

## Implementing a Web-based Transportation Data Management System

Tim Welch<sup>1</sup>, Kristin Tufte<sup>2</sup>, Ransford S. McCourt<sup>3</sup>, Robert L. Bertini<sup>4</sup> and Alan Snook<sup>3</sup>

### INTRODUCTION

Currently, most transportation organizations and firms have some amount of paper-based transportation data or their transportation data is digitally stored, but it is restricted to a limited number of people. Either way it's difficult for an individual to know exactly what data is available without a system for organizing them. Some systems do exist for accessing that data, but often times the interface is difficult to use.

We present a system that provides rapid, simple and ubiquitous access to location-based traffic and transportation data. The system provides a web-based interface allowing the viewing, insertion, and management of transportation data using only a web browser. This allows the data to be made available potentially to anyone with access to a web browser and an internet connection. Web mapping provided by Google Local allows the user to interact with a map and retrieve data spatially by simply zooming to the location they are interested in and querying the types of historical data that are available. Constraints can be placed on the queries ensuring that the results are relevant and focused. The system is designed such that the data itself can be stored anywhere as long as it is accessible by the user's web browser.

This paper presents the initial implementation of this system which supports the storage and retrieval of traffic count data for intersection and roadways. We describe the main features of the interface and how to use them. This includes querying, viewing and insertion of traffic data. Next, the design of the system is outlined including a description of the main components. The database design is also presented and the flexibility in that design for adding additional traffic data types. Finally, issues that we encountered are discussed and conclusions and future work are presented.

### USING THE SYSTEM

#### Querying Traffic Counts

Querying for traffic counts consists of specifying where to look and what to look for. The user specifies where to look using a Google Local map, which is the central part of the user interface (See Figure 1). The map controls in the top left corner of the map allow the user to pan and zoom to the location and level of detail that they want. Once a location has been selected, the user can optionally specify the type of data that they're interested in. They can specify the time of day (AM, PM), range of time (last 2 years, last 6 months, etc.), data type (intersection or roadway) and finally a specific roadway type if the roadway data type was chosen (classification count, tube count, etc). Selecting "view markers" then queries the system's traffic count records and refreshes the map with markers. Each marker represents a location that contains traffic counts

---

<sup>1</sup> Department of Computer Science, Portland State University

<sup>2</sup> Department of Civil and Environmental Engineering and Department of Computer Science, Portland State University

<sup>3</sup> DKS Associates

<sup>4</sup> Department of Civil and Environmental Engineering, Portland State University

that meets the user's criteria.

### Viewing Traffic Counts

Markers can be selected with the mouse to bring up additional information about the location including a descriptive name and the types of actions that can be taken (See the marker info "bubble" in Figure 1). Selecting "View Counts" loads the bottom frame of the window with a table containing an entry for each count that is available (See bottom of Figure 1). A table entry consists of a link to download the count, bare-minimum essential information about the count, and a link to more detailed information about the count. The bare-minimum information includes the count type, day of the week it was collected, year, start time and end time. This allows the user to glance through a potentially long list of counts and quickly see which ones they are interested in.

Following the provided link to more detailed information allows the user to view all of the count record data including who originally inserted the count, when they inserted it, and what format the file is in (See Figure 2).

The benefits of the layered access approach described above, whereby the user accesses progressively more detailed and specific information, is that they can quickly move through a potentially large dataset and only mine as deep as they need to. The user may only need to find out if a count is available, or who inserted it and not access the count itself.

