Development of a Web-Based System for Water Quality Data Management and Visualization

Wei Yang

Thesis submitted to the Graduate Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements of the degree of:

MASTER OF SCIENCE
in
ENVIRONMENTAL ENGINEERING

Approved:

Thomas J. Grizzard, Chair
Adil N. Godrej
Harold E. Post

May 03, 2010

Falls Church, Virginia

Keywords: Watershed Management, Water Quality Database, Web-based System

Development of a Web-Based System for Water Quality Data Management and Visualization

by

Wei Yang

Committee Chair: Dr. Thomas J. Grizzard

Abstract

With increasing urbanization and population growth, humankind faces multiple environmental challenges. Stresses on limited resources, especially water resources, are now greater than ever before. Watershed monitoring and management are important components of programs to abate water resource stresses. The increasing water quantity and quality monitoring has produced a need for better data management techniques to manage the vast amount of watershed monitoring data being observed. These data must be stored, error checked, manipulated, retrieved and shared with the watershed management community. The web-based data visualization and analysis technology has played a critical role in all aspects of watershed management. Especially in recent years, computer-assisted data analysis has matured enormously. This maturing technology makes web-based visualization and analysis technology change its role to become an integrated system which combines applications of databases, and internet technology.

The main objective of this study is to develop a prototype system which has ability of data visualization and analysis. Microsoft SQL Server is used to build a comprehensive database, which includes all datasets collected by OWML. A Web-Based Data Visualization and Analysis System which provides an integrated interface for permitted users to explore, analyze and download data has been developed.

ACKNOWLEDGEMENTS

I am thankful to be part of this project sponsored by the Occoquan Watershed Monitoring Program. It has given me the opportunity to explore a topic of great interest to me. The Occoquan Watershed Monitoring Laboratory has proven to be a great place for me to do research as well as to make good friends.

I would like to express my deepest acknowledgements and thanks to my committee members, Thomas J. Grizzard, Adil N. Godrej and Harold E. Post, for taking time from their busy schedules and offering me their outstanding guidance, suggestion and help. I am also thankful to my good friend Saurav Kumar, who provided much assistance and encouragement during the development of this project.

Most of all, I want to acknowledge my wife, for her endless love, support and patience. She makes everyday a wonderful day. Without her support and encouragement, I might not have finished this process. I also express my thanks to my parents for their support during this journey.

Contents

Lists of Tables

List of Figures

1. Introduction

In 2008, the world reached an invisible but momentous milestone: for the first time in human history, more than half of the populations, 3.3 billion people, were living in urban areas. By 2030, this is expected to swell to almost 5 billion (UNFPA 2009). With continuing population growth and urbanization, humankind faces multiple environmental challenges. Stress on limited resources is now greater than ever before. Water resources, one of the most important and essential materials in day-to-day life, are becoming scarce for many reasons, including overexploitation of surface water resources and reduction in infiltration rates to recharge groundwater. Climate change can also be expected to create new stresses on water resources in the future. Watershed management has evolved as an important part of solutions to abate water resources stress. Watershed management is an iterative process of integrated decision–making regarding uses and modifications of land and water within a watershed. The process can provide an opportunity for stakeholders to balance diverse goals and uses for environmental resources, and to consider how their cumulative actions may affect long-term sustainability of these resources.

An understanding of the complex nature of watershed systems would be crucial for developing a holistic approach to optimizing the utilization and management of water resources. For managing and analyzing water quality data, web-based visualization technology may play a critical role, from assessing watershed conditions through modeling impacts of human activities on water quality and visualizing impacts of alternative management scenarios (Gupta et al. 2004). The Computer-assisted geographic analysis capabilities necessary for such visualization have matured enormously over the past decade. In recent years, web-based visualization technology has become better integrated into systems which combine geospatial applications with database, data mining, and internet technology (Tim et al. 2003).

1.1 Statement of Problem

Building a watershed analysis system is useful for supporting watershed management tasks. But such an analysis system is a complex undertaking. There are two key components for developing such systems: (1) the data which are used for analysis, and (2) the platform which can be used for data analysis and distribution. Several studies using GIS techniques have resulted in the development of useful analysis systems. However, they have seldom been widely used due to the lack of efficient and easy-to-use delivery mechanisms. Because public participation is critical for effectively managing natural resources, it is important to insure that the analysis system can be widely and easily accessed which is a key determinant of a successful system. It is also important that the information and knowledge should be managed in a proper way so that members of the public without professional background can easily understand the information. In order to provide timely and easily access to the data, an efficient mechanism for information delivery is necessary (Racicot 2005).

1.2 Objectives

The overall objective of this research was to develop a prototype water quality data visualization and analysis system that could be used for information publishing, data analysis, and watershed management in the Occoquan watershed through a web-based environment. This overall objective can be divided into several sub-objectives. Specifically, the sub-objectives of this research included:

- To collect and check historical observed data of various types, including water quality, meteorologic, and hydrologic data developed by the Occoquan Watershed Monitoring Laboratory (OWML) for construction of a Watershed Monitoring and Analysis Database;

- Using Microsoft SQL Server (Delaney 2000), to design and construct a relational database, to include the various datasets collected by OWML

- To develop a Web-Based Data Visualization and Analysis System that provides an integrated, geospatially-referenced interface to permit users to explore, analyze and download data.

1.3 Approach

This research consisted of two major parts, the design and construction of a comprehensive database and the development of a web-based data visualization and analysis system.

In the database construction activity, the focus was to develop a comprehensive database which included all data sets developed by OWML using Microsoft SQL Server Database Management System (MS SQL Server DBMS). The new developed database would be required to manage the very large amount of historical and new watershed data being collected, and also to have a robust structure to meet the requirements of data integrity and minimum redundancy. By the construction of such a database, some advanced tasks such as quality assurance/quality control (QA/QC) calculations, application of correction and conversion factors, retrieval of desired data for advanced analysis, and data comparisons among multiple study sites can be simplified with the help of the database application. Web integration and local area network (LAN) database synchronization was also part of the design requirement (Carleton et al. 2005).

The web-based data visualization and analysis system was also required to provide a user-friendly interface to allow authorized users to explore, plot and download observed data for selected time ranges. This system was also to have basic built-in map interface functions such as zoom in, zoom out, pan, orientation, selection of different base map view, and other features.

1.4 Outline of the Thesis

The remainder of this document consists of five chapters that describe efforts to meet the objectives outlined in this chapter. Chapter 2 describes the study area selected for

the research. Chapter 3 describes the design and construction of the database; including data selection, data enhancement, and database management. Chapter 4 addresses the development of a watershed management and analysis system. Chapter 5 presents the project results, and Chapter 6 summarizes the study and proposes recommendations for future study. Appendices include major parameter sources, database structure and program codes.

2. Literature Review

This literature review contains information about the database design and construction, and how Web-based technology has been used on data analysis and visualization.

2.1 Water Resource Database

Data collection is a major task in watershed monitoring, research, and management. Very large amount of water resource data are often collected to monitor watershed conditions. In the course of water resource data collection, historical and real-time series data, including water quality, rainfall, stream flow and weather data are collected by watershed management organizations, and most of the time these data are stored in spreadsheet files or in databases that are not well-integrated (Beran et al. 2008).This deficiency causes researchers and other users to spend tremendous amounts of time and manual work to convert data into formats which can be directly used for data exploration, analysis, and visualization.

In recent years, there has been an increasing trend by watershed management organizations to share their water resource databases on the internet. Although many databases are now available online at no charge, it is still difficult for researchers to quickly access desired information in a consistent and integrated framework (McGuire 2008).

Some large federal agencies have developed their own unique databases. For example, the U.S. Geological Survey has developed the USGS National Water Information System (NWIS), which is a historical and real-time database of water quality, stream flow, and groundwater level observations in the United States over the past century (USGS_NWIS 2009).

Another example is the U.S. Environmental Protect Agency (EPA) Storage and Retrieval (STORET) database, which contains historical water quality observations

collected by various local, state, and federal agencies, and also by different research institution and organizations (USEPA_STORET 2009).

An additional example is provided by the hydrologic research community working to address issues of data interoperability with the development of a Hydrologic Information System (HIS) through the Consortium of Universities for the Advancement of Hydrologic Science, Inc.(CUAHSI) (Goodall et al. 2006; Zheng 2005).

The Chesapeake Bay Monitoring Program, operated by multiple states and organizations including Maryland, Virginia, Pennsylvania, the District of Columbia, U.S. U.S. Environmental Protection Agency and other institutions, has focused on the protection and restoration of the Chesapeake Bay since 1983. The public and program partners rely on monitoring program data to observe changes in the state of the bay and its watershed. The monitoring program also provides guidance to state and local government to enhance understanding and promote protection of the natural environment. It also shares valuable information to decision makers in environment protection policy development. In order to facilitate these activities, the program also developed an accessible database including water quality data, living resources data, and modeling outputs. The data may be queried by users with different inputs such as geopolitical locations and date range. The result of each query is a downloadable text file which may be imported into any program such as Microsoft Excel, Access and SAS for further analysis (Hassett et al. 2005).

There are many other local organizations also serving water resource data on the web. Some examples are identified in Table 1. Compared with USGS or USEPA, which mainly focus on hydrology and water quality data, these local agencies provide services of different types, including water resource data, air quality data, land use data and socio-economic data.

Table 1. Organizations Serving Water Resource Data (USGS 2010).

Name	**Location**	**Data Served**
Independence Lake	Maple Plain, MN	Limited Water Quality Data
Chester Creek	City of Duluth, MN	Water Chemistry, Storm Event
Lake Washington	King County, WA	Limited Water Quality Data
Seneca Rive	New York	Water Quality Data
Chesapeake Bay	District of Columbia, New York, Pennsylvania, Delaware, Maryland, Virginia, and West Virginia	Pollutants, Water Quality, Land Use, Algae, Fish, Crabs and Submerged Aquatic Vegetation
Water Resources Data System (WRDS)	Wyoming	Stream Flow, Ground Water, Water Quality and Climate Data
Fairfax County Monitoring Network	Fairfax County, Virginia	Limited Water Quality Data
Pima Association of Governments Regional Data Center	City of Tucson, Arizona	Air Quality Application, Orthophoto Application, Population, Planning Application
Paso Del Norte Maps for Public Access	UTEP, El Paso, TX	Aerial Photography, Parcel Features, Street Features, Misc. Data
Resource Geographic Information System	University of New Mexico, Albuquerque	Reference Data
Texas Commission on Environmental Quality	TCEQ Austin	Limited Water Quality Data
Texas Natural Resources Information System	TNRIS Austin, TX	Environmental Applications, Bay Water Applications (sea-level and salinity)

2.2 Web-based Data Visualization

Scientists and researchers are increasingly interested in developing effective and practical data visualization methods. In most cases, users prefer to visualize datasets so that they are able to have a better understanding of the data before having them downloaded (Jeong et al. 2005). Currently, most data publishers do not offer the full range of services that would be desired by the user community. Furthermore, to develop advanced data analysis and visualization applications which can be easily applied to different data formats, is not only difficult, but also often felt to be unprofitable for those data providers (Liu et al. 2003).

In the hydrologic community, GIS tools are widely used to serve data for stream networks and to display spatially-reference images (Frankenberger et al. 1999). The University Corporation for Atmospheric Research (UCAR) is operates a Joint Office for Science Support (JOSS) which provides some tools for retrieving data from different data sources (JOSS 2009).

The Modular Modeling System (MMS) developed and operated by the U.S. Geological Survey (USGS), is an integrated system which has the capability to support integration of multiple ecosystem and hydrological models. The MMS allows researchers to develop model components in their own research areas. The general framework provided by MMS allows users to address complex issues when using different environmental and hydrological models. MMS is also being used for development of water resource and environmental resource management tools (Leavesley et al. 2005).

Another example is the Geosciences Network (GEON), which was started in 2002 as a project funded under the NSF Information Technology Research (ITR) program. The project focused on the development of the geoinformatics, which focused on building a cyberinfrastructure for the geosciences community. The project has various goals, such as interoperability of databases, data retrieval, and data visualization (Salayandia et al. 2006). However, the geosciences mainly focus on data about subsurface structures and conditions, and the specific applications of geosciences are quite different from those of

hydrology. As a result, it is not generally appropriate to compare GEON-based tools or applications directly with hydrologic applications, even though they may share similar approaches and goals (Liu et al. 2003).

Besides using GIS tools for data visualization, *Data-Driven Application* technologies have the ability to incorporate data with an executing application such as online data archival, data retrieval and data collection, and have been widely used for data visualization (Darema 2004). As an example, a data driven website is a web application which can use data from a database to dynamically assemble web pages using live data from a database (Ceri et al. 1999).

Louisianan State University has developed a Dynamic Data Driven Application System for Hurricane Forecasting (DDDAS) which is an integrated, comprehensive, computational framework for meteorological, coastal, and ecological models (Allen 2007). In this system, real time and historical data from different sources have been collected and stored with a local storage device and a remote storage device. The system is being developed by adapting and extending existing software packages using dynamic data driven applications (Allen 2007). The visualization component of the system can predict storm time and path. The prediction results can be used for easy understanding and also can enhance public awareness.

Data Driven Application technology is also used for wildfire simulations. Researchers at the University of Colorado at Denver are building a Dynamic Data Driven Application System for short-range forecasts of wildfire. The forecast will be changed by system when new data are received. A web-based interface and visualization are also provided. The visualization results on the web site was developed using Java applets. The real time and/or near real time (NRT) visualization are supported by the system. The system was also designed with separate visible layers to present information such as maps, wind, fuel, fire location and probability of burning. The user can turn the layers on and off or modify their transparency (Mandel et al. 2005).

3. Study Area

Occoquan Reservoir Watershed

The study area for this research project is the Occoquan Reservoir Watershed, which is shown in Figure 1. The basin drains 590 square miles and includes the 1700 acre

Figure 1. Location of Occoquan Watershed (Source: OWML).

Occoquan Reservoir which lies on the boundary between Fairfax County and Prince William County. The watershed lies in Northern Virginia, on the western edge of the Washington D.C. suburbs. The basin reaches into six political subdivisions of the

Commonwealth Virginia, including portions of four counties (Fairfax County, Fauquier County, Loudoun County, and Prince William County) and the entire land area of two cities (City of Manassas and City of Manassas Park). The Occoquan Watershed is divided into two major sub-watersheds drained by two major streams: Occoquan River and Bull Run. The Occoquan River sub-watershed lies on the south and has two other major water bodies, Lake Manassas and Lake Jackson, and two principal tributary streams, Broad Run and Cedar Run (OWML 2009).

There are two major drinking water sources within the watershed. Lake Manassas, located in the upper part of the watershed, serves as the principal drinking water supply for the City of Manassas. The storage capacity reported by Eggink (2001) in the year 2000 was 4.078 billion gallons (BG). The Occoquan Reservoir is part of the water supply system for over one million residents served by Fairfax Water, and is also a major recreation site in Northern Virginia. The storage capacity is about 8.5 BG. Other major characteristics of the Occoquan Reservoir and Lake Manassas may be found in Table 2.

Table 2. Characteristics of Occoquan Reservoir and Lake Manassas (Xu et al. 2007).

Waterbody characteristics	Occoquan Reservoir	Lake Manassas
Volume (BG)	8.5	4.1
Surface area (acre)	1880	697
Length (mile)	14	3.71
Mean depth (feet)	14	18
Maximum depth (feet)	65	49.2
Mean width (feet)	492	1158
Maximum width (feet)	902	2375
Hydraulic residence time (day)	19.6	118.5

The Occoquan Reservoir, which is shown in Figure 2, is the first major drinking water supply in the U.S. to have planned water reuse to supplement the supply for more than one million people who live and work in Northern Virginia (Dougherty et al. 2002).

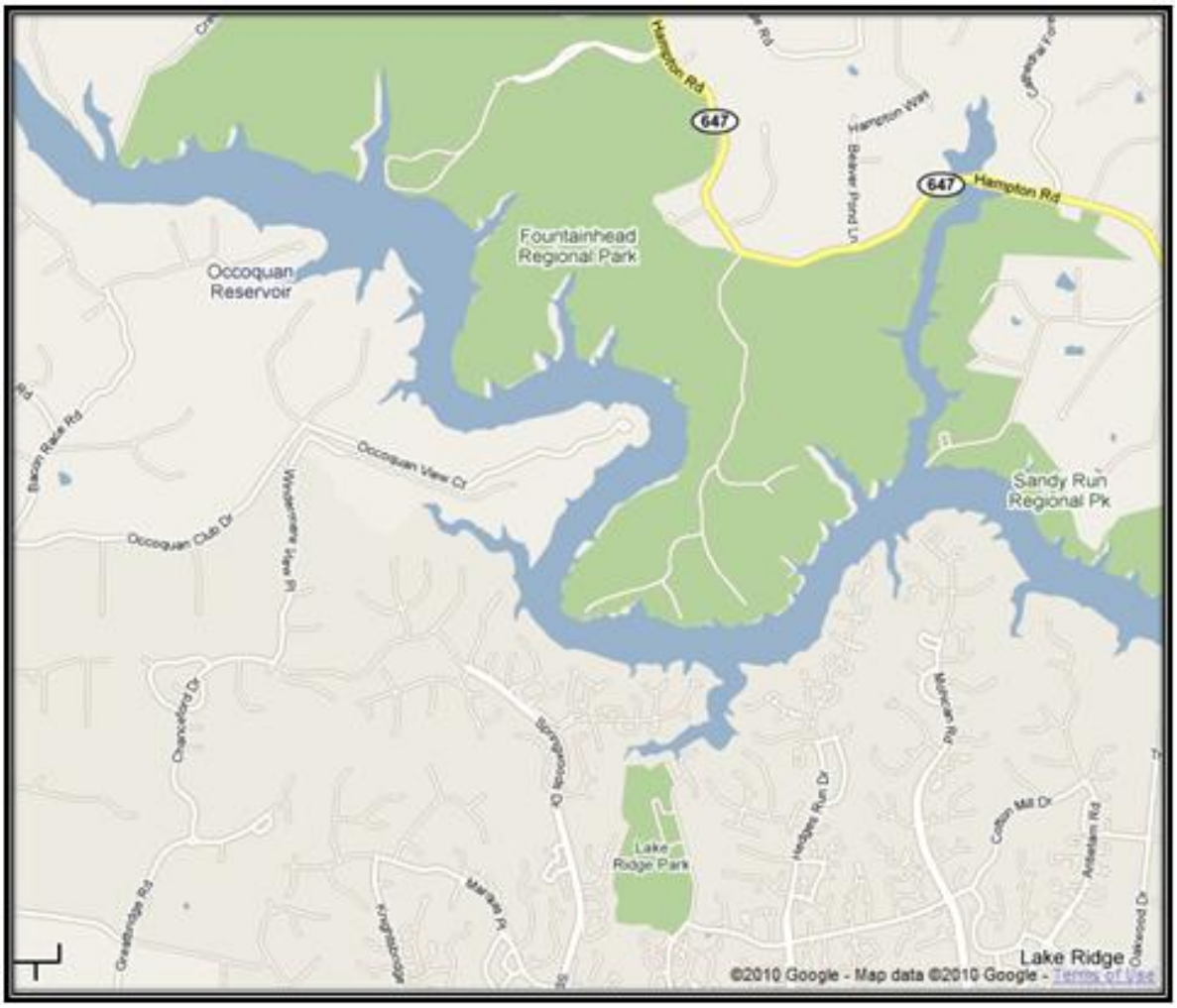


Figure 2. Location of Occoquan Reservoir (Source: OWML).

