

**PHASE III: AN OPEN SOURCE  
TOOL FOR THE VISUALIZATION,  
ANALYSIS AND REPORTING OF  
REGIONAL AND STATEWIDE  
TRANSIT NETWORKS**

**Final Report**

**PROJECT SPR-13-075**



# **PHASE III: AN OPEN SOURCE TOOL FOR THE VISUALIZATION, ANALYSIS AND REPORTING OF REGIONAL AND STATEWIDE TRANSIT NETWORKS**

## **Final Report**

**PROJECT SPR-13-075**

by

J. David Porter, David S. Kim  
Alireza Mohseni, Pouya Barahimi, Saeed Ghanbartehrani  
School of Mechanical, Industrial and Manufacturing Engineering  
Oregon State University

for

Oregon Department of Transportation  
Research Section  
555 13<sup>th</sup> Street NE, Suite 1  
Salem OR 97301

and

Federal Highway Administration  
400 Seventh Street, SW  
Washington, DC 20590-0003

**March 2016**



1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle  Phase III: An Open Source Tool for the Visualization, Analysis and Reporting of Regional and Statewide Transit Networks		5. Report Date  March 2016	
		6. Performing Organization Code	
7. Author(s)  J. David Porter, David S. Kim, Alireza Mohseni, Pouya Barahimi, Saeed Ghanbartehrani		8. Performing Organization Report No.	
9. Performing Organization Name and Address  Oregon Department of Transportation Research Section 555 13 <sup>th</sup> Street NE, Suite 1 Salem, OR 97301		10. Work Unit No. (TRAIS)	
		11. Contract or Grant No.	
12. Sponsoring Agency Name and Address  Oregon Dept. of Transportation Research Section and Federal Highway Admin. 555 13 <sup>th</sup> Street NE, Suite 1 400 Seventh Street, SW Salem, OR 97301 Washington, DC 20590-0003		13. Type of Report and Period Covered  Final Report	
		14. Sponsoring Agency Code	
15. Supplementary Notes			
16. Abstract  <p>In Phase II of the project to enhance the functionality of the Transit Network Analysis (TNA) software tool, a relational database replaced the OpenTripPlanner (OTP) graph file as the primary data repository. The main objective in Phase III was to add new features to the TNA software tool and to enhance those that were developed in the first and second phases of development.</p> <p>The most current version of the TNA software tool now contains the GTFS feeds of 64 different Oregon public and private transit agencies (the prior version included only 49 GTFS feeds), in addition to census data, park and ride data, employment data, and Title VI data. These data sources are used to populate different kinds of reports (both in visual and in tabular format) which should provide planners and analyst with a wide variety of options to visualize and evaluate the statewide public and private transit network.</p>			
17. Key Words  CENSUS DATA, GTFS, OPEN SOURCE SOFTWARE, TRANSIT NETWORK		18. Distribution Statement  Copies available from NTIS, and online at <a href="http://www.oregon.gov/ODOT/TD/TP_RES/">http://www.oregon.gov/ODOT/TD/TP_RES/</a>	
19. Security Classification (of this report)  Unclassified	20. Security Classification (of this page)  Unclassified	21. No. of Pages  XX	22. Price



**SI\* (MODERN METRIC) CONVERSION FACTORS**

APPROXIMATE CONVERSIONS TO SI UNITS					APPROXIMATE CONVERSIONS FROM SI UNITS				
Symbol	When You Know	Multiply By	To Find	Symbol	Symbol	When You Know	Multiply By	To Find	Symbol
<b><u>LENGTH</u></b>					<b><u>LENGTH</u></b>				
in	inches	25.4	millimeters	mm	mm	millimeters	0.039	inches	in
ft	feet	0.305	meters	m	m	meters	3.28	feet	ft
yd	yards	0.914	meters	m	m	meters	1.09	yards	yd
mi	miles	1.61	kilometers	km	km	kilometers	0.621	miles	mi
<b><u>AREA</u></b>					<b><u>AREA</u></b>				
in <sup>2</sup>	square inches	645.2	millimeters squared	mm <sup>2</sup>	mm <sup>2</sup>	millimeters squared	0.0016	square inches	in <sup>2</sup>
ft <sup>2</sup>	square feet	0.093	meters squared	m <sup>2</sup>	m <sup>2</sup>	meters squared	10.764	square feet	ft <sup>2</sup>
yd <sup>2</sup>	square yards	0.836	meters squared	m <sup>2</sup>	m <sup>2</sup>	meters squared	1.196	square yards	yd <sup>2</sup>
ac	acres	0.405	hectares	ha	ha	hectares	2.47	acres	ac
mi <sup>2</sup>	square miles	2.59	kilometers squared	km <sup>2</sup>	km <sup>2</sup>	kilometers squared	0.386	square miles	mi <sup>2</sup>
<b><u>VOLUME</u></b>					<b><u>VOLUME</u></b>				
fl oz	fluid ounces	29.57	milliliters	ml	ml	milliliters	0.034	fluid ounces	fl oz
gal	gallons	3.785	liters	L	L	liters	0.264	gallons	gal
ft <sup>3</sup>	cubic feet	0.028	meters cubed	m <sup>3</sup>	m <sup>3</sup>	meters cubed	35.315	cubic feet	ft <sup>3</sup>
yd <sup>3</sup>	cubic yards	0.765	meters cubed	m <sup>3</sup>	m <sup>3</sup>	meters cubed	1.308	cubic yards	yd <sup>3</sup>
NOTE: Volumes greater than 1000 L shall be shown in m <sup>3</sup> .									
<b><u>MASS</u></b>					<b><u>MASS</u></b>				
oz	ounces	28.35	grams	g	g	grams	0.035	ounces	oz
lb	pounds	0.454	kilograms	kg	kg	kilograms	2.205	pounds	lb
T	short tons (2000 lb)	0.907	megagrams	Mg	Mg	megagrams	1.102	short tons (2000 lb)	T
<b><u>TEMPERATURE (exact)</u></b>					<b><u>TEMPERATURE (exact)</u></b>				
°F	Fahrenheit	(F-32)/1.8	Celsius	°C	°C	Celsius	1.8C+32	Fahrenheit	°F

\*SI is the symbol for the International System of Measurement

## **ACKNOWLEDGEMENTS**

This research was funded by the Oregon Department of Transportation (ODOT) Research Section. We would like to thank Matthew Barnes from ODOT's Rail & Public Transit Division for providing the vision for this project. We would also like to thank Michael Bufalino and Lyn Cornell (ODOT Research) for their assistance and support of this research.

## **DISCLAIMER**

This document is disseminated under the sponsorship of the Oregon Department of Transportation and the United States Department of Transportation in the interest of information exchange. The State of Oregon and the United States Government assume no liability of its contents or use thereof.

The contents of this report reflect the view of the authors who are solely responsible for the facts and accuracy of the material presented. The contents do not necessarily reflect the official views of the Oregon Department of Transportation or the United States Department of Transportation.

The State of Oregon and the United States Government do not endorse products of manufacturers. Trademarks or manufacturers' names appear herein only because they are considered essential to the object of this document.

This report does not constitute a standard, specification, or regulation.





# TABLE OF CONTENTS

<b>1.0 INTRODUCTION .....</b>	<b>1</b>
<b>1.1 REPORT AUDIENCE AND OBJECTIVES .....</b>	<b>1</b>
<b>1.2 REPORT ORGANIZATION .....</b>	<b>2</b>
<b>2.0 SOFTWARE ARCHITECTURE AND MAIN MODULES OF THE TNA SOFTWARE TOOL .....</b>	<b>3</b>
<b>3.0 DATABASE SYSTEM .....</b>	<b>4</b>
<b>3.1 DATABASE SCHEMA.....</b>	<b>6</b>
<b>3.1.1 GTFS data tables .....</b>	<b>8</b>
<b>3.1.2 GTFS data lookup tables .....</b>	<b>9</b>
<b>3.1.3 Census data tables .....</b>	<b>9</b>
<b>3.1.4 Census data lookup tables .....</b>	<b>9</b>
<b>3.1.5 Playground tables .....</b>	<b>10</b>
<b>3.1.6 Park and ride table.....</b>	<b>10</b>
<b>3.1.7 Title VI data table.....</b>	<b>10</b>
<b>3.1.8 Employment tables .....</b>	<b>11</b>
<b>3.2 ACQUIRING/UPDATING GTFS DATA .....</b>	<b>11</b>
<b>4.0 SERVER SIDE MAIN DEVELOPMENT TASKS .....</b>	<b>12</b>
<b>4.1 LOADING GTFS DATA INTO THE POSTGRESQL RELATIONAL     DATABASE .....</b>	<b>13</b>
<b>4.2 WEB APPLICATION FOR THE TNA SOFTWARE TOOL.....</b>	<b>13</b>
<b>4.3 QUERYING AND PROCESSING SPATIAL DATA.....</b>	<b>14</b>
<b>4.4 ONEBUSAWAY-GTFS-HIBERNATE LIBRARY.....</b>	<b>14</b>
<b>4.5 MULTI-DATABASE FEATURE .....</b>	<b>15</b>
<b>4.6 METHODS FOR ON-MAP REPORTING .....</b>	<b>16</b>
<b>4.7 PLAYGROUND METHODS.....</b>	<b>16</b>
<b>4.8 ADMIN INTERFACE METHODS.....</b>	<b>17</b>
<b>5.0 CLIENT-SIDE DEVELOPMENT TASKS.....</b>	<b>18</b>
<b>5.1 THE OREGON TRANSIT AGENCIES FLOATING DIALOG BOX .....</b>	<b>20</b>
<b>5.2 LOCATION SEARCH BOX.....</b>	<b>21</b>
<b>5.3 MAP TILES .....</b>	<b>21</b>
<b>5.4 MAP-IN-MAP FEATURE.....</b>	<b>21</b>
<b>5.5 DISPLAYING STOPS AND ROUTE SHAPES .....</b>	<b>21</b>

<b>5.6</b>	<b>CLUSTERING STOPS .....</b>	<b>22</b>
<b>5.7</b>	<b>REPORTS.....</b>	<b>22</b>
<b>5.8</b>	<b>DISPLAYING GEOGRAPHICAL SHAPES ON THE MAIN MAP INTERFACE .....</b>	<b>23</b>
<b>5.9</b>	<b>LINKING TABULAR REPORTS.....</b>	<b>23</b>
<b>5.10</b>	<b>ON-MAP REPORT.....</b>	<b>24</b>
<b>5.11</b>	<b>MULTI DATABASE FUNCTIONALITY .....</b>	<b>24</b>
<b>5.12</b>	<b>PLAYGROUND INTERFACE.....</b>	<b>25</b>
<b>6.0</b>	<b>CONCLUSIONS AND FUTURE WORK .....</b>	<b>26</b>
<b>7.0</b>	<b>REFERENCES .....</b>	<b>27</b>
	<b>APPENDIX A: TABULAR REPORTS METRICS DEFINITION.....</b>	<b>29</b>

## LIST OF TABLES

Table 4.1: Server side libraries of the TNA software tool .....	13
Table 5.1: Client side libraries of the TNA software tool .....	18
Table 5.2: JavaScript libraries used in different components of the TNA software tool .....	19

## LIST OF FIGURES

Figure 2.1: Main components of the TNA software tool .....	3
Figure 3.1: Data sources and the structure of the PostgreSQL relational databases supporting the TNA software tool .....	5
Figure 3.2: Entity-Relationship (ER) diagram of the PostgreSQL relational database .....	7
Figure 5.1: The hierarchical structure of the extended OTAs floating dialog box .....	20
Figure 5.2: Reports interface of the TNA software tool .....	22
Figure 5.3: Example of a hyperlinked tag in the statewide summary report .....	24
Figure 5.4: Location of the databases button on the OTA floating dialog box .....	25

## 1.0 INTRODUCTION

This document constitutes the final report for the Oregon Department of Transportation (ODOT) research project titled “*Phase III: An Open Source Tool for the Visualization, Analysis and Reporting of Regional and Statewide Transit Networks*”, which is a continuation of the ODOT research project “*An Open Source Tool for the Visualization, Analysis, and Reporting of Regional and Statewide Transit Networks*” (1).