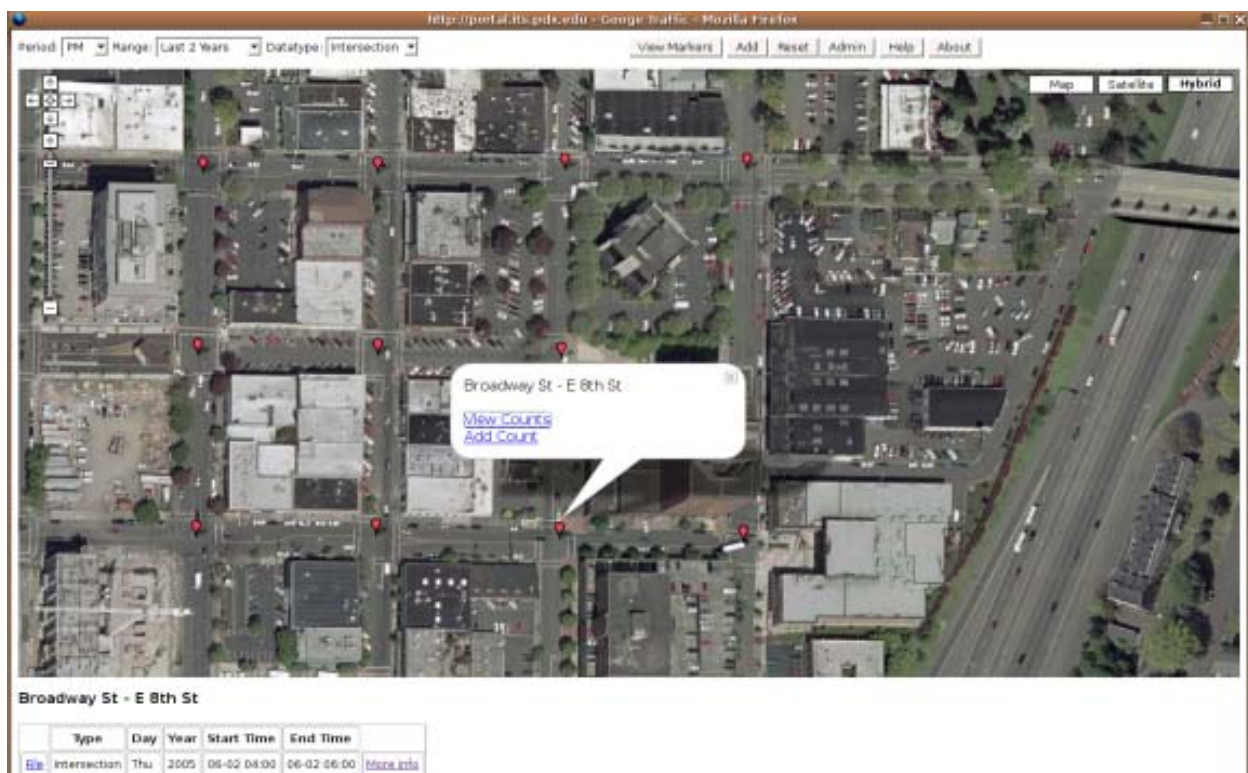


Figure 1 – Viewing All Traffic Counts Available at a Single Location.