The *Occoquan Policy*, which is formally known as *A Policy for Wastewater Treatment and Water Quality Management in the Occoquan Watershed*, was adopted by the Virginia State Water Control Board in 1971 and was revised in 1981 and 1991 (VSWCB 1991). The *Policy* mandated the construction of a high performance wastewater reclamation facility to allow highly treated wastewater to serve as a supplement to the drinking water supply. A key element of the *Policy* was the establishment of a monitoring program to provide unbiased water quality data to all stakeholders. As required by the *Policy,* the Upper Occoquan Sewage Authority (UOSA) was placed in

operation in 1978. UOSA has recently completed an expansion to a treatment capacity of 54 MGD. Since 1972, when the Occoquan Watershed Monitoring Program (OWMP) was established, the program has conducted intensive water quality monitoring to ensure that reclaimed water plants achieve desired performance. The OWMP is also responsible to establish a water quality monitoring program for the purpose of continuous recording of water quality (Grizzard 2005).

Occoquan Watershed Monitoring Laboratory (OWML)

In order to fulfill the responsibilities of OWMP, it is necessary to establish an independent organization for the purpose of water quality monitoring and evaluation. The Occoquan Watershed Monitoring Laboratory (OWML) was established by the Virginia Tech Department of Civil and Environmental Engineering in 1972, and has continued its monitoring efforts since that time. Laboratory staff have done comprehensive studies of water quality and the effects of the reclaimed water discharges into the reservoir. Since its founding, the OWML has installed or operated more than one hundred monitoring stations at various locations for the purpose of collecting meteorological data, hydrologic data, synthetic organic compound (SOC) data, reservoir storage data, and water quality data (Grizzard 2005). During over 30 years of operation, OWML has collected millions of data observations, distributed as shown in Table 3, which have played a key role in Occoquan Reservoir Watershed Management.

Table 3. Summary of Observations from Selected Stations (OWML 2009).

Station Name	Number of Observations	Station Location and/or Description
AIR	456593	Airlie station
BF	752308	Balls Ford Rd
BR02	11211	South Run, Fauquier County
BR03	11071	South Run, Prince William Co.
BR04	9679	North Fork Creek, Prince William Co.,
BR05	8460	Unnamed Stream, Prince William Co.,
BR06	8155	Unnamed Stream, Prince William Co.,
BR07	11235	South Run, Fauquier Co.,
BR08	87	Lake Manassas Streams, Prince William Co.
CE	449826	Clifton Elem. School

CHRE	428680	C.Hunter Ritchie Elem. School
CPK	451368	Crockette Park
CRW	420561	Cedar Run Wetlands
Dulles	3987	Dulles International Airport
EVR	432305	Evergreen Fire Dept.
FOP	444848	Fair Oaks Police Dept.
KBR	268194	Kings Broole
LJ	848321	Lack Jackson Dam
LM	801174	Lake Manassas Water Treatment Plant
LM01	36661	Lake Manassas, western Prince William County
LM02	31176	Lake Manassas, western Prince William County
LM03	25920	Lake Manassas, western Prince William County
LM04	31055	Lake Manassas, western Prince William County
LM05	25372	Lake Manassas, western Prince William County
LM06	23057	Lake Manassas, western Prince William County
LM07	19402	Lake Manassas, western Prince William County
LM08	21244	Lake Manassas, western Prince William County
LMDAM	399230	Lake Manassas Dam
LORT	6542	Lorton Water Treatment Plant
OWML	896475	Occoquan Watershed Monitoring Laboratory
PC40	644	S. Fork Quantico Creek , PW Integrator
PC50	1176	Powells Creek at Spriggs Rd., PW Integrator
PC60	1293	Neabsco Creek at Delany Rd., PW Integrator
PC70	483	Cow Branch at Telegraph Rd., PW NPDES
PC80	296	Rocky Run at Worthington Dr., PW NPDES
PR01	44236	Chain Bridge, VA 123, Potomac River, Arlington Co. VA.
PWLF	751841	Prince William County Regional Landfill
RE01	23542	Occoquan Reservoir, Occoquan Dam
RE02	89780	Occoquan Reservoir, second power line
RE05	30258	Occoquan Reservoir, below Sandy Run
RE10	28502	Occoquan Reservoir, Jacob’s Rock
RE15	83506	Occoquan Reservoir, Ryan’s Dam
RE20	17056	Occoquan Reservoir, below Occoquan Creek
RE25	18377	Occoquan Reservoir, above confluence with Bull Run
RE30	71836	Occoquan Reservoir, Bull Run Marina
RE35	56422	Occoquan Reservoir, Ravenwood Bridge
ST01	857574	Occoquan Dam, northeastern Prince William County
ST05	14031	Hooes Run near Occoquan Dam, near Occoquan Dam
ST10	238905	Lake Jackson, Occoquan River near Manassas
ST20	29496	Cedar Run near Aden
ST25	515823	Cedar Run Near Aden

ST30	601996	Broad Run near Bristow, central Prince William County
ST40	186806	Yates Ford, Bull Run near Clifton
ST45	607072	Yorkshire, southwest Fairfax County
ST50	527691	Cub Run near Bull Run, southwest Fairfax County
ST60	472742	Bull Run, northern Prince William County
ST70	545573	Broad Run at Buckland, northern Prince William County
TOA5	2929608	UOSA Weather Station

Taking stream-gage stations (Figure 3) as an example, a network of ten automated stream gaging and sampling stations on the major tributaries to the Occoquan Reservoir are operated by OWML. The locations of these stations are shown in Figure 3. All OWML stream-gage stations were assessed for their suitability for the dual functions of stream gaging and streamflow sampling. Sites with a suitable natural control were selected so as to simplify the development of discharge rating relationships. Automated stream monitoring stations (Figure 4) in the Occoquan Watershed are equipped with

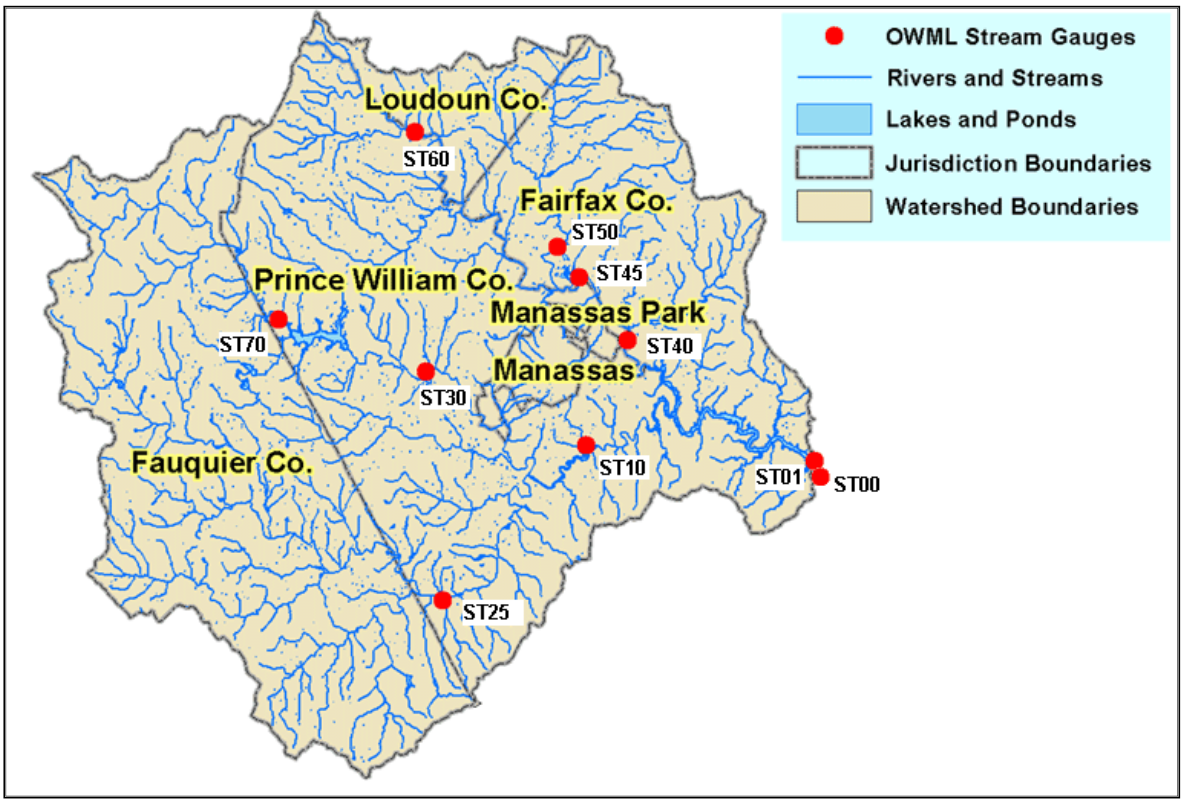


Figure 3. Stream Gage Stations of Occoquan Watershed (Source: OWML).

conventional stream gaging instrumentation which is interfaced with automated water samplers and refrigerated storage. On-site microcomputers, which communicate directly with a central server located at OWML, are used to supervise the stream gages. The on-site microcomputers are programmed to begin collecting flow weighted composite samples at the onset of a storm event. Along with storm event sampling, ambient flow conditions are routinely sampled on a time schedule (Kou et al. 2005).

Besides its continuously observed data collection, OWML has also developed an Occoquan watershed-reservoir model, which consists of a complexly-linked system of six HSPF and two CE-QUAL-W2 sub-models. The Occoquan model has been applied to simulate hydrology and water quality activities in the Occoquan reservoirs and its associated drainage areas. Both models have been fully calibrated for the period of 1988-1992 (Grizzard 2005). The water quality constituents include temperature, dissolved oxygen, ammonium nitrogen, oxidized nitrogen, orthophosphate phosphorus, and algae. The spatial and temporal distribution of hydrology processes, nutrient detachment and transport, stream temperature and dissolved oxygen were well-reproduced by HSPF sub-models. The CE-QUAL-W2 sub-models were calibrated to capture the water budgets, hydrodynamics, temperature, temporal and spatial distribution of dissolved oxygen, ammonium nitrogen, oxidized nitrogen, orthophosphate phosphorus, and algae in Lake Manassas and the Occoquan Reservoir (Xu et al. 2007).

Figure 4. Monitoring stations ST01 (Source: OWML).

As mentioned above, OWML has developed a comprehensive hydrology and water quality database and a complexly-linked water quality computer model. This unique

combination is well-suited to the development of a web-based analysis system that will be a viable component of watershed and reservoir management. Also, because of the water reuse program in the watershed, stakeholder access through a web-based system will improve involvement and promote information dissemination.

A web-based analysis system may be used to enhance the accessibility of available data to stakeholders and others interested in learning from the system. The system should have the ability to retrieve, download, and analyze the observed data, and should also help users to understand and manage watershed issues (Chen et al. 2004; Regmi 2002). Such a system can also be very effective in a situation like the Occoquan Watershed which has various counties and cities as primary stakeholders, and, as a result, a more challenging data-sharing environment (Kou et al. 2005).

4. Database Development

Since its establishment in 1972, OWML has collected, processed, and managed a large amount of spatially- and temporally-distributed water quality data in support of its major mission. In the nearly 40 years of its existence, OWML has collected data of various kinds on more than one hundred monitoring stations located principally around the Washington, D.C. National Capital Region (NCR). These stations comprise a temporally and spatially extensive database of meteorology, hydrology and water quality in the Occoquan Basin and other nearby locations in the NCR. Building a large, comprehensive database which includes all meteorology, hydrologic and water quality data is essential for the development of an effective web-based data analysis system, and requires a sophisticated relational database management system (RDBMS) to efficiently store and retrieve information. Microsoft SQL Server Database Management System (MS SQL DBMS)(Microsoft 2008), which is comprised of a database engine, analysis services and reporting services, is suitable for such applications where large, scalable and secure data storage is required, and further, where the ability to enforce data validation, allow multiple simultaneous network usages, and periodic backup is necessary (Chaudhuri et al. 1997).

4.1 Data Collection and Modification

OWML has been observing and collecting data since the start of its monitoring and research programs in the 1970s. Up to the present time, a large amount of data has been collected which includes meteorology data, hydrologic data and water quality data. Meteorological data includes solar radiation, infrared radiation, air temperature, wind speed, and is automatically collected and recorded at weather stations, and generally recorded at ten minute time intervals. Hydrologic data normally include rainfall and stream discharge. Rain data are automatically collected from sixteen rain gage stations at a time interval of ten minutes. Stream discharge data area observed at stations that are located along streams and rivers. The field flow data has a variable reporting interval that may range from a few minutes to an hour. The water quality datasets consist mainly of concentrations of a variety of constituents measured by field,

laboratory, or direct sensing methods (e.g., alkalinity, total phosphorus, conductance, respectively).

The historical data collected by OWML are currently stored in several separate flat file databases. Although the flat file format is relatively straightforward and easy to comprehend, it was not intended for use as the storage method for large datasets. Also, separate datasets result in some inconvenience in that each one has to be individually managed. In addition, retrievals or analysis of data between flat files is unduly cumbersome. In order to create a comprehensive database which includes all data sets managed by OWML, the historical data were examined carefully, and the different data types considered in beginning of design a new comprehensive database.

4.2 Database Design

The relational model was selected for the comprehensive database designed for this project. A relational database is one in which data are stored in tables that are related to one another by some common fields which are set as primary and/or foreign keys. The *Primary Key* of a table is a column or group of columns whose value are different in every row of the table. The *Foreign key* of a table is a column or group of columns which contains values that match the primary key values in the same or another table (Petkovic 2008). The creation of relationships between tables using key fields allows the enforcement of referential integrity. Referential integrity prohibits the data manager from entering records into a sub-table containing a foreign key for which there is not an associated primary key in the master table (Carleton et al. 2005).

4.2.1 Database Design Principles

Design is a very important step in constructing a new database. If the database is designed and created merely by intuition and without careful planning, the result will most likely not meet the user's long-term requirements. A poor database design can also cause superfluous data redundancy which is a data organization issue that allows the unnecessary duplication of data (Simovici et al. 1995). Minimizing data redundancy

is also crucial for database design. Normalization of data (Dutka et al. 1989), which is a process to eliminate the dependencies between the columns of a table, is a necessary technique to avoid data redundancy.

The database developed for this research project is comprised of multiple tables which include one master table (parent table) and multiple sub-tables (child tables). The tables in a relational database may contain fields and records. Fields show the different types of stored information; records, on the other hand, are the items (data) of the database. For example, in the OWML watershed monitoring database master table, the four fields might be defined as: Time, Parameter, Value and Unit. Information from one observed data record would be stored in one record in this database. The stored record allows users to quickly access the information by any field (when and what parameter has been observed, and value and unit of that parameter). The tables of a database can typically be considered conceptually as a spreadsheet in which the columns correspond to the fields and rows correspond to records.

Considering that a large number of records will be stored in the watershed monitoring database, and further, that each record may require dozens of fields to fully characterize the data, it is necessary and important to break the numerous records into smaller, more easily manageable tables linked by keys (primary/foreign key). For example, if 100,000 water quality readings are collected at a station located at “Occoquan Reservoir Dam”, it would be wasteful to repeat this location description name for each of the readings. A station ID such as “RE01” is used instead and a sub-table listing corresponding information for each station such as the location name for all stations is created. This is a key feature of a relational database: various tables (master table and sub tables) are used to efficiently store the data and relationships are created to link the tables by keys (primary key and/or foreign key).

As defined by Petkovic (2008), “Normalization of data is a process during which the existing tables of a database are tested to find certain dependencies between the columns of a table.” The following rules have been followed for data normalization

during the database design process for this project: (1) duplicate groups need to be eliminated, (2) redundant data ought to be minimized and/or eliminated, (3) columns which are not dependent on a key field need to be eliminated, and (4) relationships between the tables must be simplified. Besides rules of data normalization, the following design procedures were followed: (1) define the categories (tables) of data OWML collected, (2) determine the types of information (fields) within the categories, and (3) create the relationships between the multiple tables using key and/or index fields. Overall, the goals in designing the OWML monitoring database were to provide tables and fields to completely and accurately describe onsite and laboratory watershed data collected, while keeping the design as simple as possible to minimize the effort needed to maintain the database.

4.2.2 Database Structure and Relationships

The database structure and relationships were developed following the three design procedures mentioned above, and based closely upon the needs of OWML programs.

The first procedure was to define the categories of watershed data collected. As previously noted, different types of data are collected, including meteorology data, hydrologic data and water quality data. In detail, those data include weather data, rain data, stream flow data, reservoir pool elevation data, and analytical results. Analytical results include directly sensed measurements (e.g., turbidity analysis), field analyses (e.g., alkalinity by titration), and laboratory analyses (e.g., total phosphorus by automated colorimetry). By applying the data normalization rules and the first design procedure to the different types of data collected, and based on the sampling type and sampling frequency, the data were divided into three categories: (1) reference data, (2) automatically measured data, and (3) manually (or semi-automatically) measured data.

Reference data tables contain the reference information which is stored separated from the observed data; it is recorded only once and referred to by a unique ID number. Reference data, such as station information and parameter information, normally do not change very often. Thus, separating these data into different tables can be useful in

minimizing the database size, thereby improving its query speed and minimizing data redundancy.

Data under the automatically measured category, including weather data, rain data, pool data, and stream flow data are measured and stored and/or transmitted automatically, albeit with differing time intervals. For example, weather data are normally measured every 10 minutes, rain data are measured at 10 minutes intervals, whereas stream flow data may be measured as frequently as 1 minute, or at normal intervals ranging from 15 to 60 minutes. Finally, pool elevation data are also continually measured and recorded.

Another data category includes those measurements made automatically in the field, manually in the field, manually in the laboratory, or using automated procedures in the laboratory. Given OWML data needs, all the foregoing were lumped into a single category termed *manually measured*. Two sub-categories were created under manually measured data depending on whether depth of water is an important factor or not.

Table 4. List and description of OWML database reference tables.