The main objective in phase III of the project was to significantly enhance the visualization and reporting features of the Transit Network Analysis (TNA) software tool to provide more value to planners and analysts. To fulfill this objective, the data sources utilized to produce visual and tabular reports in the TNA software tool now include not only General Transit Feed Specification (GTFS) and census data, but also employment, park & ride, and Title VI data. These data sources are archived approximately every six months and then kept as multiple relational databases accessible through the TNA software tool. By accessing different databases planners and analysts can better understand how the statewide public and private transit networks change over time.

Several visual and tabular reports were added to the TNA software tool in this phase of the project, including:

- Statewide Report
- Transit Hubs Report
- Connected Networks Report
- Connected Agencies Report (visual and tabular)
- Park & Ride Report (visual and tabular)
- Employment Reports
- Title VI Reports

In this report, the relational database model and the software tools (i.e., programming languages and libraries) that were utilized to implement the added functionality to the latest version of the TNA software tool are described.

### 1.1 REPORT AUDIENCE AND OBJECTIVES

This report is written for two main audiences. The first audience is the ODOT project sponsors with the objective of providing a comprehensive overview of the TNA software tool architecture and the functional enhancements completed in this phase. The second audience is the community of open source developers. For these developers, the goal of this report is to provide sufficient documentation so that when used in combination with documented source code, and available open source framework component documentation, the development of TNA software tool extensions is accessible.

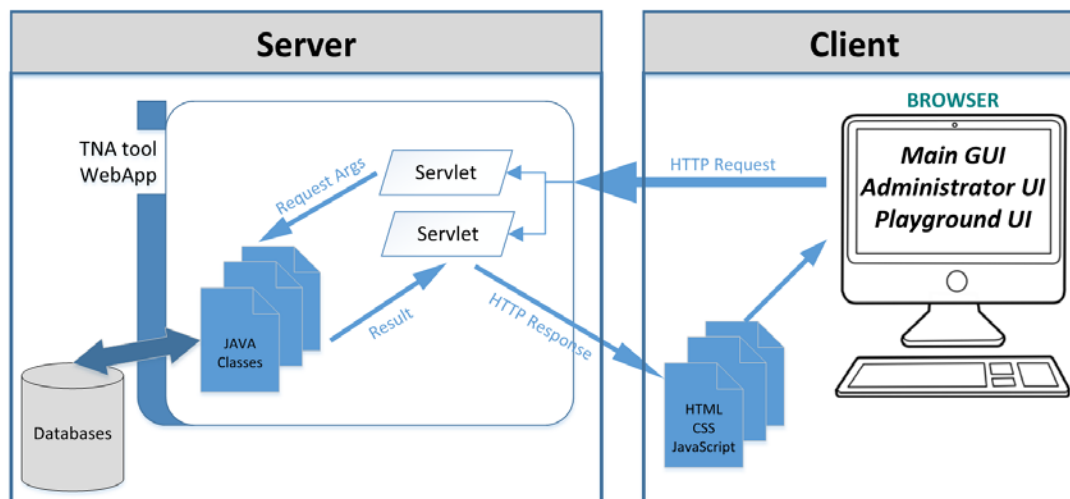
## **1.2 REPORT ORGANIZATION**

The remainder of this report describes the development work conducted to produce the latest version of the TNA software tool and is organized as follows. Chapter 2.0 describes the software architecture and the main components of the TNA software tool. Chapter 3.0 describes the database schema developed to organize the main data sources of the database management system deployed along with the GUI. In Chapter 4.0, the server-side main development tasks are discussed. Chapter 5.0 discusses the libraries and features utilized on the client-side to develop the GUI. Finally, Chapter 6.0 presents conclusions and recommendations for future work.

## 2.0 SOFTWARE ARCHITECTURE AND MAIN MODULES OF THE TNA SOFTWARE TOOL

In accordance with the Model-View-Controller (MVC) framework, the components of the TNA software tool are organized into three main groups: *Model*, *View*, and *Controller*. The MVC framework is the most commonly used design pattern when implementing user interfaces. In software architectures employed in web development, the “Model” group is typically composed of Java classes and other configuration files responsible for (1) manipulating and processing the data residing on the server, and (2) communicating with the relational database. The “View” group consists of all HTML, CSS, and JavaScript libraries used to present the graphical user interface (GUI) inside the user’s web browser. Finally, the “Controller” group is composed of the Java servlets responsible for maintaining the communication between the web browsers (i.e., the “View” group) and the Java methods (i.e., the “Model” group).

Figure 2.1 depicts the architecture of the TNA software tool. Every time a user enters a URL in the browser or submits a form, an HTTP request is sent from the browser (View) to the server. Once the HTTP request is received by the server, it is transferred to the corresponding Java servlet using a web container (e.g., Apache Tomcat 7). As a Java servlet receives the HTTP request, it extracts the input data and sends it to the appropriate Java methods (Model). The Java methods query the necessary information from the database and send back the processed data to the Java servlet. The Java servlet then puts the processed data (or object) into an HTTP response and sends it back to the browser. Once the data contained in the HTTP response is received by the browser, JavaScript and HTML scripts are used to visualize the requested data.



**Figure 2.1: Main components of the TNA software tool**

For simplicity, all the components responsible for processing the data on the server (i.e., Java servlets, Java methods, and other configuration files) are referred to as *Server Side* components throughout this report. In contrast, the components responsible for visualizing the data on the web browser are referred to as the *Client Side* components.

### 3.0 DATABASE SYSTEM

One of the most important software components of the architecture of the TNA software tool is the underlying relational database system. The relational database system provides the means to store, organize, and query many different types of data necessary to produce the many visual and tabular reports available through the TNA software tool.

The specific relational database system utilized by the TNA software tool is PostgreSQL. PostgreSQL is a powerful, open source object-relational database system with a long standing and strong industry reputation for reliability, data integrity, and correctness. PostgreSQL runs on all major operating systems, including Linux, UNIX, and Windows. PostgreSQL is fully ACID<sup>1</sup> compliant, has full support for foreign keys, joins, views, triggers, and stored procedures in multiple languages (2). For the purposes of this project, the PostgreSQL relational database system is supplemented with PostGIS, which supports spatial data types (e.g., point, line, polyline, etc.) and spatial functions and queries.

Multiple data sources are used to populate the PostgreSQL relational databases within the TNA software tool. Figure 3.1 depicts these data sources along with the process of building an up-to-date PostgreSQL relational database. Figure 3.1 also illustrates the multi-database structure of the TNA and how multiple data sources are put together to form a new database.

---

<sup>1</sup> Atomicity, consistency, isolation, durability are a set of properties that guarantee that database transactions are processed reliably.



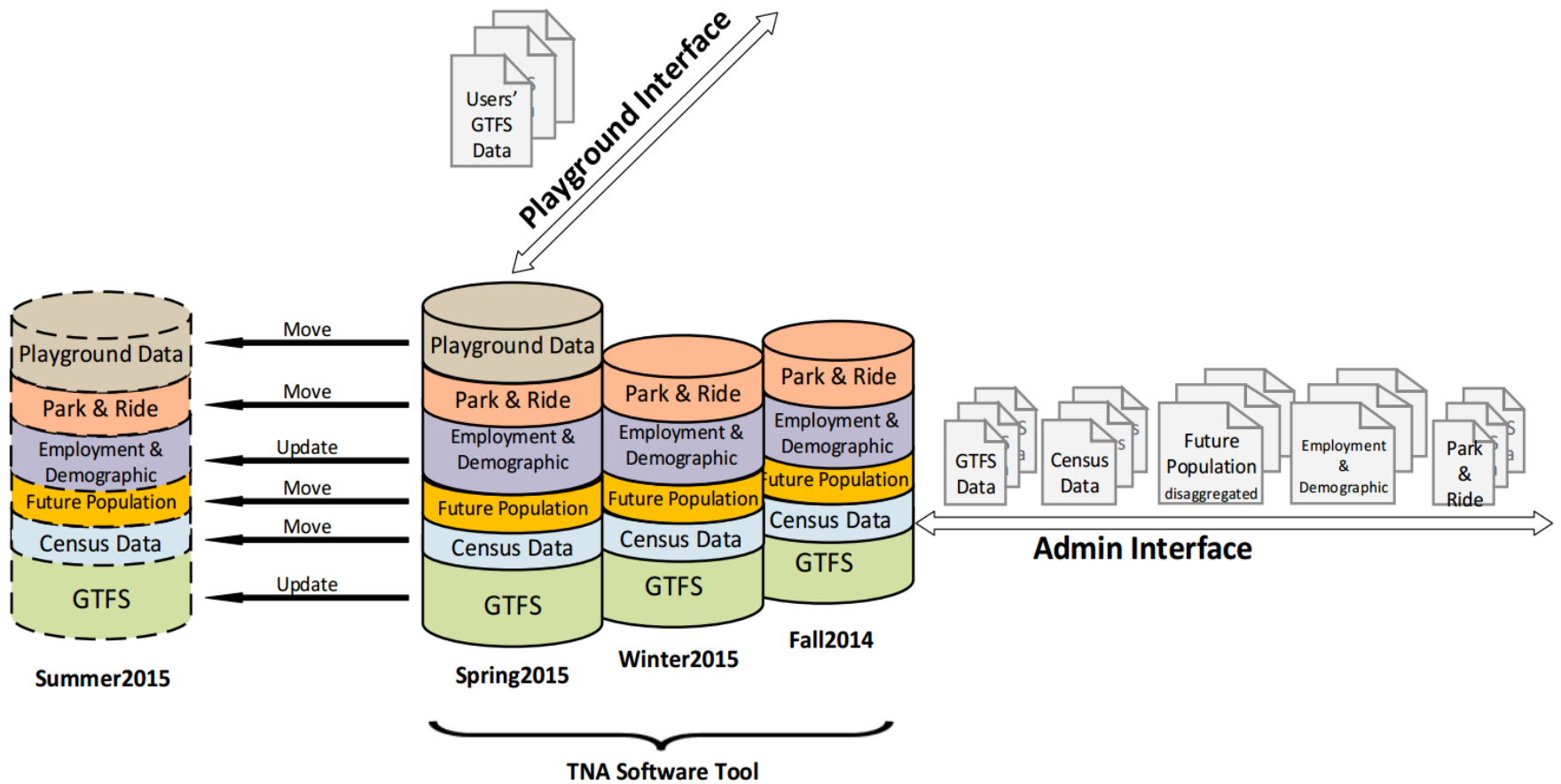


Figure 3.1: Data sources and the structure of the PostgreSQL relational databases supporting the TNA software tool