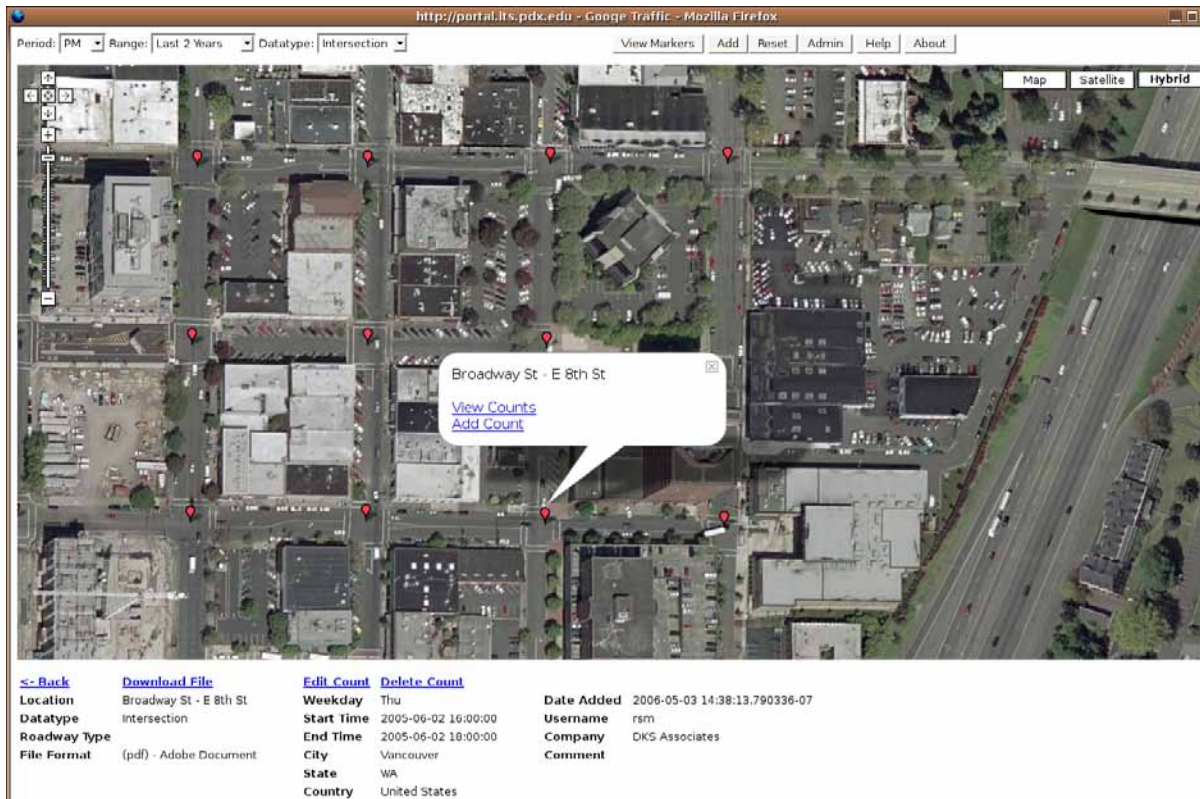


Figure 2 – Viewing Detailed Information About a Single Traffic Count

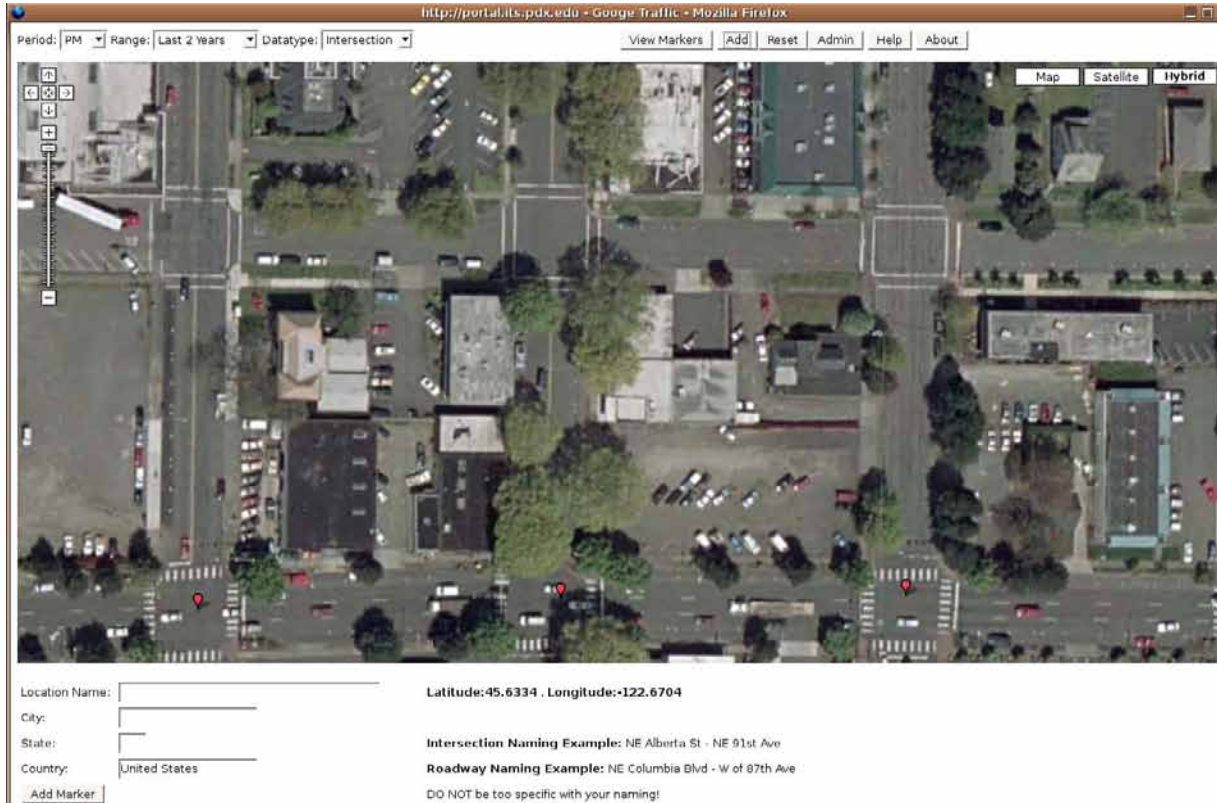


Figure 3 – Adding a Location to the System.