Table Name	Description
ParameterCodes	Information about parameters that may include parameter names, units, test methods etc.
Matrix	Information about sampling Matrix
Stations	Information about observation stations including station ID, name and location, etc.
SampleQualityCodes	Information about the quality of samples
ResultCodes	Information about the measure and/or lab-test result

The simple database structure shown in Figure 5 was generated based on the above classification criteria. The tables comprising the database were separated into two basic types: reference tables and data tables. The reference tables are shown in Table 4, and include: ParameterCodes table, Matrix table, Stations table, SampleQualityCodes table

and ResultCodes table, which together hold all the information related to or pertaining to a particular idea or object. For example, the stations table contains information related to each monitoring station including station ID, station name, station location, etc. In contrast, data tables, which are listed in Table 5, are used to store observed and/or collected watershed data. Using reservoir pool elevation as an example, each time a new observation is made, a new record (elevation) is added to the MasterData table. The MasterData table has relationships with each of the other reference tables and data tables in which unique primary and/or foreign keys serve as connections. These relationships will be addressed in the following section.

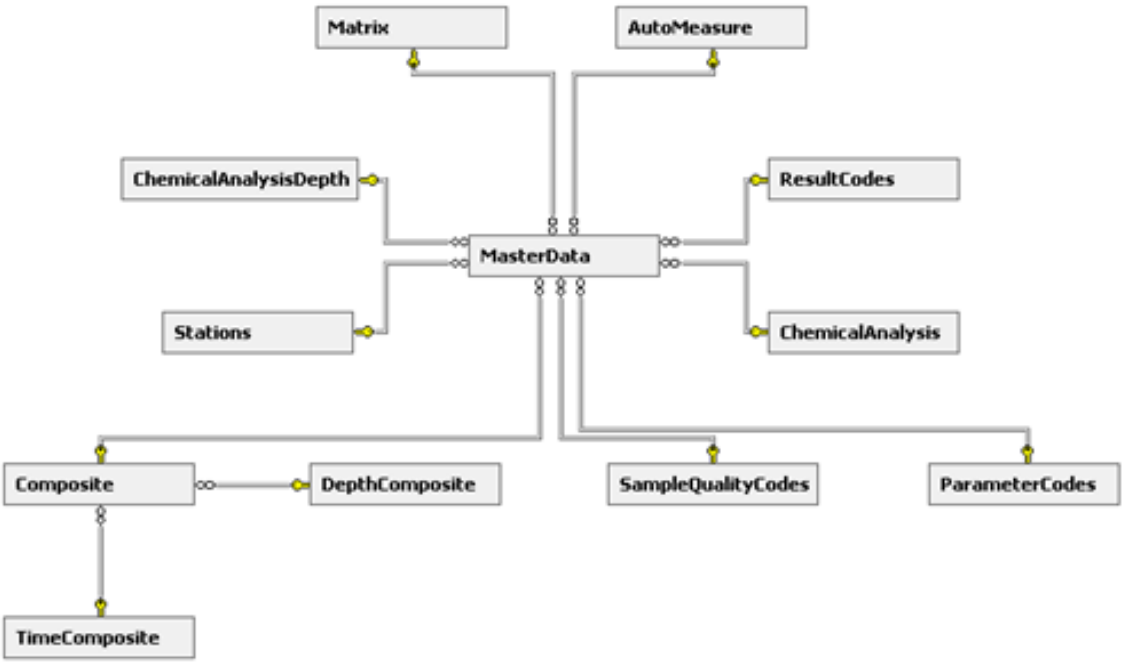


Figure 5. Simplified Database Structure and Relationships.

Table 5. List and description of OWML database data tables.

Table Name	Description
MasterData	Store all observed data(value) and corresponding reference information
AutoMeasure	Store corresponding reference information of automatically measured data
ChemicalAnalysis	Store wet chemistry data
ChemicalAnalysisDepth	Store wet chemistry data which has depth profile property
Composite	Store data for composite analysis
DepthComposite	Store depth data for depth composite analysis
TimeComposite	Store time data for time composite analysis

Once the simplified database structure and relationships are generated, the second procedure, determining the types of information within each category, needs to be finished in order to generate a detailed database structure. In order to determine the fields of each table, it is necessary to analyze the properties of each table. i.e., what type of table it is and what information will be stored. Clearly, answering these questions can help determine the fields of each table. For example, when referring to the information of a monitoring station, the station ID, name and location are the most essential information. An accurate spatial coordinate set for the station is also important. Therefore, coordinate fields should be added into the Stations table. The fields comprising each table are outlined in Table 6. Records in both the reference and data tables are uniquely identified by an ID field for each table (e.g. StationID), which is used by the database to keep track of new and updated records during database synchronization. The ID fields use globally unique identification numbers (GUID) that are uniquely and incrementally generated by the database engine. The ID fields are also used as keys (primary/foreign keys) to create the relationships of the database. An example is that records in the MasterData table are related to the information in the reference Stations table through a unique integer number (StationID) as shown in Table 6. Once fields of each table have been assigned, the detailed database structure and relationships, as shown Figure 6 were established.

Table 6. List of field names and descriptions for OWML database tables.

Field Name	Data Type	Description/purpose
AutoMeasure Table		
AutoMeasureID	bigint[1]	Identification number for auto measurement data
CreateDate	smalldatetime[2]	The date of data filling
CreateInit	varchar(6)[3]	Operator Initial
ValidateDate	smalldatetime	The date when the result available to publish
ValidateInit	varchar(6)	The initials of administrator
ChemicalAnalysis Table		
ChemID	bigint	Identification number for Chemical analysis data
LabID_CA	bigint	Lab test ID number
CreateDate	smalldatetime	The date of data filling
CreateInit	varchar(6)	Operator Initial
ValidateDate	smalldatetime	The date when the result available to publish
ValidateInit	varchar(6)	The initials of administrator
ChemicalAnalysis-Depth Table		
ChemDepthID	bigint	Identification number for Chemical depth analysis
DepthFt	numeric(6,2)[4]	Depth for sampling (feet)
WaterSurface	bit[5]	Indicates the sample is at top of water body
WaterBottom	bit	Indicates the sample is at bottom of water body
LabID_CAD	bigint	Lab test ID number
CreateDate	smalldatetime	The date of data filling

[1] Bigint: Represents interger values that can be stored in 8 bytes. The range of values is -2^{63} to $2^{63}-1$.
[2] Smalldatetime: Represents a date and time, with each value being stored as an integer value in 4 bytes or 2 bytes.
[3] Varchar[(n)]: Desctibes a variable-length string of single-byte characters($0<n<=8000$).
[4] Numeric: Synonym for decimal.
[5] Bit:Used for specifying the Boolean data type with three possible values: False, True and Null.

CreateInit	varchar(6)	Operator Initials
ValidateDate	smalldatetime	The date when the result available to publish
ValidateInit	varchar(6)	The initials of administrator
Composite Table		
CompositeID	bigint	Identification number for Composite sampling analysis
DepthCompositeID	bigint	Identification number for depth composite sampling analysis
TimeCompositeID	bigint	Identification number for time composite sampling analysis
LabID_C	bigint	Lab test ID number
CreateDate	smalldatetime	The date of data filling
CreateInit	varchar(6)	Operator Initial
ValidateDate	smalldatetime	The date when the result available to publish
ValidateInit	varchar(6)	The initials of administrator
Depthcomposite Table		
DepthCompositeID	bigint	Identification number for depth composite sampling analysis
DepthFt_1	float[6]	Depth for sampling (feet)
DepthFt_2	float	Depth for sampling (feet)
MasterData Table		
TrackID	bigint	Auto Increment ID number as a unique identity number
datetime1	datetime	The time of test, measurement, sampling etc.

[6] Float: Represents floating-point values, like Real.

StationID	int[7]	Identification number of station
ParamID	varchar(30)	Identification code of parameter
ValueN	numeric(18,6)	Numeric result value
ValueC	nchar(18)[8]	Character result value
PrecisionDigits	tinyint[9]	The number of digits after decimal
ResultID	varchar(5)	Identification code of result value
SampleQID	char(2)	Identification code of sample quality
MatrixID	char(2)	Identification code of matrix
ChemID	bigint	Identification number for Chemical analysis data
ChemDepthID	bigint	Identification number for Chemical depth analysis
CompositeID	bigint	Identification number for Composite sampling analysis
AutoMeasureID	bigint	Identification number for auto measurement data
Matrix Table		
MatrixID	char(2)[10]	Identification code of matrix (A-air, DF-dry fall, F-fish, S-sediment, W-water, WF-wet fall)
MatrixDescription	varchar(50)	Detail description of matrix indentify code
ParameterCodes Table		
ParamID	varchar(30)	Identification code of parameter (TN1,TN2,TN3)
ParamIdentifierText	varchar(50)	generic easily identifiable name for operator
ParamName	varchar(50)	The name of parameter
Units	varchar(20)	Units for parameter
TestMethod	varchar(100)	standard laboratory or test method

[7] Int: Represents integer values that can be stored in 4 bytes. The range is -2147483648 to 2147483647.
[8] Nchar[(n)]: Store fixed-length strings of Unicode characters.
[9] Tinyint: Represents nonnegative integer values that can be stored in 1 byte. The range is 0 to 255
[10] Char[(n)]: Represents a fixed-length string of single-byte characters, where n is the number of characters inside the string. N<8000

DetectLimit	numeric(18,6)	Minimum detectable result for this test according to standard laboratory or field test
PrecisionDigits-Display	tinyint	How many digit when creating the view (a guide line only)
MinVerifyLimit	numeric(18,6)	Minimum limiting values used to Identification questionable observations outside of analytical test limits
MaxVerifyLimit	numeric(18,6)	Maximum limiting values used to Identification questionable observations outside of analytical test limits
MinCriteria	numeric(18,6)	Minimum criteria values used by screening report to Identification observations outside of regulatory standards
MaxCriteria	numeric(18,6)	Maximum criteria values used by screening report to Identification observations outside of regulatory standards
ValueMultiplierHigh	numeric(6,3)	Number to be multiplied for view creation triggered by MultiplyHighPCode
ValueMultiplierLow	numeric(6,3)	Number to be multiplied for view creation triggered by MultiplyLowPCode
ResultCodes Table		
ResultID	varchar(5)	Identification code of result value
Description	varchar(30)	Description of result Identification code
MultiplyHighPCode	bit	Bit to indicate if valueMultiplierHigh in ParameterCodes table should be multiple by ValueN for view
MultiplyLowPCode	bit	Bit to indicate if valueMultiplierLow in ParameterCodes table should be multiple by ValueN for view

SampleQuality-Codes Table		
SampleQID	char(2)	Identification code of sample quality
SampleQuality-CodeName	varchar(30)	The name of sample quality code
Description	varchar(100)	
Stations		
StationID	int	Identification number of station
StationName	varchar(10)	The name of stations
StationLocation	text[11]	The location of stations
LongitudeD	numeric(7,3)	The Latitude in degree of stations
LatitudeD	numeric(7,3)	The Longitude in degree of stations
ElevationM	numeric(6,2)	The elevation in meter of stations(based on the sea level)
Alive	bit	Indicate whether the station is working or shut down
TimeComposite Table		
TimeCompositeID	bigint	Identification number for time composite sampling analysis
Datetime2	datetime	The end time of sampling

[11] Text: Represents any text data (that is, printable data)

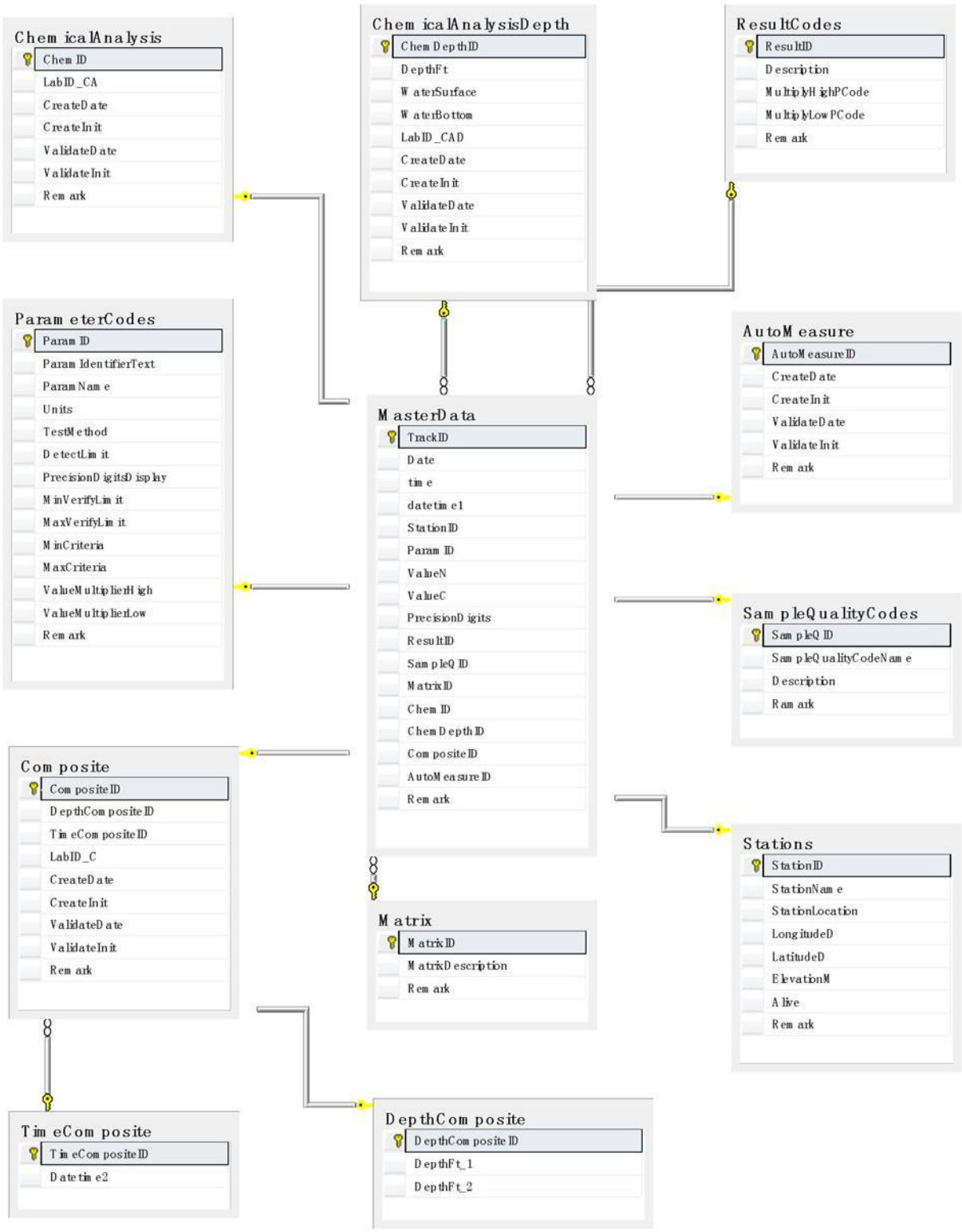


Figure 6. Database Structure and Relationship.

4.3 Data Filling and Database Enhancement

4.3.1 Data filling

In order to insert data into the new comprehensive database, several stored procedures (Appendix A) and stored functions (Appendix B) were developed. The following sections contain descriptions of the data tables as well as data filling details.

Primary Data Table:

Within the current design, the primary table is named MasterData Table, which is described in Table 7. This table contains the entire sampling event. All observed data, including field measured data and lab measured data, are stored in this table. Also, the sampling quality and result quality information for each observation are stored in the same location. Several data filling procedures are used to load the raw data into the new database: Automeasure Data insert, ChemicalAnalysis Data insert, ChemicalAnalysisDepth Data insert and Composite Data insert. The filling procedures and all data manipulating codes were written in SQL Server relational database language which is also called Transact-Structured Query Language (SQL). The SQL language is a computer language designed for managing data in a relational database management system (RDBMS) (Wilton et al. 2005). Using the example of rain data, which is automatically measured, as an example, the procedure used for rain data insert is named “StoredProcedure [dbo].[Rain Insert]”. The following SQL code illustrates how the rain data insert procedure works:

```
USE [OWML_Index]
GO
/****** Object:  StoredProcedure [dbo].[rainInsert]    Script Date: 11/04/2009 22:39:28 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
```

```
-- Author:              <Kumar,Saurav>
-- Create date: <April,28,2009>
-- Description:         <getChemID>
-- ============================================
@initials varchar(5)='wy')
As
begin
BEGIN TRANSACTION;
BEGIN TRY
declare @autoMeasureID bigint
declare @param varchar(15)
declare @sampleQID char(1)
declare @matrixID char(2)
set @param='RAIN'
set @matrixID='A'
set @sampleQID='G'
EXEC  getAutoMeasureID @initialsSupp=@initials,@ret = @autoMeasureID OUTPUT
Insert into
MasterData(datetime1,StationID,ParamID,ValueN,ValueC,PrecisionDigits,ResultID,SampleQID,
MatrixID,ChemID,ChemDepthID,CompositeID,AutoMeasureID,Remark)
select  cast (A+' '+B as datetime),
@stationID,@param,
OWMLWater_AllData.dbo.extractNumber(C,0), C,
OWMLWater_AllData.dbo.getPrecisionDigits(C),
OWMLWater_AllData.dbo.getResultCode(C,0),@sampleQID,@matrixID,0,0,0,@autoMeasureI
D,'Initial rain fill ..Saurav'
 from @tmpTable where ISdate(A+' '+B)=1 and C is not null and (ISNUMERIC(C)=1 or
rtrim(ltrim(C))='')
END TRY
BEGIN CATCH
   SELECT
      ERROR_NUMBER() AS ErrorNumber
      ,ERROR_SEVERITY() AS ErrorSeverity
      ,ERROR_STATE() AS ErrorState
```

```
        ,ERROR_PROCEDURE() AS ErrorProcedure
        ,ERROR_LINE() AS ErrorLine
        ,ERROR_MESSAGE() AS ErrorMessage;
    IF @@TRANCOUNT > 0
        ROLLBACK TRANSACTION;
END CATCH;
IF @@TRANCOUNT > 0
    COMMIT TRANSACTION;
end
```

Similar stored procedures have also been created for inserting other automatically-measured data such as weather data, pool data and stream flow data. Using these stored procedures, the mass data collected by OWML may be easily inserted into the new database without losing data integrity. In the meantime, the possibility of human errors during the data population process is minimized.

Table 7. Description of MasterData Table.