### 3.1 DATABASE SCHEMA

The Entity-Relationship (ER) diagram that represents the schema of the PostgreSQL relational database is depicted in Figure 3.2. The tables included in the schema can be classified into the following eight categories:

- **GTFS data tables.** These tables are created and populated by the module “onebusaway-gtfs-hibernate-cli”.
- **GTFS data lookup tables.** These tables are created as a result of running PostgreSQL queries on GTFS data tables. This pre-processing is used to speed up more complex queries.
- **Census data tables.** These tables contain census data and are created by importing shapefiles into the PostgreSQL relational database using the shapefile import wizard.
- **Census data lookup tables.** These tables are created as a result of running PostgreSQL queries on census data tables. This pre-processing is used to speed up more complex queries.
- **Park and Ride table:** This table is created as a result of running PostgreSQL queries on Park & Ride data provided by the Oregon Department of Transportation.
- **Employment tables.** These tables are created and populated with data provided by the Longitudinal Employer-Household Dynamics program of the US Census Bureau using a PostgreSQL query. The tables are created to enable querying on current (as well as projected) employees living/working in census blocks.
- **Title VI table.** This table is created and populated as a result of running a PostgreSQL query. The table enables spatial queries on Title VI related data.
- **Playground tables.** These tables are created as a result of running a PostgreSQL query. The data is automatically inserted/updated as new users are signed up, new feeds are added, or subset of feeds are selected by a user.

In the following sections, the tables in each of the above categories are further explored.

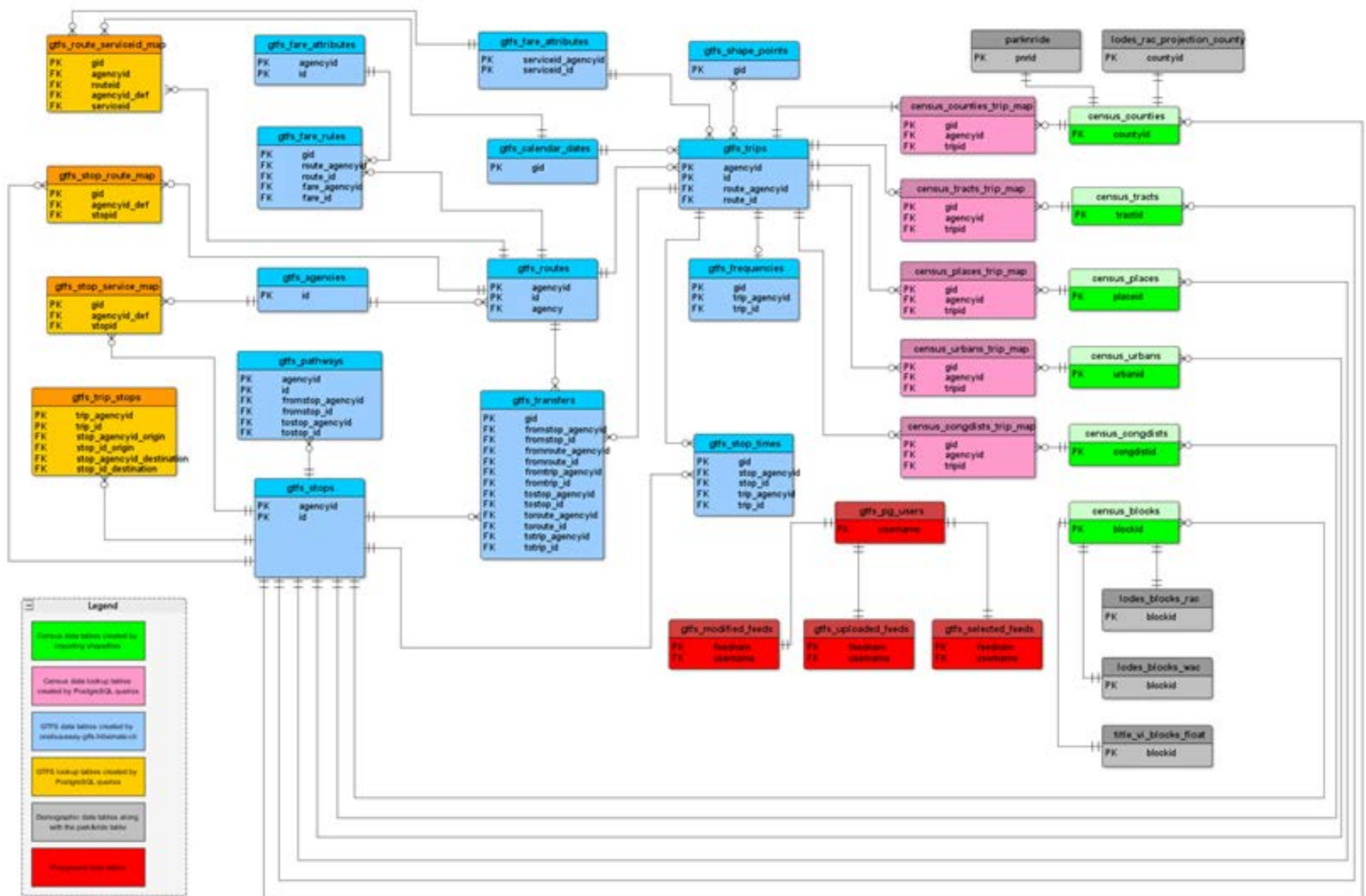


Figure 3.2: Entity-Relationship (ER) diagram of the PostgreSQL relational database

### 3.1.1 GTFS data tables

The GTFS data tables have a one-to-one correspondence to the fields of the files used by the GTFS specification. The library `onebusaway-gtfs-hibernate-cli` creates these tables by reading the GTFS feeds and loading their data into the PostgreSQL relational database. The GTFS data tables can be identified in the ER diagram depicted in Figure 3.2 by the “`gtfs_*`” prefix in their names. More information on how GTFS data are organized into files can be found on Google’s GTFS standard reference page (3). The main GTFS data tables are:

- **gtfs\_agencies.** Contains transit agency names and other information such as phone number and time zone.
- **gtfs\_stops.** Contains the list of stops served by all transit agencies in the PostgreSQL database.
- **gtfs\_routes.** Contains the list of routes served by all transit agencies in the PostgreSQL relational database.
- **gtfs\_stop\_times.** Contains the list of stop times for all trips served by all transit agencies in the PostgreSQL relational database.
- **gtfs\_trips.** Contains the list of all trips served by every route offered by all transit agencies in the PostgreSQL relational database.
- **gtfs\_frequencies.** Provides frequency information for trips that do not have stop times in the `gtfs_stop_times` table. This is a more flexible scheduling alternative offered by the GTFS standard for trips that do not have exact stop times for each stop.
- **gtfs\_transfers.** Provides transfer information between different route/trips served by every transit agency.
- **gtfs\_fare\_rules.** Contains a list of rules for computing fares for some trips served by some transit agencies.
- **gtfs\_fare\_attributes.** Contains a list of fare prices and some other information that can be used to compute the fare price for transit agencies.
- **gtfs\_pathways.** Pathways is not part of the official GTFS standard. Pathways describe connectivity among different stops and stations (4).
- **gtfs\_shape\_points.** Provides a list of trip shape points for all trips served by all transit agencies.
- **gtfs\_calendars.** Contains all the schedules used by transit agencies in the PostgreSQL database for scheduling their trips.
- **gtfs\_calendar\_dates.** Contains exceptions to service schedules in the `gtfs_calendars` table.

### 3.1.2 GTFS data lookup tables

The GTFS data lookup tables are created by running queries in the PostgreSQL relational database. The purpose of the GTFS data lookup tables is to speed up more complex GTFS data based queries. The “gtfs\_\*” prefix and “\*\_map” suffix in a table name indicate a GTFS data lookup table in the ER diagram depicted in Figure 3.2. The main GTFS data lookup tables are:

- **gtfs\_stop\_service\_map.** Used to query stops for every transit agency or transit agencies that serve each stop.
- **gtfs\_stop\_route\_map.** Used to query stops for every route or routes that serve each stop.
- **gtfs\_route\_serviceid\_map.** Used to query service ids used for a route or routes that use a specific service id.

### 3.1.3 Census data tables

Shapefiles for Oregon counties, congressional districts, census tracts, census places, urbanized areas, and census blocks were downloaded from the website of the US Census Bureau (5). The shapefile data are imported into the PostgreSQL relational database using the shapefile import wizard. This shapefile import wizard is simple to use and loads information from the shapefiles into the tables in the PostgreSQL relational database that have the same name as the input file.

The TNA software tool uses population data at the census block level to calculate population statistics around stops. The population data used by the TNA software tool corresponds to the 2010 census conducted by the US Census Bureau. The prefix “census\_\*” in a table name identifies the census data tables in the ER diagram depicted in Figure 3.2. The main census data tables are:

- **census\_blocks.** Contains all census blocks with population greater than zero in the state of Oregon.
- **census\_tracts.** Contains all census tracts within the state of Oregon.
- **census\_counties.** Contains all counties within the state of Oregon, as well as ODOT transit regions.
- **census\_places.** Contains all census places within the state of Oregon.
- **census\_urbans.** Contains all urbanized area within the state of Oregon.
- **census\_congdists.** Contains all congressional districts within the state of Oregon.