## Inserting Locations

Insertion of new locations into the system is done by selecting the “Add” button in the top menu and clicking a spot on the map to place the marker, as shown in Figure 3. This brings up a form in the bottom window frame allowing the user to provide information about the location and submit it. The latitude and longitude of the spot clicked is stored with the location record in the database. Once submitted the new location marker can be seen by clicking “view markers” to refresh the map.

## Inserting Traffic Counts

If the user is a moderator or administrator, then every marker info window will give them the option to add a count to that location (See Figure 4). Selecting “Add Count” loads a form in the bottom window frame and allows the user to enter all of the required parameters for a count record. First, if the count file is located on the user’s local machine and they want to upload and store it on the central file server then they browse and select the appropriate file. If the file is stored on a remote web server then the URL of the count is provided. Finally, the user enters the start time, end time and data type of the count and submits it.



**Figure 4 - Adding a Traffic Count to the System**

When a user inserts a traffic count, a check is first done for duplicate counts of the same type at the same location. If any duplicates exist then the user is queried for whether they want to continue in inserting the count. Automatic checks such as this are essential to maintaining the integrity of the data.

## **Administration and Management**

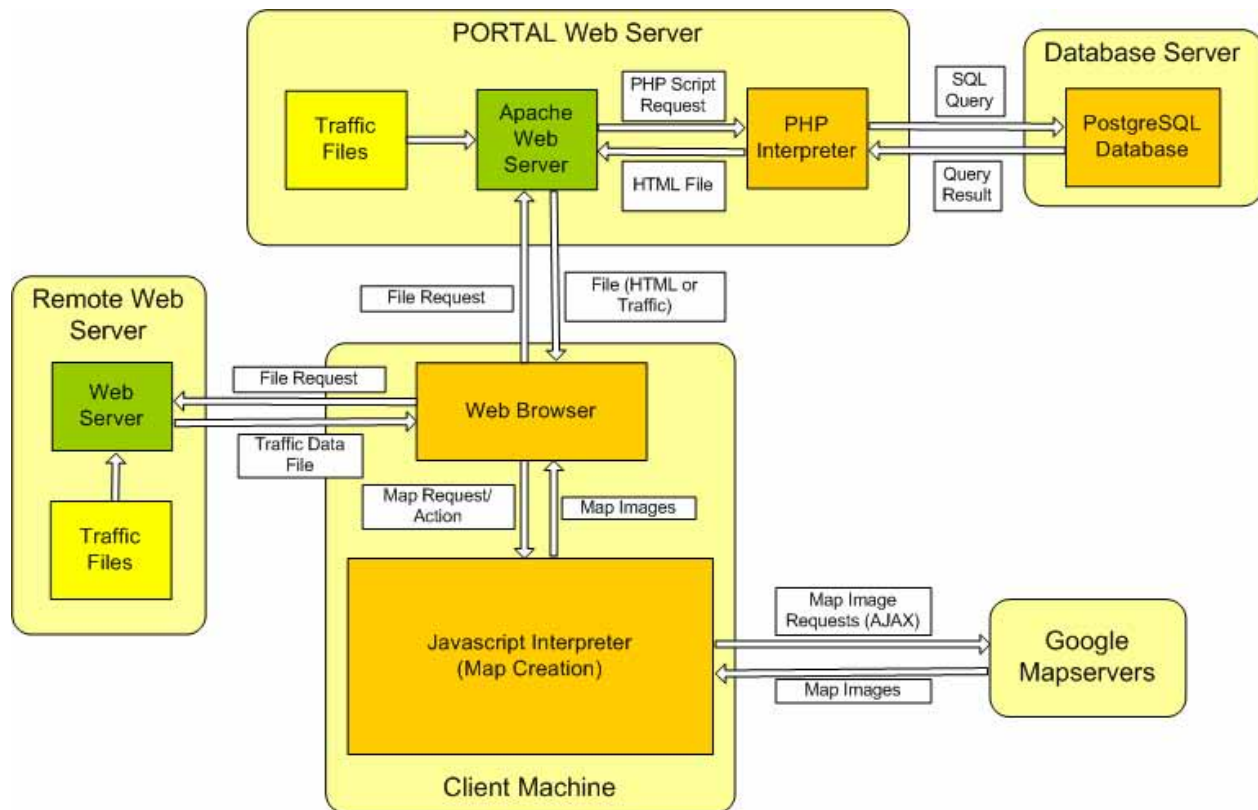
System users are separated into three basic groups: user, moderator and administrator. The “user” type allows searching and viewing of the data through the web interface as described above. Moderators can also edit or remove locations and counts, and perform basic administrative tasks such as adding/removing count datatypes and adding/removing file format descriptions directly through the web interface. Moderators are very important to maintaining the integrity and consistency of the records and data. Without them, it would be difficult to maintain a system used by multiple groups. A fully-implemented system would allow moderators to perform all of their necessary tasks using only the web interface. Finally, administrators can access everything available through the web interface and also the back-end system including the DBMS for more complex changes.