Column Name	Data Type	Description
TrackID	bigint	Auto Increment ID number as a unique identity number
datetime1	datetime	The time of test, measurement, sampling etc.
StationID	int	Identification number of station
ParamID	varchar(30)	Identification code of parameter
ValueN	numeric(18,6)	Numeric result value
ValueC	nchar(18)	Character result value
PrecisionDigits	tinyint	The digits after decimal
ResultID	varchar(5)	Identification code of result value
SampleQID	char(2)	Identification code of sample quality
MatrixID	char(2)	Identification code of matrix
ChemID	bigint	Identification number for Chemical analysis data
ChemDepthID	bigint	Identification number for Chemical depth analysis
CompositeID	bigint	Identification number for Composite sampling analysis
AutoMeasureID	bigint	Identification number for auto measurement data

Secondary Data Tables:

The Secondary data tables of the new database include the AutoMeasure Table, ChemicalAnalysis Table, ChemicalAnalysisDepth Table and Composite Table as shown in Table 8. These tables are generated according to the data type; therefore, each table has its unique primary key (e.g. AutoMeasureID or ChemID) which is used as a relationship link to the MasterData Table. Because observed data are all stored in the MasterData Table, all the secondary data tables mostly contain information about data publication, e.g., when data were inserted and who performed such insertion, or who approved the data before they were ready to be published. It should be noted that the primary key of each secondary data table is a unique, auto-generated identifying number. These ID numbers have been also inserted into MasterData Table using stored procedures such as "StoreProcedure [dbo].[GetChemID]" which get the corresponding ChemID stored in ChemicalAnalysis Table. The following query shows the GetChemID procedure:

```
USE [OWML_Index]
GO
/****** Object:  StoredProcedure [dbo].[GetChemID]    Script Date: 11/04/2009 22:47:46
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:              <Yang,Wei>
-- Create date: <April,28,2009>
-- Description:         <getChemID>
-- =============================================
ALTER PROCEDURE [dbo].[getChemID]
        -- Add the parameters for the stored procedure here
        @initialsSupp varchar(5)='',
```

```
        @labid bigint,
        @ChemID bigint OUTPUT
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;
        Declare @crtDate smalldatetime
        set @crtDate=CAST(getdate() as smalldatetime)
        declare @initialsD varchar(5)

        if @initialsSupp = ''
        begin
                SELECT  @initialsD=cast(value as varchar(5))
                FROM fn_listextendedproperty('initials', 'USER', SYSTEM_USER,default,
default, default, default, default);
        end
        else
        begin
                set @initialsD=@initialsSupp
        end
        insert into ChemicalAnalysis(LabID_CA, CreateDate,CreateInit)values
(@labid,@crtDate,@initialsD)
        select @ChemID=max(ChemID)from ChemicalAnalysis
        where  CreateInit=@initialsD and CreateDate=@crtDate
End
```

The ChemID is the unique ID number used to establish a relationship between ChemicalAnalysis Table and MasterData Table. By using the GetChemID procedure, the ChemID that was generated when a data insertion event occurs may be retrieved

from ChemicalAnalysis Table and inserted into MasterData Table, thereby creating the relationship between the tables.

The IDs of other Secondary Data Tables are generated, retrieved and inserted with a procedure similar to ChemID. The IDs act like a bridge which connects the MasterData Table to other Secondary Data Tables.

Table 8. Description of Secondary Data Tables.

Table /Column Name	Data type	Description
AutoMeasure Table		
AutoMeasureID	bigint	Identification number for auto measurement data
CreateDate	smalldatetime	The date of data filling
CreateInit	varchar(6)	Operator Initial
ValidateDate	smalldatetime	The date when the result available to publish
ValidateInit	varchar(6)	The initials of administrator
ChemicalAnalysis Table		
ChemID	bigint	Identification number for Chemical analysis data
LabID_CA	bigint	The lab test ID number of chemical analysis data
CreateDate	smalldatetime	The date of data filling
CreateInit	varchar(6)	Operator Initials
ValidateDate	smalldatetime	The date when the result available to publish
ValidateInit	varchar(6)	The initials of administrator
ChemicalAnalysisDepth Table		
ChemDepthID	bigint	Identification number for Chemical depth analysis
DepthFt	numeric(6,2)	Depth for sampling (feet)
WaterSurface	bit	Indicates the sample is at top of water body

WaterBottom	bit	Indicates the sample is at bottom of water body
LabID_CAD	bigint	The lab test ID number of chemical analysis depth data
CreateDate	smalldatetime	The date of data filling
CreateInit	varchar(6)	Operator Initials
ValidateDate	smalldatetime	The date when the result available to publish
ValidateInit	varchar(6)	The initials of administrator
Composite Table		
CompositeID	bigint	Identification number for Composite sampling analysis
DepthCompositeID	bigint	Identification number for depth composite sampling analysis
TimeCompositeID	bigint	Identification number for time composite sampling analysis
LabID_C	bigint	The lab test ID number of chemical data
CreateDate	smalldatetime	The date of data filling
CreateInit	varchar(6)	Operator Initials
ValidateDate	smalldatetime	The date when the result available to publish
ValidateInit	varchar(6)	The initials of administrator

Reference data Tables:

As introduced at the beginning of Chapter 2, the reference data tables, as described in Table 9, contain the reference information which is stored separately from the observed data. Such information is recorded only once and is referred to by unique ID numbers. Considering that the reference data are unique and only need to be recorded once, storing these data in a separate table (reference data table) can minimize database redundancy and expedite query processing. Because the volume of reference data is very small compared to primary and secondary data, these data are imported manually using SQL Server Import Wizard and a simple insert query is used to load the data into

the designated tables. Taking the Stations Table as an example, the station data are first imported into SQL server using Import Wizard, and a temporary table named TempStationTable is used to hold the information during the transaction. Each column in the TempStationTable contains a unique property of the monitoring stations. The query below is then used to retrieve these data from TempStationTable and insert them into the target table (Stations Table).

```
Insert into Stations(StationID,StationName,StationLocation,LatitudeD,LongitudeD,
ElevationM,Alive,Remark )
select ColunmA,ColunmB,ColunmC,ColunmD,ColunmE,ColunmF,ColunmG,ColunmH
from TempStation
```

Table 9. Description of Reference Data Tables.

Table Name/Column Name	Data type	Description
Matrix Table		
MatrixID	char(2)	Identification code of matrix (A-air, DF-dry fall, F-fish, S-sediment, W-water, WF-wet fall)
MatrixDescription	varchar(50)	Detail description of matrix indentify code
ParameterCodes Table		
ParamID	varchar(30)	Identification code of parameter
ParamIdentifierText	varchar(50)	generic easily identifiable name for operator
ParamName	varchar(50)	The name of parameter
Units	varchar(20)	Units for parameter
TestMethod	varchar(100)	standard laboratory or test method
DetectLimit	numeric(18,6)	Minimum detectable result for this test according to standard laboratory test or field test method
PrecisionDigitsDisplay	Tinyint	How many digits when creating the view (a guide line only)
MinVerifyLimit	numeric(18,6)	Minimum limiting values used to identify questionable observations outside of analytical test

		limits
MaxVerifyLimit	numeric(18,6)	Maximum limiting values used to identify questionable observations outside of analytical test limits
MinCriteria	numeric(18,6)	Minimum criteria values used by screening report to identify observations outside of regulatory standards
MaxCriteria	numeric(18,6)	Maximum criteria values used by screening report to identify observations outside of regulatory standards
ValueMultiplierHigh	numeric(6,3)	Number to be multiplied for view creation triggered by MultiplyHighPCode
ValueMultiplierLow	numeric(6,3)	Number to be multiplied for view creation triggered by MultiplyLowPCode
ResultCodes Table		
ResultID	varchar(5)	Identify code of result value
Description	varchar(30)	Description of result Identification code
MultiplyHighPCode	Bit	Bit to indicate if valueMultiplierHigh in ParameterCodes table should be multiple by ValueN for view
MultiplyLowPCode	Bit	Bit to indicate if valueMultiplierLow in ParameterCodes table should be multiple by ValueN for view
SampleQualityCodes Table		
SampleQID	char(2)	Identification code of sample quality
SampleQualityCodeName	varchar(30)	The name of sample quality code
Description	varchar(100)	
Stations Table		
StationID	Int	Identification number of station
StationName	varchar(10)	The name of stations

StationLocation	Text	The location of stations
LongitudeD	numeric(7,3)	The Latitude in degree of stations
LatitudeD	numeric(7,3)	The Longitude in degree of stations
ElevationM	numeric(6,2)	The elevation in meter of stations(based on the sea level)
Alive	Bit	Indicates whether the station is working or shut down

4.3.2 Database Index Selection

Database systems generally use indexes to provide fast access to relational data. Like a book index, a database index is a data structure which can improve the speed of data retrieving operations on a database table. Proper selection and tuning of indices is a key for query performance. From a user standpoint, when someone wants to retrieve data from the OWML database, the first thing s/he needs to decide is which location is interested, that is “where”; then s/he needs to think about what time or time range s/he wants the data for, that is “when”; finally, what data is of interest to the customer, that is “what”. Based on this 3W (where-when-what) rule, the indexes selected for this database are ‘StationID’, which identifies the station selected; ‘Datetime1’, which answers when these data have been collected; and ‘ParameterID’, which specifies the kind of data in which the user is interested. As an example, if one wishes to see the reservoir dissolved oxygen (DO) values for January 2009, the query which follows the “where-when-what” rule should be:

```
use OWML_Index
select *
from MasterData
where StationID=36
and datetime1 between 01-01-2009 and 01-31-2009 and ParamID='DO'
```

Once the “StationID”, “Datetime1” and “ParamID” are chosen as the indexes of the database table, a performance evaluation is necessary. This performance comparison should first determine how these indexes should be organized in terms of optimizing the speed of query operations. Second, it should determine whether creating indexes for a database table can indeed improve the speed of query operations. To support this determination, three dummy databases, “OWML”, “OWML_Index” and “OWML_Index_Limited” were created, as shown in Table 10. The database “OWML” has no index in the database table. The database “OWML_Index” has 4 indices which are “Sta_Index”, “Time_Index”, “Param_Index” and “Sta-Time-Param_Index”. “Sta_Index” uses “StationID” as an index key column; “Time_Index” selects “Datetime1” as an index key column and in the same way, “Param_Index” uses “ParamID” as an index key column. However, the “Sta-Time-Param_Index” works in a different manner compared to the last three indexes which used the combination of three columns “StationID”, “Datetime1” and “ParamID” as index key column. Lastly, the database “OWML_Index_Limited” has only one index which is “Sta-Time-Param_Index”.

An example of creating an index using SQL follows:

```
USE [OWML_Index]
GO
/****** Object:  Index [time_sta_para_Index]    Script Date: 09/05/2009 10:58:54 ******/
CREATE NONCLUSTERED INDEX [time_sta_param_Index] ON [dbo].[MasterData]
([datetime1] ASC,    [StationID] ASC,      [ParamID] ASC)
WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON,
FILLFACTOR = 1) ON [PRIMARY]
GO
```

Table 10. Description of Dummy Databases and Indexes.

Dummy Database Name	Index name	Index Key Column Selected
OWML	none	none
OWML_Index_Limited	Sta-Time-Param_Index	StationID+Datetime1+ParamID
OWML_Index	Sta_Index	StationID
	Time_Index	Datetime1
	Param_Index	ParamID
	Sta-Time-Param_Index	StationID+Datetime1+ParamID

After creating three dummy databases with different index properties, data retrieval queries (Appendix C) were implemented to compare the response time of each query, as summarized in Table 11. As may be seen, the response time of the OWML dummy database was significantly longer than other two, which supports the conclusion that creating indices for a database table can improve the speed of query operations. Also, the response time of OWML_Index_Limited dummy database is longer than those for OWML_Index, but the results were relatively close. Therefore, the database with combined multiple indexes was found to run the query faster than the database with a simple index. The same conclusion may be drawn from Figure 7, which shows that the query response time with OWML_Index (red line) was always less than the two other dummy databases. After this performance evaluation, the database OWML_Index, containing a set of combined indexes in database tables, was selected for further development.

Table 11. Different Indexed Database Query Response Time Test Results.

Query Number	OWML Response Time (min)	OWML_Index Response Time (min)	OWML_Index_Limited Response Time (min)
1	5.85	4.08	4.15
2	11.03	3.5	4.75
3	7.03	5.51	5.85
4	4.92	1.75	2.00
5	2.02	1.03	1.02
6	8.58	4.38	8.67
7	2.92	0.23	0.92
8	10.18	0.75	5.95

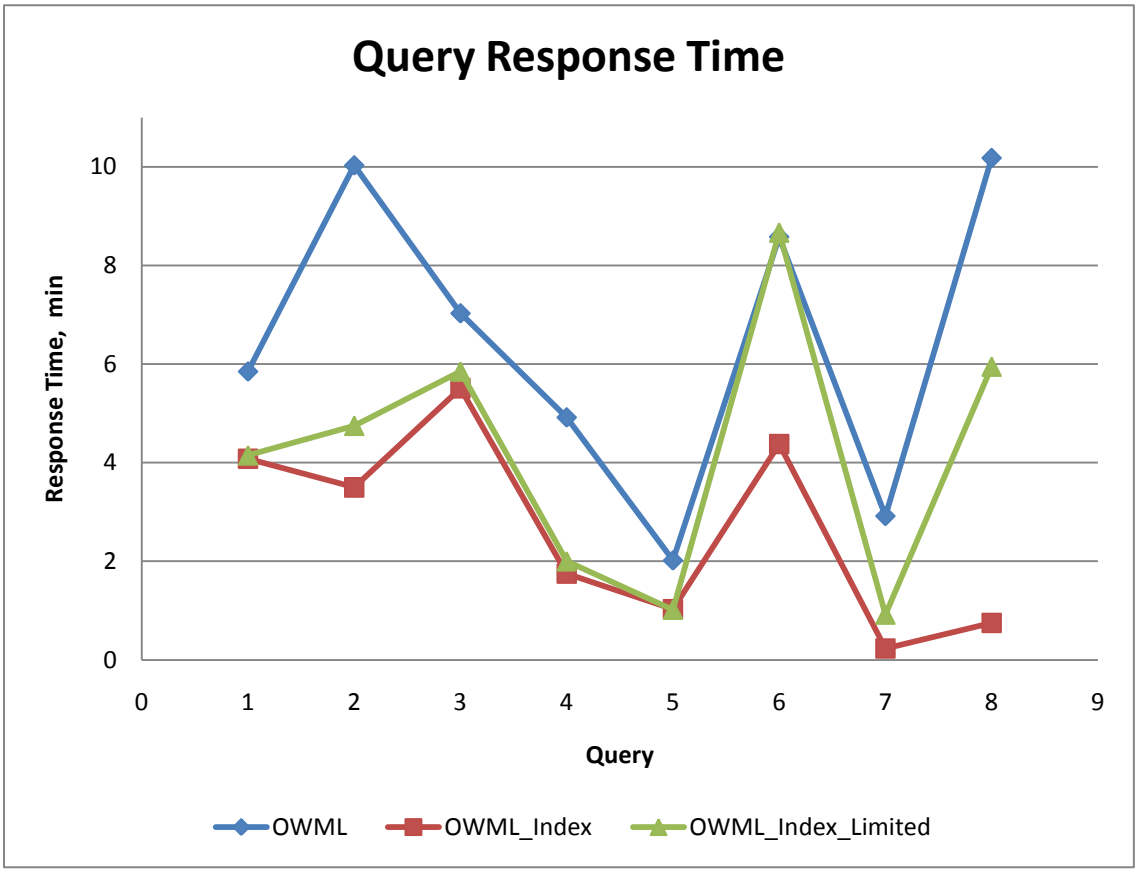


Figure 7. Query Response Time Comparison of Different Indexed Database.

4.3.3 Database Security and Backup Plan

One purpose of developing the OWML database was to support data sharing via a web-based system. Because of the vulnerability of such systems, database security is a very important part of the development. There are two concepts for database security: authentication and authorization.

Authentication requires the system to evaluate whether a user has the legitimate rights to access the database system. The database security specifies the process of validating user credentials to prevent unauthorized users from accessing the database system. Normally, the authentication is confirmed by using a combination of user name and password. Once the system determines that the user name and password are both correct, the system will allow the user to access the database system.

The authorization process occurs after the identity of a user is verified through the authentication process. During the authorization process, the system determines the access level assigned to the user, in other words, the system determines what resources of the database system can be accessed by this particular user. Three authorization levels have been created: administrators, operator and/or managers and normal users, as shown in Table 12.

Table 12. Database Security Levels.

Authorization Level	Groups	limits of authority
Administrators	Company director, Department manager, Project manager	Full access of entire system, including right of authentication and authorization and modification of database
Operator	Department manager, Faculty, Staff, Student	Full access of entire system. Response for database maintenance and management.
Normal users	Shareholder, Public	Partial access of system. Only allow user to retrieve data.

Another essential task in database system administration is database backup. Database backup is the process of copying data from a database or a data file into some backup devices. Normally, there are two backup types: Full Database Backup and Differential Backup. A full database backup obtains the database state at the time when backup is started. During the full backup processing, the database system copies the data as well as all tables with relationships of the database and the corresponding file structures. Therefore, the Full Database Backup has a higher demand on storage space. On the other hand, a differential backup captures only the parts of the database that have changed since the last full backup was completed.

The advantage of differential backup is that differential backup minimizes the time required for backup, because the volume of data to be backed up is relatively smaller than the full database backup. The OWML database was created with two backup plans: full and differential backup. The full backup plan, which is illustrated in Figure 8, was set up to occur weekly on Sunday at 04:00 AM. The differential backup plan, which is illustrated in Figure 9, was set up to occur daily at 01:00AM. Each time a backup is executed, either full or differential, the database system was set to automatically generate a transaction backup log statement in order to make all records traceable.

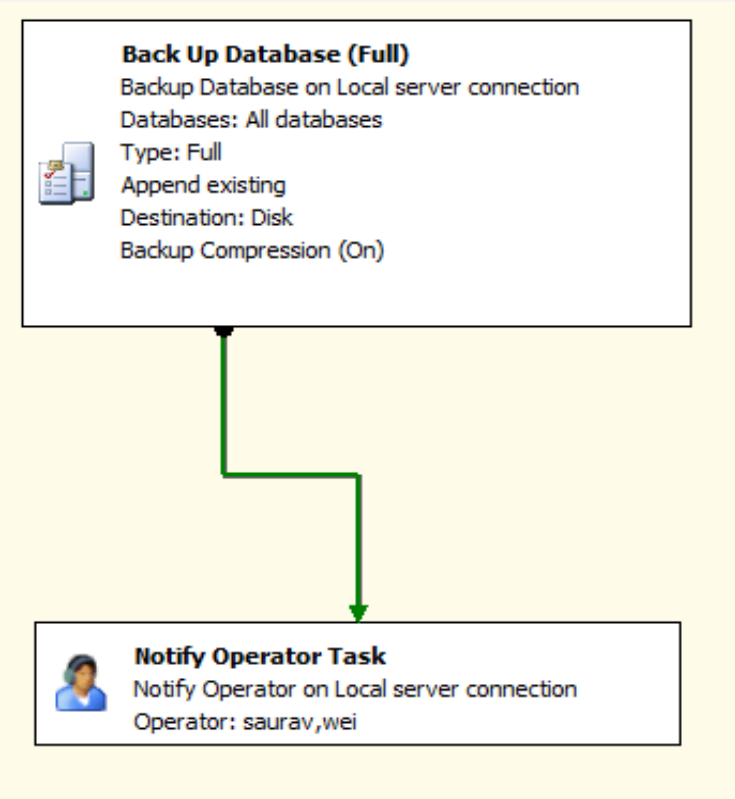


Figure 8. Structure of Full Database Backup Plan.