### 3.1.4 Census data lookup tables

The census data lookup tables are created by running queries in the PostgreSQL relational database to speed up more complex queries. The “census\_\*” prefix and the “\*\_map” suffix in a table name indicate a census data lookup table in the ER diagram depicted in Figure 3.2. The main census data lookup tables are:

- **census\_tracts\_trip\_map.** Used to query all trip sections within a census tract or to query census tracts that contain a specific trip.

- **census\_urbans\_trip\_map.** Used to query all trip sections within an urbanized area or to query urbanized areas that contain a specific trip.
- **census\_counties\_trip\_map.** Used to query all trip sections within a county or to query counties that contain a specific trip.
- **census\_places\_trip\_map.** Used to query all trip sections within a census place or to query the census place that contains a specific trip.
- **census\_congdistis\_trip\_map.** Used to query all trip sections within a congressional district or to query congressional districts that contain a specific trip.

### 3.1.5 Playground tables

The playground tables are created by running a query in PostgreSQL to store the data associated with individual users of the playground. These data include log in credentials; the GTFS feeds a user has added to the playground; and the GTFS feeds selected by each user to be analyzed in the playground environment. These data are automatically inserted and/or updated as the users sign up, and add/delete/select GTFS feeds. It should be noted that “admin” is defined as one of the users. The “admin” refers to the original setting of the playground tool in which all the original GTFS feeds (i.e., not the ones uploaded by the users) are selected and reported. The playground tables are:

- **gtfs\_pg\_users:** Used to keep track of valid users’ credentials, personal information, and the space allocated to users on the server for uploading GTFS feeds, and their total space used.
- **gtfs\_selected\_feeds:** Used to keep track of the GTFS feeds selected by playground users.
- **gtfs\_modified\_feeds:** Stores the requests issued by users to delete/add GTFS feeds. The data are stored in this table to define a list of computational intensive tasks to be performed by the server overnight.
- **gtfs\_uploaded\_feeds:** Used to keep track of the GTFS feeds uploaded by users, the amount of memory used, and also the availability of the GTFS feeds defined as public.

### 3.1.6 Park and ride table

The Park and Ride table is created by running a PostgreSQL query using data imported from a comma separated values (CSV) file provided by ODOT. Data in this table include the location information of the Park and Ride lots within the state of Oregon and their amenities. There is only one table specific to the Park and Ride data named **Parknride**.

### 3.1.7 Title VI data table

The Title VI data table includes data related to race, age, language, poverty level, and disability status of the population within the state of Oregon. Title VI data are available at the block group level via the American Community Survey database. Before storing Title VI data into the PostgreSQL relational database, they are disaggregated to the census block level based on the distribution of the population at the block group level (provided by the US Census Bureau and

already available in the census\_blocks table). The Title VI data at the census block level are then saved to a CSV format spreadsheet. The Title VI table is created manually and then imported from the CSV file by a query. There is only one table used to store the Title VI table named **title\_vi\_blocks\_float**.

### 3.1.8 Employment tables

The source of the employment data used in the TNA software tool is the LEHD Origin-Destination Employment Statistics provided by US Census Bureau. For each table, the data are put into a CSV file. Tables are created manually and the data are imported from the CSV file by a query. The following tables contain data needed to generate the employment reports:

- **lodes\_blocks\_rac:** Contains “Residence Area Characteristics Data” used to query on the employees who live in census blocks within the state of Oregon.
- **lodes\_blocks\_wac:** Contains “Workplace Area Characteristics Data” used to query on the employees who work in census blocks within the state of Oregon.
- **loades\_rac\_projection\_county:** Used as a basis for projecting the number of employees living in census blocks within the state of Oregon. The data are available at the county level for different years. The disaggregation of the data is performed simultaneously, by running a query as the user requests a report on projected employment.

## 3.2 ACQUIRING/UPDATING GTFS DATA

The TNA software tool is capable of switching between different PostgreSQL relational databases. As depicted in Figure 3.1, all the PostgreSQL databases share a common schema. As GTFS feeds are updated by transit agencies, they are organized by [Trillium](http://www.trillium.com) and pooled in the website <http://www.oregon-gtfs.com/>. The most up-to-date GTFS data for Oregon transit agencies are retrieved quarterly to build a new relational database. The TNA software tool has the ability to generate graphical and tabular reports based of any version of the relational database selected by user in the GUI.

In order to build an updated relational database, the latest (public and non-public) GTFS feeds are downloaded from the website <http://www.oregon-gtfs.com/> as a single zip archive. A new relational database is built through the *Admin Interface* of the TNA software tool. The zip archive is decompressed and GTFS feeds are uploaded in to the newly built database. Once the GTFS feeds are uploaded, the administrator populates all the GTFS Data Lookup Tables automatically through the same interface.

## **4.0 SERVER SIDE MAIN DEVELOPMENT TASKS**

One of the main advantages of a client-server architecture is that any resource intensive operations can be performed on the server side, enabling the best possible user experience on the client side. The TNA software tool takes maximum advantage of this benefit.

The data retrieved via queries from the PostgreSQL relational database can either be displayed directly onto the GUI of the TNA software tool or used to produce one of the many available tabular reports. These PostgreSQL queries are designed to minimize data processing on the client side.

In this section, the software tools and libraries used to add functionality to the TNA software tool are introduced. Eclipse was chosen as the integrated development environment (IDE) for developing the TNA software tool. Eclipse is a free and open source IDE which can be used to develop applications in the Java programming language, as well as other programming languages such as C, C++, JavaScript, Python, and CSS, to name a few (6). Additional plug-ins such as EGit, m2eclipse, and WTP, were installed in Eclipse to enable all the required functionality to develop the TNA software tool.

Numerous Java methods and classes were implemented to carry out data retrieval procedures from the PostgreSQL relational database, and processing of the data to create the visual or tabular reports requested by the user. A description of these methods are included in the code available on the GitHub repository (7).

The additional libraries used in the development of the server side modules are listed in Table 4.1.



**Table 4.1: Server side libraries of the TNA**

<b>Library Name</b>	<b>Library Description</b>	<b>Reference</b>
Hibernate ORM	Hibernate is an open source object-relational mapping library for Java.	(8)
Hibernate Spatial	Hibernate spatial is an open source object-relational mapping library with support for spatial databases.	(9)
onebusaway-gtfs	An open source library in Java with classes for GTFS entities.	(10)
onebusaway-gtfs-hibernate	An open source library in Java that uses hibernate to query GTFS data from relational databases.	(11)
onebusaway-gtfs-hibernate-cli	An open source library that uses hibernate for reading and loading GTFS data into relational databases.	(12)
GeoTools	An open source Java library that provides tools for processing geospatial data.	(13)
JDBC – Java Database Connection	Provides the API for accessing and processing data stored in a data source (usually a relational database) using the Java™ programming language.	(14)

## **4.1 LOADING GTFS DATA INTO THE POSTGRESQL RELATIONAL DATABASE**

GTFS data can be loaded into the PostgreSQL database of the TNA software tool by means of the open source library `onebusaway-gtfs-hibernate-cli`. The `onebusaway-gtfs-hibernate-cli` library is developed in Java and uses the library `Hibernate ORM` (8) to interface with various relational databases including MySQL, SQL Server, and PostgreSQL. More information about the `onebusaway-gtfs-hibernate-cli` library can be found on the OneBusAway developers' website (12).

In order to import GTFS data into the PostgreSQL database of the TNA software tool, the GTFS data need to be first copied to a local directory on the server. For example, the local directory `C:\Users\Administrator\Documents\Development\Feeds` was used as the default path in this project. Once new GTFS feeds are uploaded, the system administrator may select which GTFS feeds should be loaded to the PostgreSQL relational database.

## **4.2 WEB APPLICATION FOR THE TNA SOFTWARE TOOL**

A web application (WebApp) was developed for the TNA software tool. The WebApp is developed using components referred to as Java servlet, Jackson, and Jersey. The Java servlet is a Java program that allows the TNA software tool to process requests received from the client

application and generate responses. Jackson is a high performance open source JavaScript Object Notation (JSON) processor which is used on the webApp to create JSON objects. Jersey is an open source framework for developing RESTful web services in Java.

The WebApp includes modules for adding/updating GTFS data in the PostgreSQL database (`com.webapp.api.modifiers`); generating JSON objects in response to requests from the GUI (`com.webapp.api.model`); and methods for generating responses to the queries from the GUI (`com.webapp.api`). The source code for the methods of the WebApp can be found in the module “TNAtoolAPI-Webapp”.

### 4.3 QUERYING AND PROCESSING SPATIAL DATA

The WebApp of the TNA software tool must query spatial data (e.g., a list of census tracts) and issue spatial queries (e.g., identify census block internal points within a user defined shape). The main driver used for making a connection to the PostgreSQL relational database and querying spatial data is the Java Database Connection (JDBC) driver (14). Additionally, in order to make the process of connecting and querying the PostgreSQL relational database easier for executing simple queries, the library `library-hibernate-spatial` was utilized in multiple methods.

The library `library-hibernate-spatial` uses `Hibernate Spatial` and `Hibernate` to connect to the PostgreSQL relational database and relies on `GeoTools` to manipulate geographic data, including the creation of geographies such as point and polyline as well as performing projections. The library `library-hibernate-spatial` consists of several packages. The `com.library.model` package uses Plain Old Java Objects (POJO) to map to corresponding data tables in the PostgreSQL relational database. The `com.library.util` package has a class for managing connections to the PostgreSQL relational database, and the `com.library.samples` package has an `EventManager` class that queries spatial data from the PostgreSQL relational database.

As explained in the documentation for `Hibernate Spatial` and `Hibernate` (8), (9), an XML mapping file is used to map POJOs to the data tables in the PostgreSQL relational database. The XML mapping file is available via the following file.

```
library-hibernate-spatial/src/main/resources/mapping.hbm.xml
```

The line shown above indicates the location of the file within the source code of the project. Given this address, a developer can find the file regardless of how their system is configured. All queries used to retrieve spatial data from the PostgreSQL relational database through `hibernate` are also stored in the same file.

### 4.4 ONEBUSAWAY-GTFS-HIBERNATE LIBRARY

The WebApp of the TNA software tool uses the `onebusaway-gtfs-hibernate` library to query GTFS data from the PostgreSQL relational database. The library `onebusaway-gtfs-hibernate` includes a few standard queries that are used by the OneBusAway project for trip planning.