## **SYSTEM DESIGN**

The system is what's referred to as a mash-up, a hybrid web application which combines services and data from multiple sources. Traffic data records are stored in a central database server, a mapping interface is provided by Google for displaying information, and the data itself can be stored anywhere that is accessible by the users web browser. Each component is independent of the others. This means that each component can be replaced with a similar solution with minimal effort.

As Figure 5 shows, the system centers around the user's web browser. The browser makes file requests to the central web server. If the file is a PHP script then it is passed to the servers PHP interpreter where location and count queries are made if necessary to the database server and HTML content is generated. An HTML file is then returned and displayed in the user's web browser. The Google Local map is generated by the Javascript interpreter on the local machine and returned to the browser for display. Map tiles are requested as necessary from various Google mapservers using AJAX (asynchronous Javascript and XML) and returned to the user's web browser for display within the map. As the user pans around on the map the asynchronous loading of new map tiles can be seen. When a user clicks a link to download a count, a file request is made to the server that stores the count, whether it's the central web server or a remote server. The traffic data file is then returned to the user's web browser.

Overall, the system components and languages work well together. Passing data between the components is the single biggest issue, and is a challenge when working with a web application. We proceed to describe the individual components of the system and their functionality.



**Figure 5 - Architectural Diagram**

### Google Local

Google Local was chosen for the mapping component because it is easy to use, provides the necessary features for the initial system implementation, is widely used by the public, and provides a public API, or application programming interface, allowing it to be used within our web interface as long as Google's API Terms of Use are met. The map itself is simply a part of the interface to our system, it gets the user to the data and allows data to be inserted. At some point the user always moves "off the map" into another frame within the application to do further work.

### PostgreSQL DBMS

The open source relational DBMS (Database Management System) PostgreSQL is used to store location and traffic count records. These records are queried numerous times while using the application whether it's to generate markers and data tables or to insert and edit the records themselves.

### PHP / Javascript

The PHP and Javascript scripting languages were used to "glue" all of the components together. PHP was used to generate database search queries, process query results, generate HTML, process submission forms, and handle the flow of data between all of the components of the application. PHP provides a PostgreSQL API making the execution of database queries and retrieval of results extremely simple. Javascript was used to generate maps with the Google Local API. A Google Local map is represented

with a Javascript object. Each time the map is refreshed a new map object is created. Additional objects such as markers are generated dynamically using PHP and loaded into the map.

## DATABASE DESIGN

### Locations

Basic geographic information is stored for each location including a latitude and longitude as provided by Google Local when a location is inserted. The coordinate is used to place the location on the map.

### Counts

Counts are tied to locations via the LocationID attribute. Any traffic data types that might be added in the future, like traffic signal timing cards, would be linked to locations in the same way. The rest of the attributes describe the count including what it is, when it was collected, where it's located, and who provided it.

### Users

A user account and login system provides a simple way to tie data insertions to a user. The username of a person who inserts a count is made available to all users. The idea is that this ties data quality to a persons credibility.