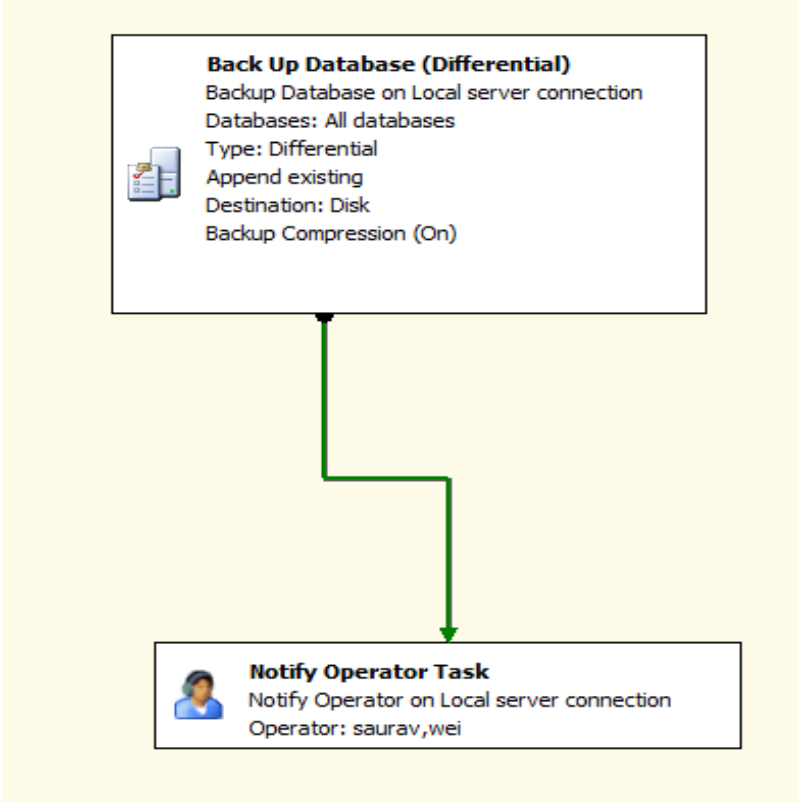


Figure 9. Structure of Differential Database Backup Plan.

4.4 Data Retrieval and Manipulation

Once the data are stored into the OWML database, the data retrieval and manipulation can be accomplished through queries written in Structured Query Language (SQL). Data retrieval is a process of searching a list of desired data files for matches to user query conditions and displaying the data using table views. The basic SQL statements and clauses can be found in any introductory SQL programming book. The following query is a simple SQL statement used to retrieve chemical oxygen demand (COD) data observed on January 2009 from the database, ordered by the date of observation:

```
SELECT [TrackID]
      ,[datetime1]
      ,[StationID]
      ,[ParamID]
```

```
      ,[ValueN]
      ,[PrecisionDigits]
      ,[ResultID]
      ,[SampleQID]
      ,[MatrixID]
      ,[ChemID]
      ,[Remark]
FROM [OWML_Index].[dbo].[MasterData]
WHERE ParamID='COD'and year(datetime1)=2009 and MONTH(datetime1)=01
ORDER BY datetime1
```

First, this query searches all data which have a Parameter ID value of 'COD' from the database; second, it selects only those COD data that match the Date/Time property, which was set as January 2009. Once the query was successfully executed, the MS SQL Server generated a table view showing the user the desired data, as shown in Figure 10.

Results | Messages

	TrackID	datetime1	StationID	ParamID	ValueN	PrecisionDigits	ResultID	SampleQID	MatrixID	ChemID	Remark
1	14361057	2009-01-12 07:53:00.000	57	COD	11.100000	1	OK	G	W	31320	Initial Fill WEI
2	14361061	2009-01-12 09:06:00.000	62	COD	4.600000	1	OK	G	W	31324	Initial Fill WEI
3	14361066	2009-01-12 09:32:00.000	58	COD	9.400000	1	OK	G	W	31329	Initial Fill WEI
4	14361060	2009-01-12 09:58:00.000	61	COD	10.400000	1	OK	G	W	31323	Initial Fill WEI
5	14361059	2009-01-12 10:42:00.000	60	COD	11.400000	1	OK	G	W	31322	Initial Fill WEI
6	14361062	2009-01-12 10:50:00.000	51	COD	13.700000	1	OK	G	W	31325	Initial Fill WEI
7	14361063	2009-01-12 11:11:00.000	52	COD	14.200000	1	OK	G	W	31326	Initial Fill WEI
8	14361058	2009-01-12 11:22:00.000	59	COD	10.200000	1	OK	G	W	31321	Initial Fill WEI
9	14361065	2009-01-12 12:38:00.000	56	COD	16.300000	1	OK	G	W	31328	Initial Fill WEI
10	14361064	2009-01-12 13:35:00.000	54	COD	15.200000	1	OK	G	W	31327	Initial Fill WEI
11	14361067	2009-01-29 08:29:00.000	57	COD	9.000000	1	OK	G	W	31330	Initial Fill WEI
12	14361075	2009-01-29 09:55:00.000	58	COD	9.000000	1	OK	G	W	31338	Initial Fill WEI
13	14361070	2009-01-29 10:56:00.000	61	COD	6.800000	1	OK	G	W	31333	Initial Fill WEI
14	14361074	2009-01-29 11:45:00.000	56	COD	9.000000	1	OK	G	W	31337	Initial Fill WEI
15	14361072	2009-01-29 11:46:00.000	52	COD	15.600000	1	OK	G	W	31335	Initial Fill WEI
16	14361069	2009-01-29 12:03:00.000	60	COD	8.200000	1	OK	G	W	31332	Initial Fill WEI
17	14361071	2009-01-29 12:17:00.000	51	COD	16.200000	1	OK	G	W	31334	Initial Fill WEI
18	14361068	2009-01-29 12:40:00.000	59	COD	9.000000	1	OK	G	W	31331	Initial Fill WEI
19	14361073	2009-01-29 12:57:00.000	54	COD	10.700000	1	OK	G	W	31336	Initial Fill WEI

Query executed successfully. | 8.18.53.30 (10.0 SP1) | wei (52) | OWML_Index | 00:00:00 | 19 rows

Figure 10. The Table View of SQL Database Query Result.

With a properly designed database, SQL Server will allow users to create spreadsheet file formats as well as subsets and data comparisons. The queries shown in the following sections will serve as examples for basic data retrieval operations. Once a user completes a query, it may be saved and easily executed at a later time. It should be noted that when using saved queries, the returns will be highly dependent upon changes made to parameters and other conditions.

Each observed datum is assigned with a corresponding result ID. The result table, which is show in Table 13, is used to store all the result scenarios. Missing data and erroneous data are stored and flagged with predetermined numeric values which are obviously incorrect. For missing data or data detected but not quantified, the designated number -888888 is used, when erroneous data are present, the predetermined number -999999 is used.

Table 13. Summary of Predetermined Result ID.

ResultID	Description
<	Less than the value displayed
>	More than the value displayed
CR	Computed Result
DNQ	Detected but not quantified (Showing -888888 in Value Column)
ER	Error Result (Showing -999999 in Value Column)
IS	Insufficient Sample
ND	Not detected
NF	Not Found
NR	Not Reported
OK	Ok result

4.4.1 Water Quality Results Query

The water quality query places the constituent names as column headers and sample times as row headers. An SQL statement is necessary to achieve this format. The following is an example query which retrieves DO observed on June 5th, 2007 before 10:00 AM from the MasterData Table.

```
select   Datetime1 as 'Date Time',
         ValueN as 'DO(mg/L)',
         Stationname as 'Station Name',
         MasterData.StationID
from     MasterData inner join Stations
on       MasterData.StationID=Stations.StationID
where    ParamID='DO' and
         Datetime1 between '2007-06-05 00:00:00.000' and '2007-06-05 10:00:00.000'
order by Datetime1
```

The results of this query are shown in Table 14, which has “Date Time” and “DO” as column names and the value of sample times as row names. The user may then use the resulting data for any desired purpose, such as calculating average DO value, or plotting in a time series. The visualization system which will be addressed later is an example usage of these result data.

Table 14. Result Table for DO Query.

Date Time	DO(mg/L)	Station Name	StationID
6/5/07 8:22	9.36	RE02	33
6/5/07 8:23	9.22	RE02	33
6/5/07 8:24	9.07	RE02	33
6/5/07 8:26	8.86	RE02	33
6/5/07 8:33	8.40	RE02	33
6/5/07 8:34	1.70	RE02	33
6/5/07 8:35	0.25	RE02	33
6/5/07 8:37	0.12	RE02	33
6/5/07 8:39	0.08	RE02	33
6/5/07 8:40	0.11	RE02	33
6/5/07 8:41	0.44	RE02	33
6/5/07 8:43	1.66	RE02	33
6/5/07 8:45	1.59	RE02	33
6/5/07 8:47	0.18	RE02	33
6/5/07 9:05	8.48	RE05	35
6/5/07 9:07	8.44	RE05	35
6/5/07 9:08	8.34	RE05	35
6/5/07 9:09	8.22	RE05	35
6/5/07 9:10	8.02	RE05	35
6/5/07 9:11	3.65	RE05	35
6/5/07 9:12	0.41	RE05	35
6/5/07 9:13	0.10	RE05	35
6/5/07 9:14	0.10	RE05	35
6/5/07 9:16	0.04	RE05	35
6/5/07 9:17	0.07	RE05	35
6/5/07 9:18	0.07	RE05	35
6/5/07 9:38	9.67	RE10	36
6/5/07 9:39	9.66	RE10	36
6/5/07 9:41	9.61	RE10	36
6/5/07 9:42	9.50	RE10	36
6/5/07 9:43	9.44	RE10	36
6/5/07 9:45	1.32	RE10	36
6/5/07 9:46	0.16	RE10	36
6/5/07 9:47	0.07	RE10	36
6/5/07 9:49	0.35	RE10	36
6/5/07 9:51	0.13	RE10	36
6/5/07 9:45	8.02	PR01	182

4.4.2 Auto Measured Data Query

As noted previously, automatically measured data are collected with very short time intervals, most often every ten minutes. Nevertheless, an automatically measured data query is very similar to the water quality results query shown earlier. This query also produced a result table with constituent names as column headers and observation times as row headers. The following example shows the query which retrieved the incoming solar radiation data on January 1st 2009. The full results, which are shown partially censored in Table 15, consisted of 145 rows.

```
select datetime1 as 'Date Time', ValueN as 'Solar Radiation'
from MasterData
where ParamID='CM3UP_AVG'and datetime1
between '2009-01-01 00:00:00.000'and '2009-01-01 23:59:00.000'
```

Table 15. Result Table for Solar Radiation Query.

Date Time	Solar Radiation
1/1/09 0:00	-4.48
1/1/09 0:10	-5.01
1/1/09 0:20	-4.53
1/1/09 0:30	-5.04
1/1/09 0:40	-4.51
1/1/09 0:50	-5.01
1/1/09 1:00	-4.70
……	……
……	……
1/1/09 22:50	-5.06
1/1/09 23:00	-4.45
1/1/09 23:10	-4.45
1/1/09 23:20	-4.45
1/1/09 23:30	-4.45
1/1/09 23:40	-4.45
1/1/09 23:50	-4.45

Because automatically measured data are collected so frequently, it is often more convenient to use average values, summarized monthly, seasonally, or annually, rather than individual observation results. Producing such a summary requires a somewhat more advanced query such as the example below, where the input data are 10-minute rainfall data:

```
WITH dailyrain_CTE (DateInt,StationID, TotRain)
as (select date1=cast ( datetime1 as int)+1,
StationID, sum(ValueN)
from MasterData
where ParamID='RAIN' and ResultID='OK'
        and YEAR(datetime1)=2008 and  StationID=66
group by StationID, cast ( datetime1 as int)+1 )

select year(CAST(DateInt as datetime)) as Years,MONTH(CAST(DateInt as datetime))
as Months, StationID ,AVG(TotRain) as 'Daily Average Rain' from dailyrain_CTE
group by cube(year(CAST(DateInt as datetime)), MONTH(CAST(DateInt as datetime)),
StationID) order by StationID, years,months
```

The query calculated the value of daily average rainfall by month, and produced the result shown in Table 16. While not excessively complex for an experienced SQL user, the query is sufficiently complicated that it was stored as a function that operators or users can directly access for calculation purposes.

During future operations of the database system, more advanced function queries will be created and stored so as to provide increased functionality and convenience because common queries will not have to be remembered or rewritten.

Table 16. Query Result Table of Daily Average Rain.

Years	Months	Station ID	Daily Average Rain
2008	1-12	66	0.107
2008	1	66	0.044
2008	2	66	0.084
2008	3	66	0.075
2008	4	66	0.119
2008	5	66	0.185
2008	6	66	0.191
2008	7	66	0.117
2008	8	66	0.068
2008	9	66	0.210
2008	10	66	0.029
2008	11	66	0.058
2008	12	66	0.102

5. Development of Web-based Data Visualization and Analysis System (WebDAS)

The OWML Database was built to support easy retrieval of desired data via queries in SQL Server. However, this data retrieval method requires that each user have access to the MS SQL Server. It is impractical to give such access to all users for reasons of security, software availability, and user authentication level, among others. In fact, the system was developed to constrain such full access to only a few administrators or operators. The need still exists, however, to make the database accessible to a variety of other approved users. A data visualization system was determined to be the vehicle for meeting this need, and also to support other watershed management goals.

5.1 System development platform

The 3W (where-what-when) rule, which was used for selecting database indexes, was also strictly followed during the development of WebDAS. Following the 3W rule, the system was divided into two major parts: Mapping Interface (MI) and Data Exploration Interface (DEI). The Mapping Interface (MI) was constructed to help users answer the question of from “where” (which station) the data is desired. The Data Exploration Interface (DEI) was constructed to address “what” and “when” aspects of the desired data. The system was designed to give users the ability to select parameters and time ranges of interest under the Data Exploration Interface. The system was also constructed to provide the ability to create reports which include data outputs suitable for plotting or inclusion in a tabular listing.

WebDAS was developed using Google Maps API (GoogleMap-API 2009) and Microsoft Visual Web Developer (Microsoft 2008). Google Maps API is a free web-based service that can be embedded into the user’s own webpage using JavaScript. The most important feature is that by using Google Maps API, the user is freed from maintaining a map server and does not need to periodically perform map updates, thereby reducing required staff time. Google Maps API provides a Mapping Interface that gives provides the user with an intuitive Occoquan Watershed map page on which stations and

watershed boundaries are located. The Microsoft Visual Web Developer is a web development tool developed by Microsoft that allows developers to create, edit and work with ASP.NET web applications. Visual Web Developer is also widely used for designing data driven websites that use databases to store information which can be visualized during an internet session.

5.2 User Interface

5.2.1 Mapping Interface

Using Google Maps API to develop a Mapping Interface (MI) provides an efficient and cost-effective point of entry into data visualization. While using JavaScript to develop the mapping interface (Appendix D), the first step is to load the Google Maps API and Map Controls. The code below shows how the API and Controls are loaded programmatically:

```
if (GBrowserIsCompatible()) {
    map = new GMap2(document.getElementById("map"));
    //Add map controls
    map.addControl(new GScaleControl());
    map.addMapType(G_PHYSICAL_MAP);
    map.addControl(new GLargeMapControl());
     var mapControl = new GHierarchicalMapTypeControl();
    mapControl.clearRelationships();
    mapControl.addRelationship(G_SATELLITE_MAP, G_HYBRID_MAP, "Labels", true);
    map.addControl(mapControl);
```

This snippet of code first loads the Map onto a Webpage and then adds Map Controls onto the map. The Map Controls, which are illustrated in Figure 11, include zoom in and zoom out control, navigation control and map type control. The latter allows selection of three different map modes: normal map view, satellite map view and terrain map view.