However, the TNA software tool requires many more queries to retrieve very detailed data to generate reports and on map visualizations. Therefore, the POJOs in the library `onebusaway-gtfs` had to be modified to accommodate new attributes. Also, the Hibernate object mapping file was updated to match the changes made in the database schema of the PostgreSQL relational database and the POJOs. The XML mapping file is stored in:

```
onebusaway-gtfs-  
hibernate/src/main/resources/org/onebusaway/gtfs/model/GtfsMapping.hibernate.xml
```

The queries are stored in a separate XML file:

```
onebusaway-gtfs-  
hibernate/src/main/resources/org/onebusaway/gtfs/impl/HibernateGtfsRelationalDaoI  
mpl.hibernate.xml
```

## 4.5 MULTI-DATABASE FEATURE

One salient feature of the TNA software tool is the ability to access and compare GTFS data and census data from different time periods (e.g., first quarter versus last quarter of a given year). To enable this functionality, several instances of the PostgreSQL relational database can now be stored on the server and accessed by the TNA software tool. The different instances of the PostgreSQL relational databases must be created manually by an administrator on a specific schedule (e.g., quarterly or yearly).

An instance of the PostgreSQL relational database available through the TNA software tool may contain customized GTFS and shape data. However, each individual instance of the PostgreSQL relational database must store a complete set of census data available through the US Census Bureau (e.g., 2010 census data are currently used by the TNA software tool). This is important to note because there are data tables in the PostgreSQL relational database that are created based on both transit (i.e., GTFS and shape data) and census data that have relationships with both transit and census data tables. In addition, for every newly created database, a copy of all the tables other than gtfs and census tables (i.e., employment, Title VI, Park & Ride and Playground tables) have to be copied to support the functionality of all the features/reports of the TNAST.

The TNA software tool has three libraries that access the PostgreSQL relational database (i.e., JDBC, `onebusaway-gtfs-hibernate`, and `library-hibernate-spatial`). The library `onebusaway-gtfs-hibernate` accesses GTFS data using Hibernate, which uses an XML configuration file that contains the URL, username, password, and type of each instance of the PostgreSQL relational database. An individual XML configuration file is needed for each individual instance of the PostgreSQL relational database. The Hibernate configuration files for the library `onebusaway-gtfs-hibernate` are located at:

```
onebusaway-gtfs-hibernate/src/test/resources/org/onebusaway/gtfs/examples/
```

The XML configuration files for the library `library-hibernate-spatial` are located at:

```
library-hibernate-spatial/src/main/resources
```

The WebApp uses the information stored in a Java object to find the location and the quantity of the Hibernate configuration files, as well as the name of each instance of the PostgreSQL relational database that will be displayed on the GUI of the TNA software tool. Database names are currently displayed as “Database1” and “Database2”, but they can be changed to more meaningful names. The Java object that contains all the information described above is located at:

```
onebusaway-gtfs/src/main/java/org/onebusaway/gtfs/impl/Databases.java
```

## 4.6 METHODS FOR ON-MAP REPORTING

The TNA software tool generates a variety of visual reports directly on the GUI. These GUI-based reports are referred to as *on-map* reports. On-map reports can be generated for a specific geographical area defined by a geometric shape drawn by the user (e.g., circle, rectangle, or polygon), or by clicking on single stops cluster icons (i.e., stops clusters that represent a single stop) on the main map interface of the TNA software tool. On-map reports are based on transit data (e.g., stops, routes, schedules, etc.) and on geospatial data (e.g., census tracts, census block internal points, etc.)

A method was developed in the WebApp to invoke other methods in the libraries `onebusaway-gtfs-hibernate`, `library-hibernate-spatial` and `JDBC` modules to acquire the GTFS, Census and Park & Ride data required for generating on-map reports. The results are then presented as JSON objects and sent to the GUI of the TNA software tool for display.

## 4.7 PLAYGROUND METHODS

For the registration process of the Playground interface, users have to fill out a form. Once the form is submitted, an email is sent to the administrator to verify the user. This is done using the Java package `javax.mail.internet.MimeMessage`. Once verified by the administrator, the user will be able to log into the system. User passwords are encrypted using the MD5 hashing method and then saved into the PostgreSQL relational database.

The Playground interface uses the `FileUpload` servlet to transfer the uploaded GTFS zip files to the server. Once transferred, the zip files are extracted and modified using the JAVA package `net.lingala.zip4j`. Since different users may upload the same GTFS feeds to the Playground, the fields “agency id”, “agency name”, and “feed name” of the GTFS feeds are modified based on a user’s username to avoid duplication in primary keys of some of the tables.

The GTFS feeds are then scheduled to be uploaded into the PostgreSQL relational database between 1AM and 6AM. The main reason for scheduling uploads and updates of GTFS feeds during this specific time frame is because these operations are computationally intensive and could affect the performance of the TNA software tool (it also safe to assume that the likelihood of the TNA being used during this time frame is low). Another reason is that the TNA software tool cannot manipulate the PostgreSQL relational database while it is being used or

queried by other users. The `java.util.concurrent.Executors` package is used for scheduling the Playground tasks.

Once the TNA software tool has uploaded, updated, and deleted the necessary GTFS feeds, a VACUUM process is automatically performed by the PostgreSQL relational database to ensure that all the deleted records are physically (and permanently) removed to free up storage space.

## 4.8 ADMIN INTERFACE METHODS

The Administrative interface is designed to work mostly with the servlet `DbUpdate` for adding, updating, or removing a PostgreSQL relational database from the TNA software tool. Once a user requests that a new PostgreSQL relational database is built, all the necessary connections between the TNA software tool and the PostgreSQL relational database are configured. All the information regarding these configurations is kept in a file called “`dbInfo.csv`”:

TNAtoolAPI-Webapp/WebContent/admin/dbInfo.csv
---

Once the connection between the TNA software tool and the PostgreSQL relational database is set up, all the census, employment, Title VI, and Park & Ride data are automatically copied into the newly built PostgreSQL relational database. Then, the selected GTFS feeds are imported into the PostgreSQL relational database one by one using the library `OneBusAway`.

To finalize the newly created PostgreSQL relational database, a batch file that contains SQL commands is run to update tables and create specific mappings among tables. These mappings among tables are pre-computed joined tables used for enhancing the performance of the TNA software tool (see Chapter 2.0).

## 5.0 CLIENT-SIDE DEVELOPMENT TASKS

The client side functionality of the TNA software tool was implemented using hypertext markup language (HTML) (15). Cascading Style Sheets (CSS) (16) was the tool used for styling the web pages and JavaScript (17) was the main scripting language used in the GUI. Table 5.1 provides a brief description of the additional JavaScript plug-ins and libraries that were used to add functionality to the client side of the TNA software tool.

**Table 5.1: Client side libraries of the TNA software tool**

Name	Description	Reference
jQuery	jQuery is a feature rich JavaScript library that is compatible with most web browsers.	(18)
jQuery UI	jQuery UI is a plug-in for jQuery that provides some effects, widgets and interactions for jQuery.	(19)
jQuery UI DataTables	A JavaScript library used for displaying reports in tables with features like searching, sorting, and exporting.	(20)
jQuery UI dialogextend	An open source JavaScript library that extends features of the standard jQuery UI dialog box.	(21)
Bootstrap Dropdown	An open source JavaScript library for generating dropdown menus.	(22)
jstree	An open source free JavaScript plug-in that provides a tree menu structure.	(23)
Leaflet	Leaflet is a versatile and widely used JavaScript library, which is open source and free. It is used for building interactive maps.	(24)
Leaflet markercluster	Leaflet markercluster is a JavaScript library that provides clustering capability for leaflet markers. Markercluster can display up to 50,000 markers on map efficiently.	(25)
Leaflet.encoded	An open source leaflet library used to decode the encoded polyline into an array of L.LatLng objects that can be displayed on a map.	(26)
Leaflet MiniMap	An open source leaflet library that adds mini map feature to LeafLet maps.	(27)
Google Maps API	A free library that allows using Google maps features such as map tiles and street view.	(28)
OpenStreetMap API	A free library for using OpenStreetMap features like location search.	(29)
jQuery Datepicker	A javascript library for selecting multiple dates as input.	(30)
Slidebars	A jQuery plugin for implementing app style off-canvas menus and sidebars into website.	(31)
jQuery File Upload	File Upload widget with multiple file selection, drag&drop support, progress bars, validation.	(32)
tile.stamen.js	Stamen is a free map-display JavaScript library for displaying OpenStreetMap data.	(33)

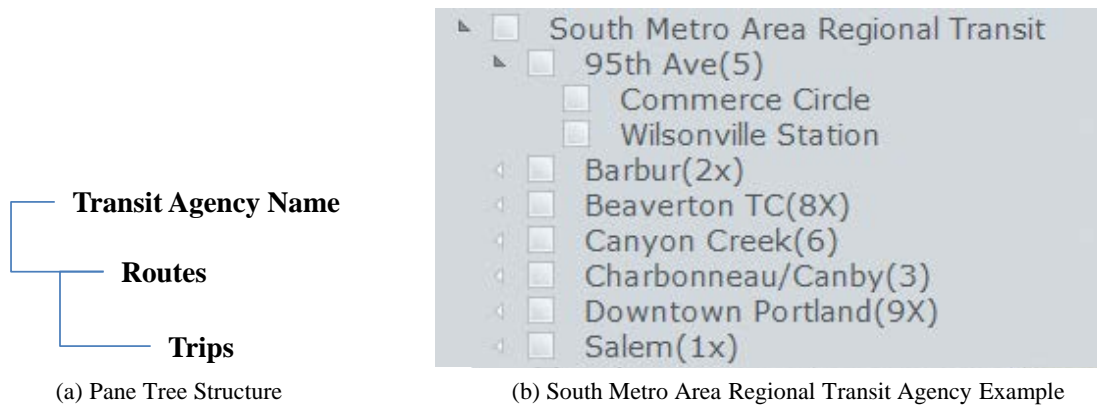
Individual components of the TNA software tool may require a different set of inputs, or different interactions with the user via the GUI to display results. Based on these requirement, each component of the TNA software tool was implemented using corresponding JavaScript libraries. Table 5.2 shows a mapping of which JavaScript libraries that are used to implement the different components or functionality of the TNA software tool.

