit is also deleted.

To allow the user to use multiple database operations from the same page multiple forms were created. Each form has inputs that allow the user to specify the different parameters related to their operations. Each form also has a hidden input named "Action" that describes the form's purpose. When any form is submitted to the page, the first step of preprocessing is to determine which action the form has. Based on this, the preprocessor can determine which other attributes to expect from the form and structure an appropriate database query to do what the user intends. Since database updates are achieved by an http post request to the current page, this is always followed by an automatic get request to the same page. The purpose of this is to prevent resubmission of the same form when the page is reloaded.

Since user input is collected and included in SQL queries, the input must be sanitized to prevent the user from injecting extra code that could alter the database in unintended ways. This is achieved by defining the query string within single quotes and canceling in single quotes in the user input by prepending a backslash to them before including the user's input string in the query string. Similarly, sanitization is also required when including user input data in html element attributes. This is achieved by replacing single quotes with "&#39", a code that represents an equivalent character. Another limitation enforced on user input is that the user cannot attempt to create an entity if another entity with the same primary key already exists. This can be easily overridden by the user if they have developer tools within their browser, however doing so will only result in an SQL error occurring which is displayed to the user.

6. Limitations

While the application achieves the desired level of functionality, it can be further improved. Currently the web application provides no way for a user to delete a facility or to manually manage emission limits outside of the logic included in assigning them to a unit. Extra pages could be created to add this functionality. User input sanitization could be more robust in several ways. Attribute domains could be made stricter with further research into the size and total scope of what can be considered a valid input. This would also improve security. There are various situations where user input could cause a cascading delete in the database. A useful feature would be to alert the user of this and provide them with options on how to deal with the situation.

7. Conclusion

A database application was create to manage sources of air pollution using a php based web application and a MySQL database. The stationary source air emissions database allows users to organize facilities and their emission limits in a way that can be easily modified and searched for specific information. This is an improvement from the existing system of using pdf documents. The database application also has limitations and could be improved upon by parsing user inputs more safely and precisely and adding more database operations that the user can easily perform. Further development of the application may include this additional functionality as well as adding more relations to store further data on the sources.

Bibliography

Commonwealth of Kentucky. (2023). ESearch. https://dep.gateway.ky.gov/eSearch/AgencyInterest

Elmasri, R., & Navathe, S. (2016). Fundamentals of Database Systems (7th ed.). Pearson.

Open JS Foundation (2023). jQuery API. jQuery API Documentation. https://api.jquery.com/

PHP Manual. php. (2023, December 5). https://www.php.net/manual/en/

Toussaint, A. (2023, December 5) CS443G_FinalProject https://github.com/ajtoussaint/CS443G_FinalProject