Once the Map and Map Controls have been loaded, the watershed boundary and the center of the map must be specified. The boundary of the Occoquan Watershed was generated by a Keyhole Markup Language (KML) file. KML (Abdulrahman 2008) is an XML-based file format that is used to visualize geographic data in a Map browser such as Google Maps. The GPS for land survey was used to find the center coordinate of Occoquan Watershed at “38.72, -77.5”, which are the latitude and longitude in decimal degrees. Converted to DMC (Degrees, Minutes and Seconds) units, the coordinate are “38° 43' 12.00" N, 77° 29' 60.00 W" in unit of DMC unit. The following code was

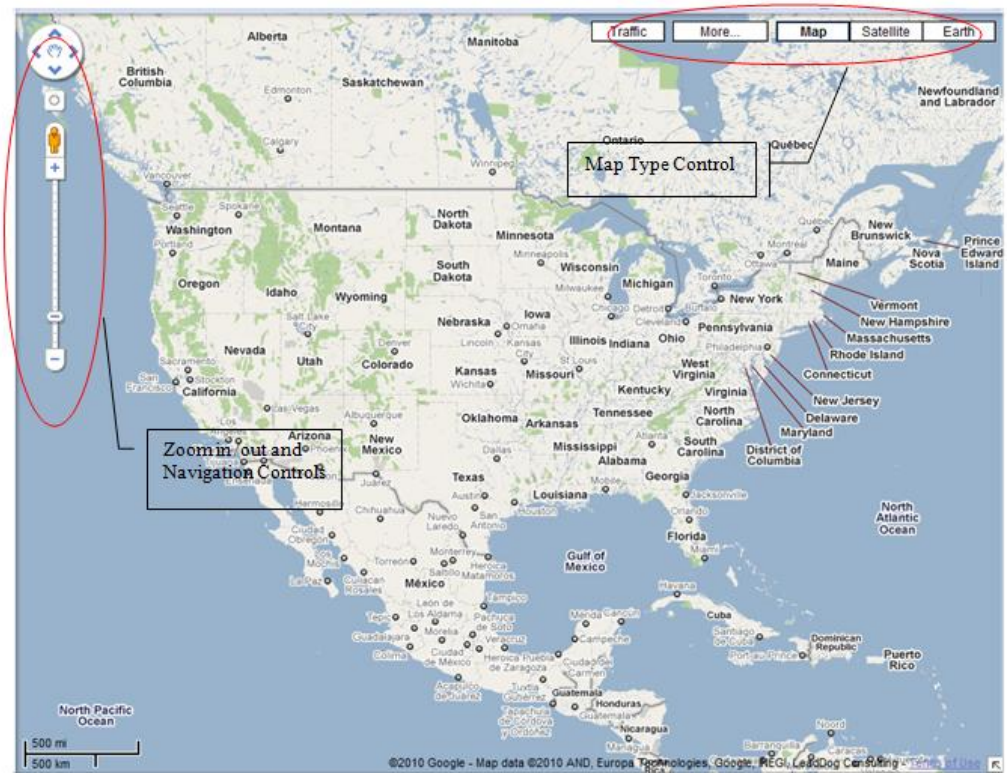


Figure 11. Map and Map Controls.

developed to set the watershed boundary and center of the watershed on the map.

```
//The boundary
 var outline = new GGeoXml("http://www.ecolve.com/owml/OCC_Contour.kmz");
  map.addOverlay(outline);
//Set the center of the map
map.setCenter(new GLatLng(38.72, -77.5), map.getBoundsZoomLevel(bounds),
G_PHYSICAL_MAP);
```

The third step of the map interface design was to add point files to represent monitoring stations within the Occoquan Watershed. Before adding point files, it was necessary to create Marker (point) files. JavaScript has its own function for creating markers. Once created, the point files were added (as stations) on the map by using code such as shown below:

```
function createMarker(point, name, html, icontype, zIdx)
        { var marker = new GMarker(point, gicons[icontype]);
          GEvent.addListener(marker, "click", function()
{ marker.openInfoWindowHtml(html); xgpoint = marker.getPoint(); });
          marker.tooltip = '<div class="sitetip">' + name + '</div>';
  GEvent.addListener(marker, "mouseover", function() {showTooltip(marker);});
  GEvent.addListener(marker, "mouseout", function() {tooltip.style.visibility = "hidden"});
          map.addOverlay(marker);
          return marker;
// add stations as point files onto map
        point = new GPoint(-77.674, 38.755); marker = createMarker(
        point, 'BR02', "<a href='DataExplore.aspx?StID=3'>StationID: 3</a> <br>
Station Name: BR02 <br> Location:South Run Fauquier County Vint Hill Farms Military
Res northern corner<br> ", "mrk_03", 1000)
        map.addOverlay(marker)
         point = new GPoint(-77.666, 38.770); marker = createMarker
        (point, 'BR03', "<a href='DataExplore.aspx?StID=4'>Station ID: 4</a> <br/>
StationName: BR03 <br/> Location:South Run,Prince William Co., at bridge on State
Hwy. 684 (Buckland Mill) <br/> ", "mrk_03", 1000);
        map.addOverlay(marker);
          .
          .
          .
          .
          .
          .
```

Following completion of the above-mentioned procedures, the basic mapping interface, as shown in Figure 12, was finished. As may be seen in the screen extract of Figure 12, the mapping interface contains the boundary of the watershed area and provides access to detailed information on each monitoring station such as station ID number, station name, and station location.

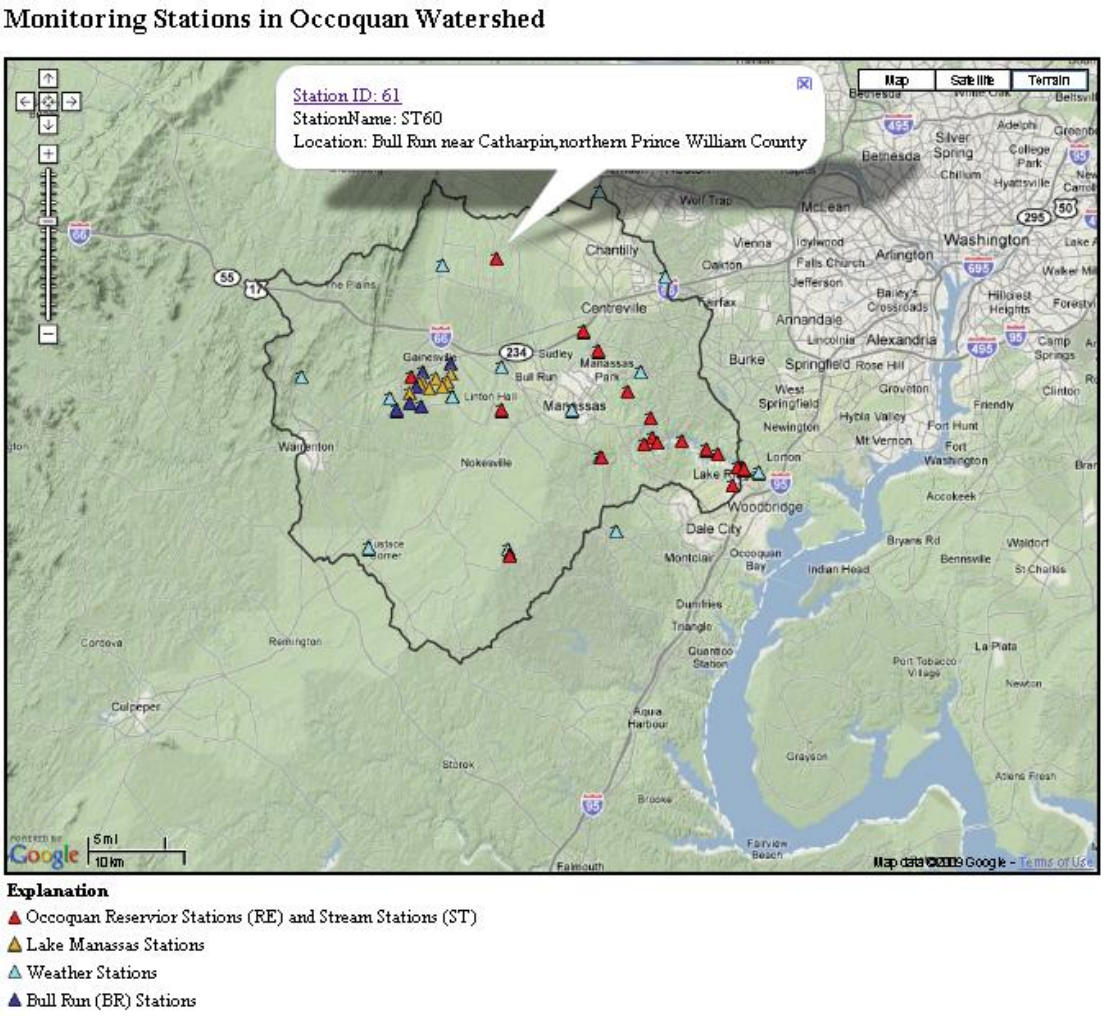


Figure 12. Map Interface.

Users can retrieve the detailed station information by simply clicking any colored triangle icon representing a station. The pop up window shown in Figure 12 opens and automatically adjusts its size according to the information shown inside. This pop up window contains the detailed station information including station ID, station name and station location. Users may also retrieve more information by clicking the “StationID”

which is set as a hyperlink pointing to the data exploration interface webpage, which will be addressed in the following section. The map legend at the bottom of the mapping interface page provides explanations of the different type of stations.

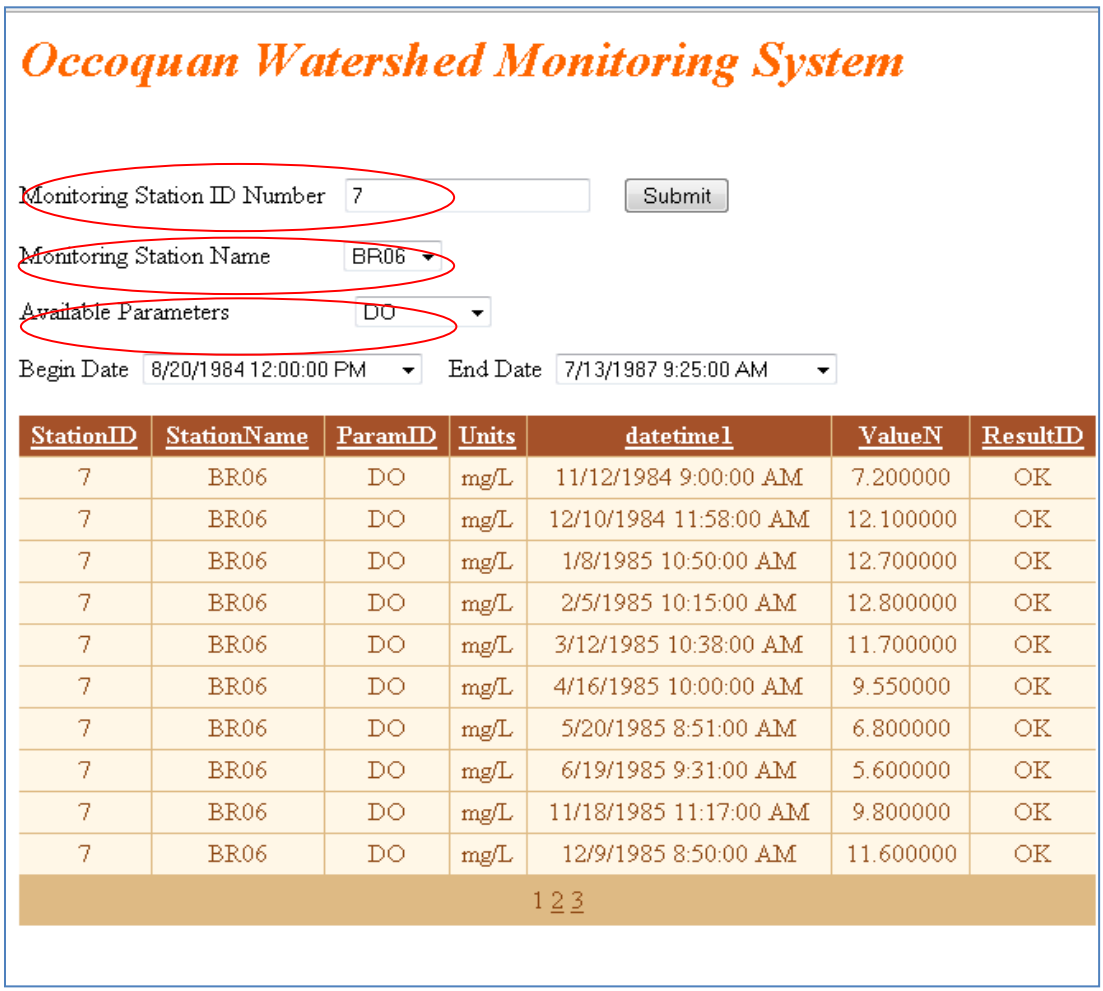
5.2.2 Data Exploration Interface

As mentioned above, the click-station-ID event (Figure 13) redirects the user to another web page called Data Exploration Interface (DEI), which is illustrated in Figure 14. Upon reaching this page, the system passes the selected station ID number to the DEI page. Once the station ID has been passed, the system retrieves the information available. For example, if the user clicks a station icon on the MI page and then clicks the “station ID: 7” on the pop up information window as shown in Figure 13, the system will receive a request to redirect the web page to the DEI page while passing the result of the requested selection query string shown below onto the designated text box on the DEI page. The script (Appendix E) follows:

Figure 13. Map Interface Page with Selected Station.

```
Protected Sub
Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    StationName.Text = Request.QueryString("StID")
End Sub
```

On the DEI page, the number “7” will appear in the text box named Monitoring Station ID Number, as shown in Figure 14. Once the system receives the request that the selected station ID number is equal to 7, the following query will be executed:



Occoquan Watershed Monitoring System

Monitoring Station ID Number 7 Submit

Monitoring Station Name BR06

Available Parameters DO

Begin Date 8/20/1984 12:00:00 PM End Date 7/13/1987 9:25:00 AM

StationID	StationName	ParamID	Units	datetime1	ValueN	ResultID
7	BR06	DO	mg/L	11/12/1984 9:00:00 AM	7.200000	OK
7	BR06	DO	mg/L	12/10/1984 11:58:00 AM	12.100000	OK
7	BR06	DO	mg/L	1/8/1985 10:50:00 AM	12.700000	OK
7	BR06	DO	mg/L	2/5/1985 10:15:00 AM	12.800000	OK
7	BR06	DO	mg/L	3/12/1985 10:38:00 AM	11.700000	OK
7	BR06	DO	mg/L	4/16/1985 10:00:00 AM	9.550000	OK
7	BR06	DO	mg/L	5/20/1985 8:51:00 AM	6.800000	OK
7	BR06	DO	mg/L	6/19/1985 9:31:00 AM	5.600000	OK
7	BR06	DO	mg/L	11/18/1985 11:17:00 AM	9.800000	OK
7	BR06	DO	mg/L	12/9/1985 8:50:00 AM	11.600000	OK

1 2 3

Figure 14. Data Exploration Interface with Selected Data.

```
SELECT [StationName]
FROM [Stations]
WHERE ([StationID] = @StationID)
```

The variable @StationID is assigned with the value of Station ID number selected by the user. In the illustrated case, the @StationID is equal to 7. The result of this query is to get the Monitoring Station Name: BR06, as shown in Figure 14.

Once the station of interest has been selected, the system then launches a request query, as shown below, in order to obtain the available parameters under station BR06. The results are returned as shown in Figure 14.

```
SELECT DISTINCT [ParamID]
FROM [MasterData]
WHERE ([StationID] = @StationID)
ORDERBY ParamID
```

As may be seen in Figure 14, the available parameters are returned in alphabetical order to promote selection convenience. For the purposes of this demonstration, the parameter DO was selected (Figure 14). After the station and available parameters are set, the user may then enter the desired time range for the data query. For security purposes, the system default allows users to select no more than 60 days of data. Users requiring longer periods may be assigned a higher authorization level. The Begin Date and the End Data dropdown list contain the available DateTime of the selected parameter: DO. This DateTime information will be automatically retrieved from the OWML database by executing the following query:

```
SELECT DISTINCT [datetime1]
FROM [MasterData]
WHERE (([StationID] = @StationID) AND ([ParamID] = @ParamID))
ORDER BY [datetime1]
```

Once the user has selected a Begin Date and End Date, the system has received all necessary information to define “Where”, “What” and “When” for the query. The data selection command below will then be executed automatically.

```
SELECT
(MasterData.StationID, MasterData.ParamID, MasterData.datetime1,
MasterData.ValueN, MasterData.ResultID, Stations.StationName,
ParameterCodes.Units ]
FROM  [MasterData]
INNER JOIN
(Stations ON MasterData.StationID = Stations.StationID)
INNER JOIN
 (ParameterCodes ON MasterData.ParamID = ParameterCodes.ParamID)
WHERE (MasterData.StationID = @StationID)
AND (MasterData.ParamID = @ParamID)
AND (MasterData.datetime1= @datetime1)
AND (MasterData.datetime1= @datetime12)
 AND (MasterData.ResultID = @ResultID
```

The successful execution of this above script will generate a table view, illustrated in Figure 15, which contains all the data meeting the conditions set by the selection command. The table contains 10 rows on each page and allows paging and sorting. Users may sort the data by location, time and value. The limitation of this table view is that the user cannot download and/or manipulate the data.

In order to meet the requirements of data download and/or further manipulation, an additional function entitled Report View has been developed, as show in Figure 16. This feature allows the user to generate a report which includes a time series plot and a data table. The selected data chart is plotted with datetime1 as the independent variable,

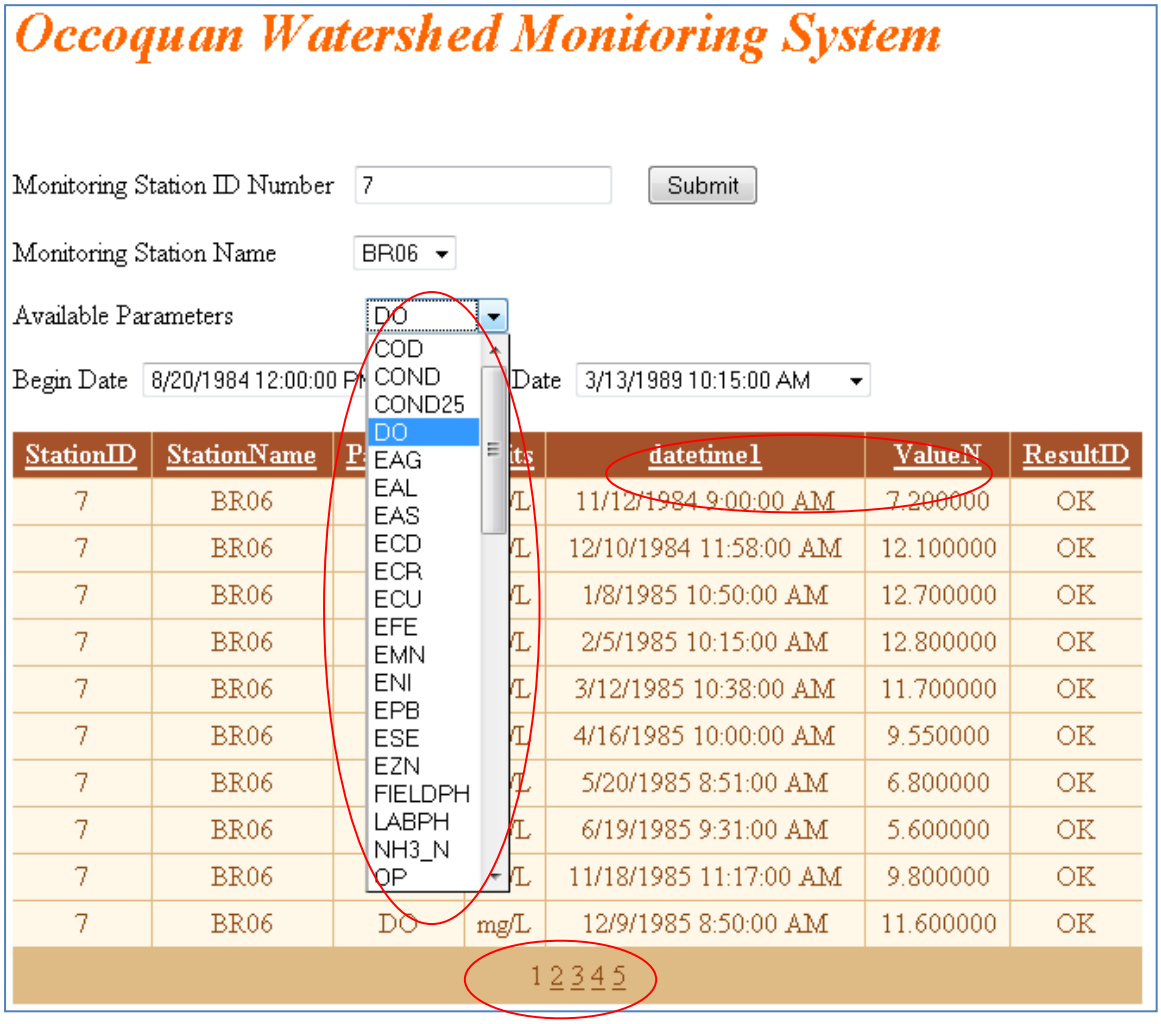


Figure 15. Available Parameters and Table View on DEI.

and the value of selected parameter as the dependent variable. Similar to the table view mentioned above, the data table contains the selected data too. The chart and table may both be exported in either PDF format or Excel format (Jesse et al. 2003), as shown in Figure 17. This feature allows the user to move the retrieved data to a local computer for further data manipulation, thereby conserving server system resources.