**Table 5.2: JavaScript libraries used in different components of the TNA software tool**

<b>Name</b>	<b>Main Interface (On-map reports)</b>	<b>Tabular Reports</b>	<b>Tabular Reports (Employment &amp; Title VI reports)</b>	<b>Admin Interface</b>	<b>Playground</b>
jQuery	✓	✓	✓	✓	✓
jQuery UI	✓	✓	✓	✓	✓
jQuery UI DataTables	✓	✓	✓		
jQuery UI dialogextend	✓				
Bootstrap Dropdown	✓				✓
jstree	✓		✓		
Leaflet	✓				
Leaflet markercluster	✓				
Leaflet.encoded	✓				
Leaflet MiniMap	✓				
Google Maps API	✓				
OpenStreetMap API	✓				
jQuery Datepicker	✓	✓	✓		
Slidebars			✓		
jQuery File Upload					✓
tile.stamen.js	✓				

## 5.1 THE OREGON TRANSIT AGENCIES FLOATING DIALOG BOX

Figure 5.1 depicts the Oregon Transit Agencies (OTAs) floating dialog box, which is used in the TNA software tool to store a transit agency tree-like menu. The OTAs floating dialog box is implemented using the libraries jQuery UI and jQuery UI dialogextend. The OTAs floating dialog box is transparent to allow tracking of changes that take place on the main map interface when the contents of the OTA floating dialog box are clicked. The OTAs floating dialog box can also be moved, resized, minimized, collapsed, and maximized.



**Figure 5.1: The hierarchical structure of the extended OTAs floating dialog box**

Inside the OTAs floating dialog box, a tree-like menu is implemented to view and select one or multiple transit agencies for visualization. The transit agency tree-like menu has three levels:

- **Level 1.** Lists all transit agencies in the state of Oregon in alphabetical order. Clicking an item at this level unchecks all of its child nodes (if checked) and displays the stops that belong to the selected transit agency on the main map interface. Right clicking on a transit agency name opens a menu that allows displaying/hiding all trip shapes served by the transit agency.
- **Level 2.** Lists all the routes that belong to the selected transit agency (parent node). Selecting any nodes at this level unchecks its parent node (if checked) and displays the stops that are served by the selected route. Right clicking on a route name opens a menu that allows displaying/hiding all trip shapes served by the route.
- **Level 3.** Lists all trips with unique shapes for the route (expanded parent node). Checking a node at this level, displays the trip shape on map.

The open source JavaScript plug-in `jstree` is used to implement the tree-like menu used in the OTAs floating dialog box. The `jstree` plug-in is customized so that any of the nodes in the tree-like menu can be selected by selecting one of the available checkboxes. Selecting a checkbox in the tree-like menu triggers an event based on the type of node selected (i.e., either a transit agency, a route, or a trip). In order to populate the tree-like menu, a JSON object is created on



the server side application (see section 4.2). Therefore, no extra manipulation is required for data conversion, which ensures a swift experience on the client side of the TNA software tool. The tree-like menu is loaded once the client side main page is called and it is used as long as the page remains open.

Right clicking on objects in levels 1 and 2 of the tree-like menu allows the display of all trip shapes on the main map interface. However, only the shape of the longest trip is displayed on the main map interface. In addition, right clicking on level 1 of the tree, provides access to the *On-Map Connected Agencies Report*, which is implemented using jQuery UI, jQuery UI dialogextend and leaflet. The *On-Map Connected Agencies Report* displays the transit agencies that are connected to the transit agency from which the user generated the report, and gives the option to visually track the connections by selecting agencies listed in the OTAs floating dialog box.

## **5.2 LOCATION SEARCH BOX**

The location search box enables searching of specific locations based on their names (e.g., cities, street addresses, places and businesses, etc.) The location search box is implemented using the free and open source OpenStreetMap (OSM) API.

## **5.3 MAP TILES**

The OSM layer and the Toner layer by Stamen Design (33) were available in the earlier version of the TNA software tool. The toner map tiles provide improved visualization and printing of a transit agency characteristics (i.e., stops and routes). Additionally, the Google Aerial photography map tiles replaced the OSM Aerial map tiles used in the former version of the TNA software tool to provide higher quality imagery using the Google API.

## **5.4 MAP-IN-MAP FEATURE**

The map-in-map feature provides a broader view of the area that is being displayed in the main map interface of the TNA software tool. The map-in-map feature is implemented with the open source JavaScript library Leaflet MiniMap and utilizes an orange rectangle to indicate which region of the state of Oregon (or other large geographical area) is being displayed in the main map interface. The map-in-map feature can also be used for moving over large areas and can be minimized by clicking on the arrow icon on its lower right corner.

## **5.5 DISPLAYING STOPS AND ROUTE SHAPES**

Methods were developed on the server to provide stops and trip shapes to the client side of the TNA software tool. The JavaScript library Leaflet was used to implement all the visualization capabilities of the TNA software tool.

When displaying stops, the list of stop names and coordinates are provided. The trip shapes are received in encoded polyline form from the server and, therefore, need to be decoded into

coordinates in order to be displayed on the main map interface using Leaflet. `Leaflet.encoded` is a plug-in for leaflet that adds the capability to encode and decode polylines to Leaflet.

## 5.6 CLUSTERING STOPS

The TNA software tool displays stops as clusters that change in size as the level of zooming is changed. The software component used to implement this functionality is the plug-in library `Leaflet.markercluster`.

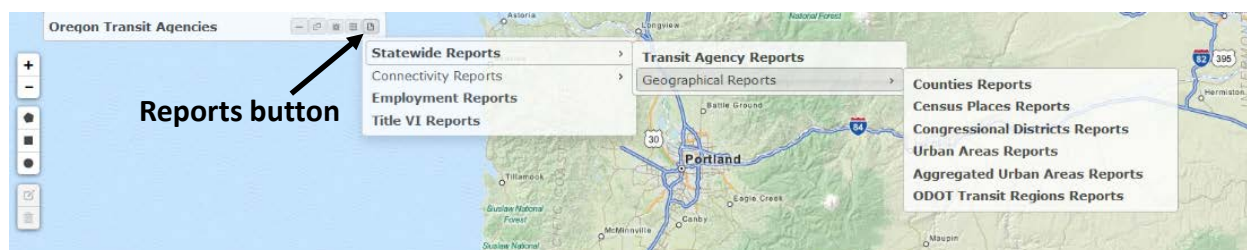
As the mouse is hovered over stops clusters that represent more than two stops, a convex polygon (convex hull) shows the area covered by the transit stops present in the cluster. In the new version of the TNA software tool, a stops cluster becomes transparent when the polygon is displayed to provide more contrast and better visibility.

## 5.7 REPORTS

The new version of the TNA software tool has an improved reports interface with many new reports added. Reports are now displayed in a tabular format with various features. The library `jQuery UI DataTables` is used to display report content. This library allows reports to be sorted, searched, and presented in several pages to improve usability. The library `jQuery UI DataTables` also supports exporting reports in CSV formats. Also the `jQuery DatePicker` library is used to enable date selection by the user.

With the exception of the “Transit Agency Summary” report and the “Stops” report, all the transit agency reports in the TNA software tool include a full calendar and a date picker tool. With the full calendar and date picker, a report can be generated for a specific date or a selection of dates. Also, the generated reports are more accurate since the exceptions in service are also taken into account. For an extensive definition of the metrics used in reports, refer to Appendix A.

The **Reports** button, located on the far right of the OTAs floating dialog box (see Figure 5.2), opens a menu that shows a list of all available reports in the TNA software tool.



**Figure 5.2: Reports interface of the TNA software tool**

The option *Transit Agency Reports* provides access to the “Transit Agency Summary” report, which contains general information on all transit agencies in the state of Oregon, as well as access to the “Routes” report, “Schedule” report, “Stops” report, and “Transit Agency Extended” report. Other report options include the *Counties Reports*, *Census Places Reports*, *Congressional*

*Districts Reports, Urban Areas Reports, and ODOT Transit Regions.* Clicking on any of the mentioned report categories opens summary level reports which presents a list of the selected geographic areas in the state of Oregon. Clicking on the Geo ID of the geographic areas in the list will open the extended reports for the selected area to provide more in depth metrics.

## 5.8 DISPLAYING GEOGRAPHICAL SHAPES ON THE MAIN MAP INTERFACE

The TNA software tool can display the shapes of all Oregon counties, ODOT transit regions, urbanized areas (i.e., urban areas with population over 50,000), and congressional districts as a map layer. The corresponding shapes were obtained from the US Census Bureau (5) in *.shp* format (i.e., the native format provided by the US Census Bureau) and imported into the PostgreSQL relational database. The shape data were then queried and converted to GeoJSON format (i.e., the format that the library Leaflet supports) using a PostgreSQL query. The resulting file was fed to the online mapshaper tool (34) to reduce the size of the layer and make it more suitable for visualization purposes by removing a portion of the points from the shapes and also discarding a few decimal points off the coordinate values. This process results in slightly less accurate, but very small shapefiles that can be easily overlaid on top of many layers such as stops and trip shapes on the main map interface. Every geographic area shape includes name and surface area (in square miles) which are displayed in an `L.Control.Locate` Leaflet object that shows up when the mouse is hovered over a county shape on the map.

All shape layers are styled to be transparent to allow other layers beneath it to remain visible. There are also functions that slightly highlight the shapes and update the data displayed in the box over the map when the mouse is hovered over a shape. If a shape is clicked on with the mouse, the map zooms in on the corresponding area.

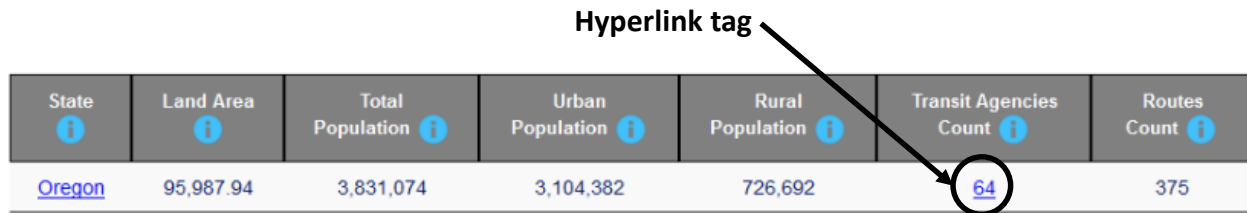
## 5.9 LINKING TABULAR REPORTS

Some instances of tabular reports are linked to other, lower-level reports through hyperlinks. To implement this feature, a hyperlink tag was added to the corresponding fields in the parent report (i.e., the report that provides access to a lower level report). Figure 5.3 depicts an example of one of these hyperlinked tags in the Statewide Summary Report (**Note:** Only a portion of the Statewide Report is displayed in the picture for illustration purposes).

Since these links are not actual hyperlinks (i.e., they do not refer to a URL), a JavaScript function was implemented to capture the click events. Two custom attributes (i.e., `Type` and `ID`) were necessary to identify the type and parameters of the report that needed to be loaded. Therefore, each time a hyperlink is clicked, the JavaScript function decides which type of report to load by looking into the `Type` attribute and then passes the appropriate parameters to the query that generates the report by looking into the `ID` attribute.