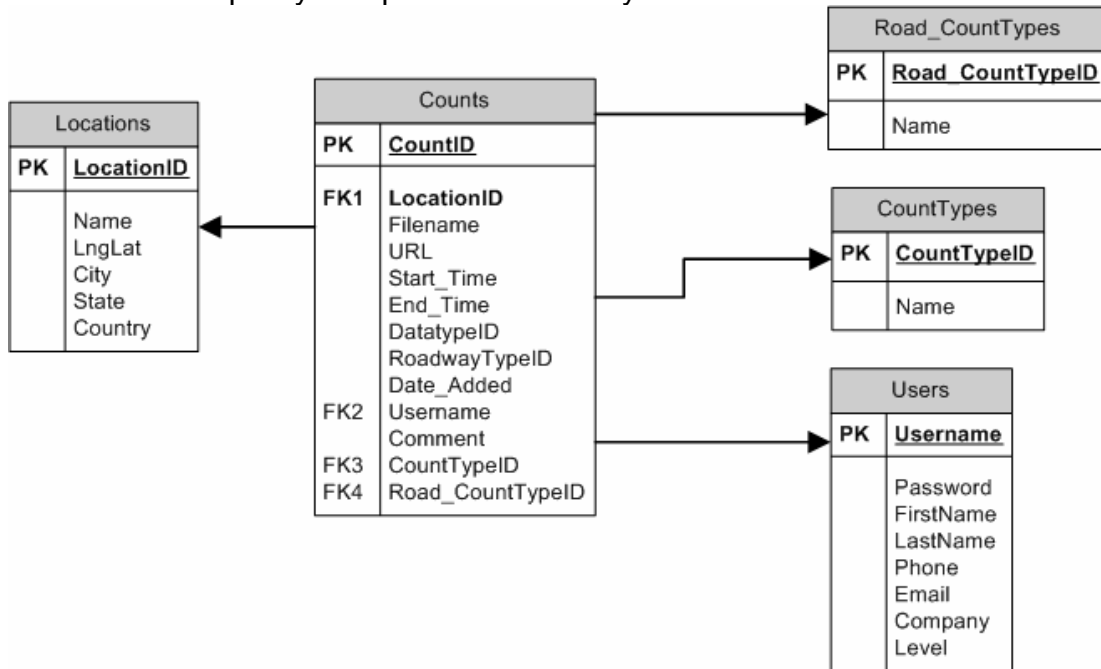


Figure 6 - Database Design

## ISSUES ENCOUNTERED

### Data Collection

Many local organizations have data and are willing to make it available to everyone within a system such as this, but the biggest roadblock is getting it in. Many traffic counts are stored in “dumb” formats, such as a PDF, where the necessary count attributes required by the system cannot be retrieved except by a person. The person-

hours necessary to prepare the data for insertion is enough that some organizations are apprehensive about doing it.

Another issue is the current method for inserting data where it is done manually by a person through a web form interface. An improvement would be to provide a web service allowing a large number of locations and/or traffic records to be inserted automatically into the system. This method would work well for data providers that already have the required information for each count, or have a way of automating the extraction of that information.

### **Data Quality**

Different organizations use different data formats. In addition, the completeness and quality of that data varies. During this project we quickly realized that if the system were to be used, certain data requirements would need to be put forth and met by anyone providing information. This would ensure some level of quality and completeness in the information provided.

### **Manageability**

If traffic data is being referenced from multiple locations, manageability does become an issue. As the size of the datasets increase, network bandwidth and computation time also become an issue.

### **CONCLUSION AND FUTURE WORK**

Overall, the system is very effective for the storage and retrieval of traffic data. The combination of a map interface and the ability to view the data in progressively greater detail allows a user to quickly query the system and access the data they need in an efficient manner. Most users with a minimal amount of knowledge are able to use the web interface right away. This is mainly attributed to peoples familiarity with navigating websites and basic interactive mapping functions such as panning and zooming.

Future work would include adding support for additional traffic data types to the system such as: traffic signal timing cards, more... Also, support could be added for the retrieval of information from other data stores. Some organizations have or will design their own systems for archiving their traffic data. Adding support for the querying and retrieval of data from outside sources would greatly increase the completeness of the pool of data. Also, this system could provide it's records to outside sources. Thus everyone's data would be shared between everyone making it truly ubiquitous. This would also allow organizations to use our interface yet control their own data. Support for basic spatial queries can be added to the system; allowing the user for example to restrict search queries using geographic boundaries.