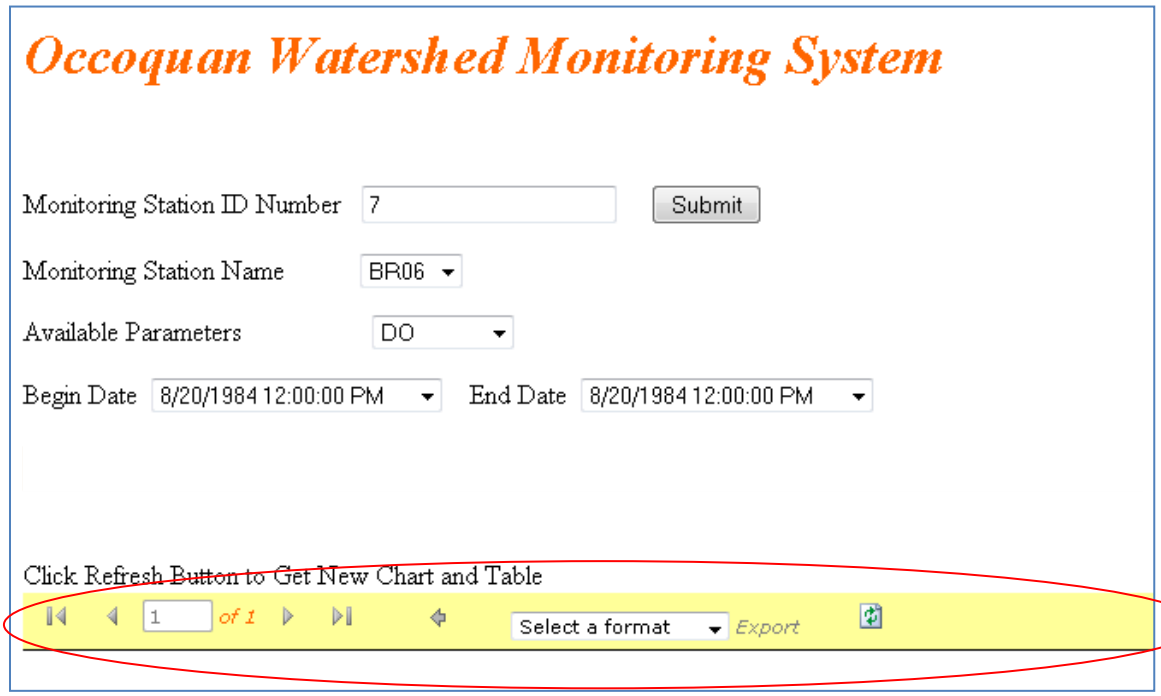


Figure 16. Header of Report View on DEI.

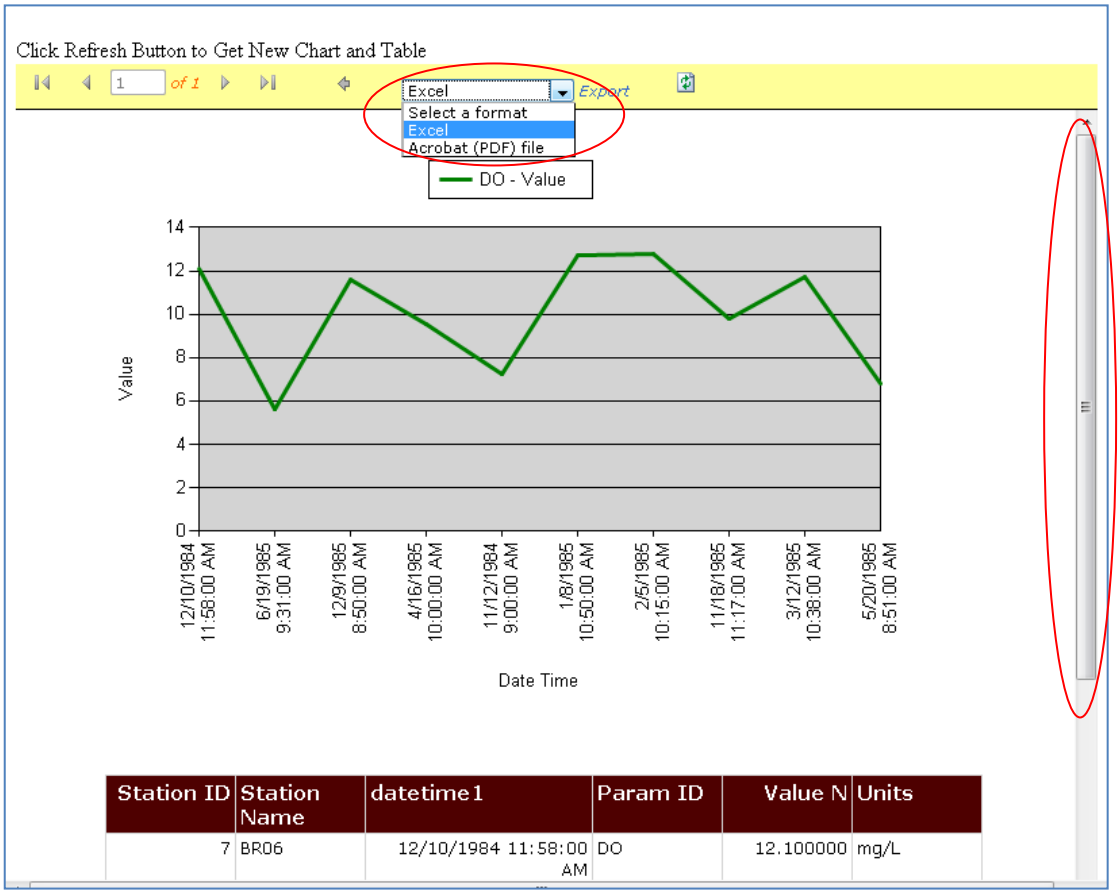


Figure 17. Detailed Report View on DEI.

6. Results

The major outcomes of this research project were: (1) the collection and verification of historical observed data and construction of a comprehensive database for watershed management and analysis; and (2) development of the Web-Based Data Analysis System (WebDAS) that integrates various data visualization capabilities.

Any working database, especially one aimed at watershed management, requires a collection of historical data. Such historical data must be carefully examined to insure that errors have been removed before being loaded into the new database. The processing and verification of the data collection can ensure that:

- all historical data observed by OWML have been collected for the new database design;
- data checking and verification have insured that the data are accurate, unified and ready for loading into the new database.

Completing the database accomplishes more than creating a secure repository. The new structure was designed to enhance access, utility, and analysis. Users may run various types of queries to retrieve and analyze the desired data. With an easy to understand database relationship, which was shown in Figure 5, and an explicit database dictionary, which is shown in Table 17, new developers and/or users can quickly become familiar with the whole database system.

The prototype system WebDAS, which is a data driven website developed using ASP programming, allows users to browse, plot and download desired data. Use of the system does not require the user to install any sophisticated software. The only requirement for an authorized user to access this system is internet connectivity. The choice of a suitable development platform is not only saves the human and material resources while developing this system, but also helps the OWML lower the maintenance expense thereafter.

Table 17. OWML Database Dictionary.

Name	Description
Alive	Indicate whether the station is working or shut down
AutoMeasureID	Identification number for auto measurement data
ChemDepthID	Identification number for Chemical depth analysis
ChemID	Identification number for Chemical analysis data
CompositeID	Identification number for Composite sampling analysis
CreateDate	The date of data filling
CreateInit	Operator Initial
Datetime1	The time of test, measurement, sampling etc.
Datetime2	The end time of sampling
DepthCompositeID	Identification number for depth composite sampling analysis
DepthFt	Depth for sampling (feet)
Description	Description
DetectLimit	Minimum detectable result for this test according to standard laboratory or field test
ElevationM	The elevation in meter of stations(based on the sea level)
LatitudeD	The Longitude in degree of stations
LongitudeD	The Latitude in degree of stations
MatrixDescription	Detail description of matrix indentify code
MatrixID	Identification code of matrix (A-air, DF-dry fall, F-fish, S-sediment, W-water, WF-wet fall)
MaxCriteria	Maximum criteria values used by screening report to identify observations outside of regulatory standards

Table 17, Continued.

MaxVerifyLimit	Maximum limiting values used to identify questionable observations outside of analytical test limits
MinCriteria	Minimum criteria values used by screening report to identify observations outside of regulatory standards
MinVerifyLimit	Minimum limiting values used to identify questionable observations outside of analytical test limits
MultiplyHighPCode	Bit to indicate if valueMultiplierHigh in ParameterCodes table should be multiple by ValueN for view
MultiplyLowPCode	Bit to indicate if valueMultiplierLow in ParameterCodes table should be multiple by ValueN for view
ParamID	Identification code of parameter
ParamIdentifierText	generic easily identifiable name for operator
ParamName	The name of parameter
PrecisionDigits	The number of digits after decimal
PrecisionDigitsDisplay	How many digit when creating the view (a guide line only)
ResultID	Identification code of result value
SampleQID	Identification code of sample quality
SampleQualityCodeName	The name of sample quality code
StationID	Identification number of station
StationLocation	The location of stations

Table 17, Continued.

StationName	The name of stations
TestMethod	standard laboratory or test method
TimeCompositeID	Identification number for time composite sampling analysis
TrackID	Auto Increment ID number as a unique identity number
Units	Units for parameter
ValidateDate	The date when the result available to publish
ValidateInit	The initial of administrator
ValueC	Character result value
ValueMultiplierHigh	Number to be multiplied for view creation triggered by MultiplyHighPCode
ValueMultiplierLow	Number to be multiplied for view creation triggered by MultiplyLowPCode
ValueN	Numeric result value
WaterBottom	Indicate the sample is at bottom of waterbody
WaterSurface	Indicate the sample is at top of waterbody

7. Conclusions and Recommendation

7.1 Summary

During this research, a prototype Web based Data Analysis System was developed for to support Occoquan Watershed management activities. Watershed management is often a critical element of plans to protect waterbodies from pollution. A critical requirement for the success of watershed protection is to coordinate various interest groups into a cooperative team to improve the environment. To achieve this goal of cooperation, a critical step is to help all participants have the ability to access near real time water quality data which can help them have a better understanding of the situation of the water body and how their decisions may affect the environment of the watershed.

A web based data analysis system can be a useful tool for this purpose. A data analysis system requires data inputs, and also needs to transform inputs into outputs that may be used for decision-making processes involving different groups. The Web based Data Analysis System for Occoquan Watershed management (WebDAS) allows watershed researchers to more efficiently access and distribute various types of data. The system can also help watershed management groups share the data in a more timely manner and efficiently across the entire user community. Particularly, the system allows inexperienced users to retrieve and analyze water quality data without prior training, which would be very difficult without such a system due to the limitation of software and hardware availability.

The objectives of this research project were:

- To assess the sources, types, frequencies of collection, and matrices of data collected by OWML;
- To construct a database to efficiently store, retrieve, and make accessible to users the data types identified for watershed management purposes;

- To develop a system for efficiently and economically sharing and visualize data for a range of users and user communities.

This thesis first describes the background of this research project, study objective and the methods used to achieve the study objectives.

The second part reviewed the literature on two related fields. The first field was the current research in the design of databases to support watershed management activities, including the software environments and tools that have been used, and the various design concepts that have been considered. The second topic was the application of web-based technology on sharing, visualizing and managing data. A brief introduction of web based decision support systems was also included.

Following the literature review, a brief introduction of the Occoquan Watershed Monitoring Laboratory and the local study area was presented.

Database development using SQL Server was addressed in Chapter 4. The chapter described the processing of data collection and modification. The data types collected included meteorology data, hydrologic data and water quality data. Three principles of database design were addressed subsequently, including (1) defining the categories (tables) of data OWML collected; (2) determining the types of information (fields) within these categories; (3) creating the relationships amongst the multiple tables using key and/or index fields. By following the design principles, the database structure and the relationship between tables were developed and described. This chapter also introduced the procedures of data filling and the “3W” rule of database index selection. Issues about database security, backup and restoration were also outlined in this chapter. Finally, several sets of data retrieving queries were introduced.

Chapter 5 described the development of WebDAS. Different types of user interfaces, including a mapping interface and a data exploration interface were introduced and corresponding demonstrations were also included. The mapping interface selected was

Google maps API developed with JavaScript. This interface includes features such as zoom in/out, orientation, and multiple types of base map presentation. The data exploration interface selected was an ASP.net web page developed under Visual Web Developer that supports customized input editing, data visualization on the web page, data exporting and many other features. This chapter also introduced typical data accessing and analyzing procedures in a step by step fashion.

7.2 Conclusions

The goals of this research were to (1) Collect and enhance historical observed data developed by the Occoquan Watershed Monitoring Laboratory for construction of a Watershed Monitoring and Analysis Database; (2) Use Microsoft SQL Server to build the database, which includes all datasets collected by OWML and (3) Develop a basic Web-Based Data Visualization and Analysis System to provide an integrated interface for permitted users to explore, analyze and download data.

It can be concluded that the goals of this research project have been met. This research effort reaffirms the application of web based data visualization and analysis system in watershed management. Users of WebDAS, as illustrated in Chapters 4 and 5, can successfully explore, access, and perform basic analysis of the data stored in the OWML database. Users may explore the data by selecting the desired station, parameter and time range. Using selected data, users may also create a report view including a chart and a table which can later be exported as either PDF format or Excel format for further analysis. The WebDAS is accessible over the internet and is compatible with most browsers.

7.3 Recommendations

The research and study of data analysis systems for watershed management and environmental protection is still under development. Several aspects are particularly important for future study and research.

The first aspect is to standardize the structure of the watershed data warehouse. In most current watershed database design research, each project defines its own database design principles and database structure. Because most watershed monitoring data were generally based on similar types of observations, standardizing the structure of watershed databases can makes it easy to share the data with different interested groups or even with different organizations. This can help us significantly improve efficiency and minimize expense.

The second aspect is to improve the mapping interface. So far, all stations shown on the mapping interface are hard coded using JavaScript. Once there is a station change, such as adding a new station, or relocating an old station; the operator or administrator must manually update the program in order to synchronize the mapping interface with the OWML database. If an automated schema can be developed to solve this problem, it will significantly improve the maintenance performance of the monitoring system.

The third aspect is to improve the DEI design. Information technology development provides a high potential for more sophisticated web based applications. The DEI page should be a platform that provides efficient tools for data visualization. The improvement of the design layout and the advanced functionalities can make this data visualization more efficient and powerful. More studies are needed to define the standard functionalities for data analysis ideally based on a wide scale user survey.

The fourth aspect is the efficiency of the system. The WebDAS uses individual processing operation system to activate the SQL Server Database processing and data visualization. The system performance may be reduced or sometimes even timed out when there is a massive data transferring or multiple query requests running at the same time. If the massive transaction requests can be processed by several well equipped machines, the response time of the WebDAS will be shorter and it can also make the system maintenance easier.

References

Abdulrahman, F.H. (2008). "Using Kml Files as Encoding Standard to Explore Locations, Access and Display Data in Google Earth."

Allen, G. (2007). "Building a Dynamic Data Driven Application System for Hurricane Forecasting." Proceedings of the 7th international conference on Computational Science, Part I: ICCS 2007, Springer-Verlag, Beijing, China.

Beran, B., Valentine, D., Ingen, C.v., Zaslavsky, I., and Whitenack, T. (2008). "A Data Model for Environmental Observations."

Carleton, J.C., Dahlgren, A.R., and Tate, W.K. (2005). "A Relational Database for the Monitoring and Analysis of Aatershed Hydrologic Functions: Database Design and Pertinent Queries." *Computers & Geosciences*, 31(4), 393-402.

Ceri, S., Fraternali, P., and Paraboschi, S. (1999). "Data-Driven, One-to-One Web Site Generation for Data-Intensive Applications." Proceedings of the 25th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc.

Chaudhuri, S., and Narasayya, V. "An Efficient, Cost-Driven Index Selection Tool for Microsoft Sql Server." 146-155.

Chen, C.W., Herr, J., and Weintraub, L. (2004). "Decision Support System for Stakeholder Involvement." *Journal of environmental engineering*, 130, 714.

Darema, F. (2004). "Dynamic Data Driven Applications Systems: A New Paradigm for Application Simulations and Measurements ", Springer Berlin / Heidelberg, 662-669.

Delaney, K. (2000). *Inside Microsoft Sql Server 2000*, Microsoft Press.

Dougherty, M., Dymond, R.L., Grizzard, T.J., Godrej, A.N., and Zipper, C.E. "Evaluation of Land Use Andpopulation Change on Nutrient Delivery from an Urbanizing Watershed in Northern Virginia." 47.

Dutka, A.F., and Hansen, H.H. (1989). *Fundamentals of Data Normalization*, Addison-Wesley Longman Publishing Co., Inc.

Frankenberger, J.R., Brooks, E.S., Walter, M.T., Walter, M.F., and Steenhuis, T.S. (1999). "A Gis-Based Variable Source Area Hydrology Model." HYDROLOGICAL PROCESSES, 805-822.

Goodall, J., Valentine, D., and Maidment, D. (2006). "Exposing Hydrologic Databases Using Web Services."

GoogleMap-API. (2009). "Http://Code.Google.Com/Apis/Maps/."

Grizzard, T.J. (2005). "An Assessment of the Water Quality Effects of Nitrate in Reclaimed Water Delivered to the Occoquan Reservoir." Occoquan Watershed Monitoring Laboratory.

Gupta, P., Minhas, D.S., Tamhane, R.M., and Mookerjee, A.K. (2004). "Application of Geographical Information System (GIS) Tools in Watershed Analysis."

Hassett, B., Palmer, M., Bernhardt, E., Smith, S., Carr, J., and Hart, D. (2005). "Restoring Watersheds Project by Project: Trends in Chesapeake Bay Tributary Restoration." *Frontiers in Ecology and the Environment*, 3(5), 259-267.