## Transit Network Analysis Tool Reports

### Statewide Summary Report



The screenshot shows a table with 7 columns: State, Land Area, Total Population, Urban Population, Rural Population, Transit Agencies Count, and Routes Count. Each column has an information icon (i) in the header. The first row of data is for Oregon. The value '64' in the 'Transit Agencies Count' column is circled, and a callout line labeled 'Hyperlink tag' points to it.

State	Land Area	Total Population	Urban Population	Rural Population	Transit Agencies Count	Routes Count
<a href="#">Oregon</a>	95,987.94	3,831,074	3,104,382	726,692	64	375

Search:

Showing 1 to 1 of 1 entries

**Figure 5.3: Example of a hyperlinked tag in the statewide summary report**

## 5.10 ON-MAP REPORT

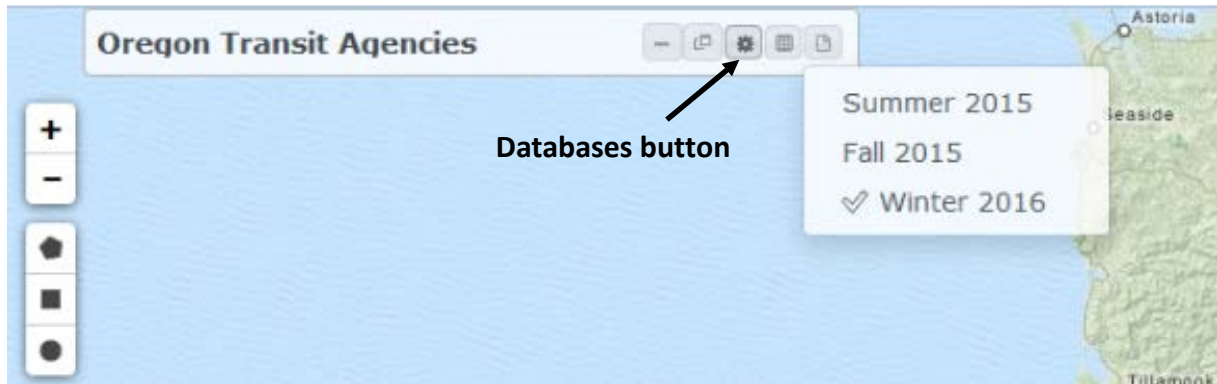
The on-map report can be invoked by either drawing a geometric shape (e.g., circle, rectangle, or a polygon) on the main map interface, or by clicking on a single stop cluster to reveal a window with a button to generate the on-map report.

The library `Leflet.draw` is used for drawing, editing, and deleting shapes on the main map interface of the TNA software tool. Once the shape is drawn, it is sent to the server to generate the on-map report. The contents of the on-map report are displayed using a jQuery UI dialog box on the main map interface. Invoking an on-map report hides the other layers (e.g., stops, routes, etc.) and collapses the OTAs floating dialog box.

## 5.11 MULTI DATABASE FUNCTIONALITY

The multi database feature is implemented on both the visualization and reporting interfaces of the GUI in the TNA software tool. A button was added to the OTAs floating dialog box to allow a user to specify the database that should be used for analyses purposes (see Figure 5.4). When the TNA software tool loads, the first database in the list is chosen as the default database. Changing the database on the visualization interface reloads the page (which removes all visualized objects such as stops, routes, and on-map reports from the map), re-populates a new instance of the OTAs floating dialog box based on the new database, and all new visualizations and on-map reports will be from the new database from this point. The multi database feature is also available in the reports interface through a menu box on the top right corner of every report. Changing the database on any report, reloads the report with the new values extracted from the selected database.

A method is developed in the WebApp to allow the GUI of the TNAST to retrieve available database names and populate the database list based on that information. To implement the multi database feature, a new parameter named “dbindex” is added to all API commands. This parameter is also visible in the address bar of the web browser at the very end of the URL. This way, the database parameter is sent to another page while opening a new report.



**Figure 5.4: Location of the databases button on the OTA floating dialog box**

## 5.12 PLAYGROUND INTERFACE

The backbone of the Playground interface, which deals with uploading or deleting GTFS feeds by the user, is designed via the JavaScript library `jquery.fileUpload`. This library uses asynchronous connections to the server to send and save the uploaded GTFS zip files.

Since there is limited space on the server for uploading GTFS files, the number of users that can register for the Playground and the amount of space that each user is allowed for uploading GTFS feeds is limited. At this point, the maximum number of users that can register for the Playground interface is 20, and each user is provided 10 MB of space to store GTFS feeds as zip files.

Once logged in, users can see the amount of free space they have left for uploading additional GTFS feeds. If they try to upload a file (or files) that will exceed their available space, the TNAST blocks the upload process using the JavaScript library `jquery.fileUpload` and the user is notified.

## 6.0 CONCLUSIONS AND FUTURE WORK

The main objective of this project was to add new features to the TNA software tool and to enhance those that were developed in the first and second phases of development.

The most current version of the TNA software tool now contains the GTFS feeds of 64 different Oregon public and private transit agencies (the prior version included only 49 GTFS feeds), in addition to census data, park and ride data, employment data, and Title VI data. These data sources are used to populate different kinds of reports (both in visual and in tabular format), which should provide planners and analyst with a wide variety of options to visualize and evaluate the statewide public and private transit network.

A special version of the TNA software tool called the “*playground*” was also completed in this phase of development. Through the playground, a user can selectively add either a subset of the available 64 GTFS feeds and then visualize and generate tabular reports based exclusively on the transit agencies added. Furthermore, users have the option to create their own GTFS feed (or make changes to an existing one), add this newly created feed to the playground, and analyze the impact of these changes on the performance of their system.

It is important to note that the necessary documentation and source code to install and run the TNA software tool (as well as the playground version) is currently available at GitHub (7). This documentation and source code will be updated as the project progresses and additional enhancements to the TNA software tool are completed.

## 7.0 REFERENCES

1. Porter, J. D., Kim, D. S., Ghanbartehrani, S., Mohseni, S.A. *An Open Source Tool for the Visualization, Analysis, and Reporting of Regional and Statewide Transit Networks*. Publication SPR 13-075, Jan 2015.
2. PostgreSQL. PostgreSQL. *PostgreSQL: About*. [Online] [Cited: Feb. 24, 2016.] [www.postgresql.org/about/](http://www.postgresql.org/about/).
3. Inc., Google. General Transit Feed Specification Reference - Transit. *Google Developers*. [Online] [Cited: September 04, 2013.] [https://developers.google.com/transit/gtfs/reference#frequencies\\_fields](https://developers.google.com/transit/gtfs/reference#frequencies_fields).
4. Hughes, J. Proposal: Provide a way to indicate entrances and pedestrian pathways - Google Groups. [Online] [Cited: December 24, 2014.] <https://groups.google.com/forum/#!msg/gtfs-changes/Gk88QYQYgtw/UNdVCH0v-RIJ>.
5. United States Census Bureau. *TIGER Products - Geography*. [Online] [Cited: September 3, 2013.] <http://www.census.gov/geo/maps-data/data/tiger.html>.
6. Eclipse Foundation. Eclipse. *The Eclipse Foundation open source community website*. [Online] [Cited: Jan 30, 2014.] <https://www.eclipse.org>.
7. Ghanbartehrani, S., Mohseni, A. and Barahimi, P. GitHub. *tnatool/test*. [Online] [Cited: Feb 24, 2016.] <https://github.com/tnatool/test/>.
8. Red Hat Projects Community. *Hibernate ORM*. [Online] [Cited: Dec. 7, 2014.] <http://hibernate.org/orm/>.
9. Geovise open source GIS solutions. *Overview / hibernate Spatial*. [Online] [Cited: Dec. 23, 2014.] <http://www.hibernatespatial.org/>.
10. Onebusaway project. *onebusaway-gtfs - About*. [Online] [Cited: Dec. 23, 2014.] <http://developer.onebusaway.org/modules/onebusaway-gtfs-modules/1.3.4-SNAPSHOT/onebusaway-gtfs/index.html>.
11. Onebusaway project. *onebusaway-gtfs-hibernate - About*. [Online] [Cited: Dec. 23, 2014.] <http://developer.onebusaway.org/modules/onebusaway-gtfs-modules/1.3.4-SNAPSHOT/onebusaway-gtfs/index.html>.
12. Onebusaway project. *onebusaway-gtfs-hibernate-cli - About*. [Online] [Cited: Dec. 23, 2014.] <http://developer.onebusaway.org/modules/onebusaway-gtfs-modules/1.3.4-SNAPSHOT/onebusaway-gtfs-hibernate/index.html>.
13. Open Source Geospatial Foundation. *GeoTools Documentation - GeoTools*. [Online] [Cited: Dec. 23, 2014.] <http://docs.geotools.org/>.
14. JDBC - Oracle. *Java SE Technologies - Database*. [Online] [Cited: Feb. 24, 2016.] <http://www.oracle.com/technetwork/java/javase/jdbc/index.html>.
15. World Wide Web Consortium. *W3C HTML*. [Online] <http://www.w3.org/html/>.
16. World Wide Web Consortium. *Cascading Style Sheets*. [Online] [Cited: Jan. 30, 2014.] <http://www.w3.org/Style/CSS/Overview.en.html>.
17. Flanagan, D. *JavaScript: the definitive guide*. Beijing : O'Reilly Media, Inc., 2011.
18. The jQuery Foundation. *jQuery*. [Online] [Cited: Sep. 01, 2013.] <http://jquery.com/>.
19. The jQuery Foundation. *jQuery UI*. [Online] [Cited: Jan. 1, 2013.] <http://jqueryui.com/>.
20. SpryMedia Ltd. *DataTables: Table plug-in for jQuery*. [Online] [Cited: Dec. 23, 2014.] <http://www.datatables.net/>.

21. ROMB. *Jquery-dialogextend*. . [Online] [Cited: Dec. 23, 2014.] <http://romb.github.io/jquery-dialogextend/>.
22. Otto (@mdu), M., and Jacob (@fat). *Bootstrap*. [Online] [Cited: Dec. 23, 2014.] <http://getbootstrap.com/javascript/#dropdowns>. .
23. Bozhanov, I. *jsTree » Home*. [Online] [Cited: Dec. 24, 2014.] <http://www.jstree.com/>. Accessed Aug. 26, 2013..
24. Agafonkin, V. *Leaflet - a JavaScript library for mobile-friendly maps*. [Online] [Cited: Sep. 3, 2013.] <http://leafletjs.com/>.
25. Timberlake, A. *Leaflet/Leaflet.markercluster*. [Online] [Cited: Sep. 3, 2013.] <https://github.com/LeafLet/leaflet.markercluster>.
26. Waagmeester, J. P. *Leaflet.encoded*. [Online] [Cited: Aug. 25, 2013.] <https://github.com/jieter/Leaflet.encoded>.
27. Nordan, N. R. *Leaflet-MiniMap · GitHub*. [Online] [Cited: Dec. 23, 2014.] <https://github.com/Norkart/Leaflet-MiniMap>.
28. Google Inc. *Google Maps JavaScript API v3 - Google Developers*. [Online] [Cited: Dec. 23, 2014.] <https://developers.google.com/maps/documentation/javascript/tutorial>.
29. OpenStreetMap Project. *API - OpenStreetMap Wiki*. [Online] [Cited: Dec. 24, 2014.] <http://wiki.openstreetmap.org/wiki/API>.
30. The jQuery Foundation. *Datepicker / jQuery UI*. [Online] [Cited: Feb. 24, 2016.] <https://jqueryui.com/datepicker/>.
31. Smith, A. C. *Slidebars - Adam Charles Smith*. [Online] [Cited: Feb. 23, 2016.] <http://http://plugins.adchsm.me/slidebars/>.
32. Tschan, S. *jQuery File Upload Demo*. [Online] [Cited: Feb. 23, 2016.] <https://blueimp.github.io/jQuery-File-Upload/>.
33. Stamen Design. [Online] [Cited: Feb. 23, 2016.] <http://maps.stamen.com/>.
34. Bloch, M. *mapshaper*. [Online] [Cited: Dec. 24, 2014.] <http://mapshaper.org/>.



## APPENDIX A: TABULAR REPORTS METRICS DEFINITION

Report Name	Metric Name	Definition/Computation method
Geographical Areas Extended Report	Route Miles	Summation of the lengths of the longest trips within the given geographic area
Geographical Areas Extended Report	Stops per square mile	Stop count in the given geographic area divided by the area of the geographic area
Geographical Areas Extended Report	Stops per service mile	Stop count in the given geographic area divided by service miles
Geographical Areas Extended Report	Service hours	Total hours a transit agency spends on serving all round trips of routes within the given geographic area. Service hours may be calculated for a specific date or a set of dates specified using the calendar. Reported number are cumulative over the selected dates.
Geographical Areas Extended Report	Service miles	Total miles driven over all round trips of routes within the given geographic area. Service miles may be calculated for a specific date or a set of dates specified using the calendar. Reported number are cumulative over the selected dates.
Geographical Areas Extended Report	Service miles per square mile	Service miles divided by the area of the geographic area.
Geographical Areas Extended Report	Miles of service per capita	Service miles divided by the population of the geographic area.
Geographical Areas Extended Report	Urban population served (unduplicated)	Summation of unduplicated urban population within an X-mile radius (i.e., stop distance) of all stops within the given geographic area.
Geographical Areas Extended Report	Rural population served (unduplicated)	Summation of unduplicated rural population within an X-mile radius (i.e., stop distance) of all stops within the given geographic area.
Geographical Areas Extended Report	Percent of population served	Summation of unduplicated population within an X-mile radius (i.e., stop distance) of all stops within the given geographic area divided by total population of the area.

Report Name	Metric Name	Definition/Computation method
Geographical Areas Extended Report	Percent of population served at level of service	Percent of Population Served at Level of Service <em title="Summation of unduplicated population within an X-mile radius (i.e., stop distance) of all stops within the given geographic area that receive a specified minimum level of service divided by total population of the given geographic area.
Geographical Areas Extended Report	Urban population served at level of service	Urban Population Served at Level of Service <em title="Summation of unduplicated urban population within an X-mile radius (i.e., stop distance) of all stops within the given geographic area that receive a specified minimum level of service.
Geographical Areas Extended Report	Rural population served at level of service	Rural Population Served at Level of Service <em title="Summation of unduplicated rural population within an X-mile radius (i.e., stop distance) of all stops within the given geographic area that receive a specified minimum level of service.
Geographical Areas Extended Report	Percent of Population Unserved	Percent of Population Unserved. One minus the percent of population served.
Geographical Areas Extended Report	Service Stops	Total stops within the given geographic area multiplied by the number of times each stop is being served for the given date(s). Reported number is cumulative over the selected dates.
Geographical Areas Extended Report	Urban Population Served By Service	Total unduplicated urban population impacted within an X-mile radius (i.e., stop distance) of all stops within the given geographic area. Urban population served by service is calculated as service stops multiplied by the unduplicated urban population within an X-mile radius (i.e., stop distance) of all stops within the given geographic area for every trip. Reported number is cumulative over the selected dates.
Geographical Areas Extended Report	Rural Population Served By Service	Total unduplicated rural population impacted within an X-mile radius (i.e., stop distance) of all stops within the given geographic area. Urban population served by service is calculated as service stops multiplied by the unduplicated urban population within an X-mile radius (i.e., stop distance) of all stops within the given geographic area for every trip. Reported number is cumulative over the selected dates.
Geographical Areas - Extended	Service Days	Set of days (from the selected days) in which at least one trip within the given geographic area is served.
Geographical Areas - Extended	Connected Communities	List of geographic areas of the same type that are connected to the area of interest through routes.

<b>Report Name</b>	<b>Metric Name</b>	<b>Definition/Computation method</b>
Geographical Areas - Extended	Hours of Service	Earliest and latest arrival and departure times of all transit stops within the given geographic area
Transit Agencies Extended Report	Stops Per Route Mile	Number of stops in the agency's routes divided by Route Miles.
Transit Agencies Extended Report	Service Hours	Total hours the transit agency spends on serving all round trips of routes. Service hours may be calculated for a specific date or a set of dates specified using the calendar. Reported number are cumulative over the selected dates.
Transit Agencies Extended Report	Service Miles	Total miles driven by a transit agency over all round trips of a route. Service miles may be calculated for a specific date or a set of dates specified using the calendar. Reported number are cumulative over the selected dates.
Transit Agencies Extended Report	Urban Population Served (Unduplicated)	Summation of unduplicated urban population within X-mile radius (i.e., stop distance) of all stops that the transit agency serves.
Transit Agencies Extended Report	Rural Population Served (Unduplicated)	Summation of unduplicated rural population within X-mile radius (i.e., stop distance) of all stops that the transit agency serves.
Transit Agencies Extended Report	Service Stops	Number of trips scheduled at a stop in a route. The service stops for a route is calculated as its stop count multiplied by the number of visits per stop. Reported number are cumulative over the selected dates.
Transit Agencies - Extended	Urban Population Served By Service	Total unduplicated urban population impacted within an X-mile radius (i.e., stop distance) of all stops that the transit agency serves. Urban population served by service is calculated as service stops multiplied by the unduplicated urban population within an X-mile radius (i.e., stop distance) of all stops that the transit agency serves. Reported number is cumulative over the selected dates.
Transit Agencies - Extended	Rural Population Served By Service	Total unduplicated rural population impacted within an X-mile radius (i.e., stop distance) of all stops that the transit agency serves. Urban population served by service is calculated as service stops multiplied by the unduplicated urban population within an X-mile radius (i.e., stop distance) of all stops that the transit agency serves. Reported number is cumulative over the selected dates.
Transit Agencies - Extended	Service Days	Set of days (from the selected days) in which at least one trip is served by the selected transit agency.
Transit Agencies - Extended	Hours of Service	Earliest and latest arrival and departure times of all stops served by the transit agency.

Report Name	Metric Name	Definition/Computation method
Hubs report	Visits count	Total number of times all stops in the cluster are served. This metric is calculated by adding up all the number of visits of the stops belonging to the cluster.
Hubs report	Park and ride lots count	Number of park and ride lots located within X-radius of the cluster centroid.
Hubs report	Counties count	Number of counties that the cluster is located in.
Hubs report	Census places count	Number of census places that the cluster is located in.
Hubs report	Population served	Unduplicated sum of the population within the X radius distance of all stops in the cluster. This metric is date-independent.
Hubs report	Urban population	Sum of the population of urban areas that stops in the cluster are located in.
Hubs report	Transit agencies	List of transit agencies that serve at least one of the stops in the cluster.
Connected transit networks	Connected agencies	List of transit agencies that have at least one stop within the specified distance of any other agencies in the list.
Connected agencies summary report	Number of connected agencies	Number of transit agencies that have at least one stop within the X-radius of the selected transit agency stops.
Connected agencies summary report	Connected Agency Names	List of the transit agencies that have at least one stop within the X-radius of the selected transit agency stops.
Connected agencies extended report	Number of connections	Number of pair-wise stops that are located within the X-radius of each other. The source of a connection is a stop belonging to the transit agency for which the report is generated and the destination is a stop that belongs to the transit agency listed in the row.
Connected agencies extended report	Connections	A pair of stops that are located within the X-radius of each other. The source of a connection is a stop belonging to the transit agency for which the report is generated and the destination is a stop that belongs to the transit agency listed in the row.
Employment report	EMPS (WAC)	Employees Served: Unduplicated summation of employees of the selected category working within X-radius distance of any stop. This metric is date independent.
Employment report	EMPSLOS (WAC)	Employees Served at Level of Service: Unduplicated summation of employees of the selected category that receive the specified minimum level of service throughout the selected dates. This metric is date dependent. The WAC dataset is used to calculate this metric.

Report Name	Metric Name	Definition/Computation method
Employment report	EMPSS (WAC)	Employees Served by Service: Unduplicated summation of employees of the selected category served by service is calculated by multiplying service stops by the unduplicated employees working within an X-mile radius (i.e., stop distance) of all stops. Reported number is cumulative over the selected dates.
Employment report	EMPS (RAC)	Employees Served: Unduplicated summation of employees of the selected category living within X-radius distance of any stop. This metric is date independent.
Employment report	EMPSLOS (RAC)	Employees Served at Level of Service: Unduplicated summation of employees of the selected category that receive the specified minimum level of service throughout the selected dates. This metric is date dependent.
Employment report	EMPSS (RAC)	Employees Served by Service: Unduplicated summation of employees of the selected category served by service is calculated by multiplying service stops by the unduplicated employees living within an X-mile radius (i.e., stop distance) of all stops. Reported number is cumulative over the selected dates.
Title VI report	<i>EmploymentCategory-S</i>	Number of Employees Served: Unduplicated summation of individuals of the <i>EmploymentCategory</i> who are living within X distance of any stop. This metric is date/service independent.
Title VI report	<i>EmploymentCategory-SLOS</i>	Number of Employees Served at Level of Service: Unduplicated summation of individuals of the <i>EmploymentCategory</i> who receive the specified minimum level of service.
Title VI report	<i>EmploymentCategory -SS</i>	Number of Employees Served by Service: Unduplicated summation of employees in the <i>EmploymentCategory</i> who are served by service is calculated as service stops multiplied by the unduplicated individuals living within an X-mile radius (i.e., stop distance) of all stops. Reported number is cumulative over the selected dates.