Jeong, S., and Liang, X. (2005). "Survey of Cuahsi His at Uc Berkeley. Cuahsi Hydrologic Information System Symposium, University of Texas at Austin."

Jesse, L., and Dan, H. (2003). *Programming Asp.Net*.

JOSS. (2009). "Http://Www.Joss.Ucar.Edu/."

Kou, Y., Lu, C., N.Godrej, A., Grizzard, T.J., and Post, H. (2005). "A Web-Based Data Visualization and Analysis System for Watershed Management." *The International Association of Chinese Professionals in Geographic Information Science*, Vol. 11, No. 1, 10.

Leavesley, G.H., Markstrom, S.L., Viger, R.J., and Hay, L.E. (2005). "The Modular Modeling System (Mms): A Toolbox for Water- and Environmental-Resources Management."

Liu, Z., and Liang, Y. (2003). "Design of Metadata in a Hydrological Integrated Scientific Data Management System." *The Seventh Joint Conference on Information Sciences (JCIS), Cary, North Carolina,*, 418-421.

Mandel, J., Bennethum, L.S., Chen, M., Coen, J.L., Douglas, C.C., Franca, L.P., Johns, C.J., Kim, M., Knyazev, A.V., Kremens, R., Kulkarni, A., Qin, G., Vodacek, A., Wu, J., Zhao, W., and Zornes, A. (2005). "Towards a Dynamic Data Driven Application System for Wildfire Simulation."

Michael McGuire, Aryya Gangopadhyay, Anita Komlodi, Christopher Swan. (2008). "A User-Centered Design for a Spatial Data Warehouse for Data Exploration in Environmental Research." *ECOLOGICAL INFORMATIC S3*, 13.

Microsoft. (2008). "Http://Www.Microsoft.Com."

OWML. (2009). "Http://Www.Owml.Vt.Edu."

Petkovic, D. (2008). *Microsoft Sql Server 2008 a Beginner's Guide*.

Racicot, A.P. (2005). "Application of Web-Based Gis in Coastal Margin Observatories ", University of Washington.

Regmi, B. (2002). "Web-Enabled Spatial Decision Support System for Interdisciplinary Watershed Management," Virginia Tech.

Salayandia, L., Silva, P.P.d., Gates, A.Q., and Salcedo, F. (2006). "Workflow-Driven Ontologies: An Earth Sciences Case Study." *University of Texas at El Paso, Department of Computer Science, Technical Report* UTEP-CS-06-38.

Simovici, D.A., and Tenney, R.L. (1995). *Relational Database Systems*, Academic Press, Inc.

Tim, U.S., and Mallavaram, S. (2003). "Application of Gis Technology in Watershed-Based Management and Decision Making." *Watershed Update*, 8(1).

UNFPA. (2009). "Http://Www.Unfpa.Org."

USEPA_STORET. (2009). "Http://Www.Epa.Gov/Storet/."

USGS. (2010). "Http://Water.Usgs.Gov/."

USGS_NWIS. (2009). "Http://Waterdata.Usgs.Gov/Nwis/Rt."

VSWCB. (1991). "A Policy for Waste Treatment and Water Quality Management in the Occoquan Watershed." V. S. W. C. Board, ed., Richmond, Virginia.

Wilton, P., and Colby, J.W. (2005). *Begining Sql*, Wiley Publishing, Inc., Indianapolis, Indiana.

Xu, Z., N.Godrej, A., and Grizzard, T.J. (2007). "The Hydrological Calibration and Validation of a Complexly-Linked Watershed-Reservoir Model for the Occoquan Watershed, Virginia." *Journal of Hydrology*, 345(3-4), 167-183.

Zheng, C. (2005). "Summary of User Needs, Cuahsi Hydrologic Information System Symposium."

Appendix A. Database Construction Stored Procedures

Rain Data Insert Procedure:

```
USE [OWML_Index]
GO
/****** Object:  StoredProcedure [dbo].[rainInsert]
Script Date: 11/11/2009 13:45:08 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:                  <Kumar,Saurav>
-- Create date: <April,28,2009>
-- Description: <getChemID>
-- =============================================

ALTER procedure [dbo].[rainInsert] (@stationID int,@tmpTable RainDataTable READONLY, @initials varchar(5)='')
As
begin
BEGIN TRANSACTION;
BEGIN TRY
declare @autoMeasureID bigint
declare @param varchar(15)
declare @sampleQID char(1)
declare @matrixID char(2)
set @param='RAIN'
set @matrixID='A'
set @sampleQID='G'
```

```
EXEC getAutoMeasureID @initialsSupp=@initials,
@ret = @autoMeasureID OUTPUT
insert into
MasterData(datetime1,StationID,ParamID,ValueN,ValueC,PrecisionDigits,ResultID,SampleQID,MatrixID,ChemID,ChemDepthID,Co
mpositeID,AutoMeasureID,Remark)
select  cast (A+' '+B as datetime),
@stationID,@param,
OWMLWater_AllData.dbo.extractNumber(C,0), C,
OWMLWater_AllData.dbo.getPrecisionDigits(C),
OWMLWater_AllData.dbo.getResultCode(C,0),
@sampleQID,@matrixID,0,0,0,@autoMeasureID,
'Initial rain fill ..Wei'
from @tmpTable
where ISdate(A+' '+B)=1 and C is not null and
(ISNUMERIC(C)=1 or rtrim(ltrim(C))='')
END TRY
BEGIN CATCH
   SELECT
      ERROR_NUMBER() AS ErrorNumber
      ,ERROR_SEVERITY() AS ErrorSeverity
      ,ERROR_STATE() AS ErrorState
      ,ERROR_PROCEDURE() AS ErrorProcedure
      ,ERROR_LINE() AS ErrorLine
      ,ERROR_MESSAGE() AS ErrorMessage;

   IF @@TRANCOUNT > 0
      ROLLBACK TRANSACTION;
END CATCH;

IF @@TRANCOUNT > 0
   COMMIT TRANSACTION;
end
```

Get ChemicalAnalysis ID Procedure

```
USE [OWML_Index]
GO
/****** Object:  StoredProcedure [dbo].[getChemID]    Script Date: 11/11/2009 13:45:04 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:                  <Yang,Wei>
-- Create date: <April,28,2009>
-- Description: <getChemID>
-- =============================================
ALTER PROCEDURE [dbo].[getChemID]
        -- Add the parameters for the stored procedure here
        @initialsSupp varchar(5)='',
        @labid bigint,
        @ChemID bigint OUTPUT
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;
        Declare @crtDate smalldatetime
        set @crtDate=CAST(getdate() as smalldatetime)
        declare @initialsD varchar(5)

        if @initialsSupp = ''
        begin
                SELECT  @initialsD=cast(value as varchar(5))
                FROM fn_listextendedproperty('initials', 'USER', SYSTEM_USER,default, default, default, default);
```

```
        end
        else
        begin
                set @initialsD=@initialsSupp
        end
        insert into ChemicalAnalysis(LabID_CA, CreateDate,CreateInit)values (@labid,@crtDate,@initialsD)
        select @ChemID=max(ChemID)from ChemicalAnalysis
        where  CreateInit=@initialsD AND CreateDate=@crtDate
End
```

Get TimeCompositeID Procedure

```
USE [OWML_Index]
GO
/****** Object:  StoredProcedure [dbo].[getTimeCompositeID_time2]    Script Date: 11/11/2009 13:45:06 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:                  <Yang,,Wei>
-- Create date: <April,21,2009>
-- Description: <get TimeCompositeID>
-- =============================================
ALTER PROCEDURE [dbo].[getTimeCompositeID_time2]
        -- Add the parameters for the stored procedure here
        @initialsSupp varchar(5)='',
        @time2 datetime,
        @labid bigint,
        @CompositeID bigint OUTPUT
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;
        Declare @crtDate smalldatetime
        set @crtDate=CAST(getdate() as smalldatetime)
        declare @intialsD varchar(5)

        if @initialsSupp = ''
        begin
select  @intialsD=cast(value as varchar(5)) FROM fn_listextendedproperty('initials', 'USER', SYSTEM_USER,default, default, default,
default);
```

```
        end
        else
        begin
        set @intialsD=@initialsSupp
        end
    insert into TimeComposite(Datetime2) values (@time2)
    declare @TimeCompositeID bigint
    select @TimeCompositeID=TimeCompositeID
    from TimeComposite
    Where Datetime2=@time2
    insert into Composite(DepthCompositeID, TimeCompositeID, LabID_C, CreateDate,CreateInit, Remark)
    values ('0', @TimeCompositeID, @labid, @crtDate, @intialsD,'Filling by Wei')
    select @CompositeID=CompositeID
    from Composite
    where TimeCompositeID= @TimeCompositeID


END
```

Get ChemicalDepthID Procedure

```
USE [OWML_Index]
GO
/****** Object:  StoredProcedure [dbo].[getChemDepthID]    Script Date: 11/11/2009 13:45:03 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:                  <Yang,Wei>
-- Create date: <May,04,2009,>
-- Description: <Get ChemDepthID,>
-- =============================================
ALTER PROCEDURE [dbo].[getChemDepthID]
        -- Add the parameters for the stored procedure here
        @initialsSupp varchar(5)='',
        @labid varchar(10),

        @ChemDepthID bigint output,
        @DepthFt numeric(6,2)
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;
        Declare @labidWrite bigint
        if ISNUMERIC(@labid)=1
        set @labidWrite=cast(@labid as bigint)
        else
        set @labidWrite=null
        DECLARE @crtDate smalldatetime
        set @crtDate=CAST(GETDATE()as smalldatetime)
```

```
	declare @initialsD varchar(5)
	if @initialsSupp = ''
	begin
		select @initialsD=CAST(value as varchar(5))
		from fn_listextendedproperty('initials','USER', SYSTEM_USER, default,default,default,default)
	end
	else
	begin
		set @initialsD=@initialsSupp
	end
	INSERT INTO ChemicalAnalysisDepth(DepthFt,LabID_CAD,CreateDate,CreateInit)
	values (@DepthFt,@labidWrite,@crtDate,@initialsD)

	SELECT @ChemDepthID=MAX(ChemDepthID)
	from ChemicalAnalysisDepth
	Where CreateInit=@initialsD AND CreateDate=@crtDate
END
```

Get AutoMeasureID Procedure

```
USE [OWML_Index]
GO
/****** Object:  StoredProcedure [dbo].[getAutoMeasureID]
Script Date: 11/11/2009 13:45:01 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:                  <Yang,Wei>
-- Create date: <October,28,2009>
-- Description: <getChemID>
-- =============================================
ALTER procedure [dbo].[getAutoMeasureID] @initialsSupp varchar(5)='',@ret bigint OUTPUT
 as
begin
        declare @crtDate smalldatetime
        set  @crtDate= cast( getdate() as smalldatetime)
        declare @intialsD varchar(5)
        if @initialsSupp = ''
        begin
select  @intialsD=cast(value as varchar(5)) FROM fn_listextendedproperty('initials', 'USER', SYSTEM_USER,default, default, default,
default);
        end
        else
        begin
                set @intialsD=@initialsSupp
        end
        insert into AutoMeasure(CreateDate,CreateInit)values (@crtDate,@intialsD)
        select @ret=max(AutoMeasureID)from AutoMeasure  where  CreateInit=@intialsD AND CreateDate=@crtDate
end
```

Check Number Procedure

```
USE [OWML_Index]
GO
/****** Object:  StoredProcedure [dbo].[checkNumber]
Script Date: 11/11/2009 13:44:57 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:                <Kumar,Saurav>
-- Create date: <April 21, 2009,>
-- Description: <pharse numeric entry>
-- =============================================
ALTER PROCEDURE [dbo].[checkNumber]
        -- Add the parameters for the stored procedure here
        @number varchar(15),
        @minusIntended bit =0,
        @numberRet numeric(18,6) OUTPUT,
        @RcodeRet varchar(5) OUTPUT,
        @precisionDigits tinyint OUTPUT
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;
        --get rid of spaces
        SET @number =LTRIM(rtrim(@number))
        if @number!=''
        begin
        -- get first character
                declare @firstC tinyint
```

```
        set @firstC= ascii(LEFT(@number,1))
if @minusIntended=0
if @firstC=45 ---(45)
SET @firstC=60 --<(60)
        --checks on first character
if (@firstC >=48 and @firstC <=57)OR @firstC=45-- check if the character is between 0=48 to 9=57
begin
set @numberRet=cast(@number as numeric(18,6))
set @RcodeRet='OK'
set @precisionDigits=dbo.getPrecisionDigits(@number)
end
else
if @firstC= 43 OR @firstC= 62  -- check for +(43) or >(62)
begin
set @number=right(@number,len(@number)-1)
set @numberRet=cast(@number as numeric(18,6))
set @RcodeRet='>'
set @precisionDigits=dbo.getPrecisionDigits(@number)
end
else
if  @firstC= 60 -- check for -(45) or <(60)
begin
set @number=right(@number,len(@number)-1)
set @numberRet=cast(@number as numeric(18,6))
set @RcodeRet='<'
set @precisionDigits=dbo.getPrecisionDigits(@number)

end
else
if @firstC= 46 -- check for .(46)
begin
if len (@number)>1
```

```
        begin
        set @numberRet=cast(@number as numeric(18,6))
        set @RcodeRet='OK'
        set @precisionDigits=dbo.getPrecisionDigits(@number)
        end
        else
        begin
        set @numberRet=-888888
        set @RcodeRet='DNQ'
        set @precisionDigits=0
        end
        end
        end
        else
        begin
                --returning for blank as -999999,ER,0
                set @numberRet=-999999
                set @RcodeRet='ER'
                set @precisionDigits=0
        end


END
```

Appendix B. Database Construction Stored Functions

The Function of Checking Maximum Value of Parameter

```
USE [OWML_Index]
GO
/****** Object:  UserDefinedFunction [dbo].[check_paramValMax_status]
Script Date: 11/11/2009 13:51:12 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER function [dbo].[check_paramValMax_status](@PCode_val varchar(10))
returns numeric(18,6)
as
begin
  declare @ret numeric(18,6)
  set @ret = (Select PCodesTable.MaxLimit from PCodesTable where PCodesTable.Pcode like @PCode_val)
  return @ret
end
```

The Function of Checking Minimum Value of Parameter

```
USE [OWML_Index]
GO
/****** Object:  UserDefinedFunction [dbo].[check_paramValMin_status]
Script Date: 11/11/2009 14:11:33 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER function [dbo].[check_paramValMin_status](@PCode_val varchar(10))
returns numeric(18,6)
as
begin
  declare @ret numeric(18,6)
  set @ret = (Select PCodesTable.MinLimit from PCodesTable where PCodesTable.Pcode like @PCode_val)
  return @ret
end
```

The Function of Changing Degree Minutes Seconds into Decimal Degree

```
USE [OWML_Index]
GO
/****** Object:  UserDefinedFunction [dbo].[dms2decimaldegree]
Script Date: 11/11/2009 14:11:57 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER FUNCTION [dbo].[dms2decimaldegree](@expr char(15))
returns float
as
begin
	Declare @space int, @deg char(5), @min char(5), @sec char(5), @ret float, @exp char(15)
	set @space =0
	set @ret=0
	set @exp=@expr
	set @space= charindex(' ',rtrim(ltrim(@exp)),0)
	if @space=0
	begin
		if isnumeric(@exp)=1
		begin
			set @ret = cast(@exp as float)
			return @ret
		end
		else
			 return -1
	end
	else
		set @deg=substring(@exp,1,@space)
		set @exp=substring(@exp,@space+1,len(@exp))
```

```
		set @space= charindex(' ',rtrim(ltrim(@exp)),0)
		if @space=0
		begin
			if isnumeric(@exp)=1
			begin
			set @ret = cast(@deg as float)+cast(@exp as float)/60
			return @ret
			end
			else
			return -1
		end
		else
		set @min=substring(@exp,1,@space)
		set @exp=substring(@exp,@space+1,len(@exp))
		set @space= charindex(' ',rtrim(ltrim(@exp)),0)
		if @space<>0
		begin
		set @sec=substring(@exp,1,@space)
		end
		else
		 set @sec=@exp
		if isnumeric(@sec)=1
		begin
		set @ret = cast(@deg as float)+cast(@min as float)/60+cast(@sec as float)/3600
		return @ret
		end
		else
		return -1

	return @ret
end
```

The Function of Extracting Number

```
USE [OWML_Index]
GO
/****** Object:  UserDefinedFunction [dbo].[extractNumber]
Script Date: 11/11/2009 14:12:43 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
--Author:                   <Yang,Wei>
-- Create date: <April 20, ,>
-- Description: <return number take care of +,-,blank,.>
-- =============================================
ALTER FUNCTION [dbo].[extractNumber] (@number varchar(30),@minusIntended tinyint =0)
--@minusIntended = 0 minus removed
--@minusIntended = 1 minus remain same
--@minusIntended = 2 minus converted to -99999

RETURNS numeric(18,6)
AS
BEGIN
        declare @numberRet numeric(18,6)
        SET @number =LTRIM(rtrim(@number))
        if @number!=''
        begin
        -- get first character
                declare @firstC tinyint
                set @firstC= ascii(LEFT(@number,1))
                if @minusIntended=2 and @firstC=45
                set @numberRet=-999999
```

```
        else
        begin
-- if first character is bewteen 0-9 or . with length more than one phrase as number and return
        if @minusIntended=0
        if @firstC=45 ---(45)
        SET @firstC=60 --<(60)

        --checks on first character
        if (@firstC >=48 and @firstC <=57)OR @firstC=45-- check if the character is between 0=48 to 9=57
        begin
        set @numberRet=cast(@number as numeric(18,6))
        end
        else
        if @firstC= 43 OR @firstC= 62  -- check for +(43) or >(62)
        begin
        set @number=right(@number,len(@number)-1)
        set @numberRet=cast(@number as numeric(18,6))
        end
        else
        if  @firstC= 60 -- check for  <(60)
        begin
        set @number=right(@number,len(@number)-1)
        set @numberRet=cast(@number as numeric(18,6))
        end
        else
        if @firstC= 46 -- check for .(46)
        begin
        if len (@number)>1
        begin
        set @numberRet=cast(@number as numeric(18,6))
        end
        else
```

```
                begin
                set @numberRet=-888888
                end
                end
        end
        end
        else
        begin
                --returning for blank as -999999,ER,0
                set @numberRet=-999999
        end
return @numberRet
END
```

The Function of Getting Precision Digits

```
USE [OWML_Index]
GO
/****** Object:  UserDefinedFunction [dbo].[getPrecisionDigits]    Script Date: 11/11/2009 14:13:01 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:                <Kumar,,Saurav>
-- Create date: <April 20, ,>
-- Description: <return precision digits>
-- =============================================
ALTER FUNCTION [dbo].[getPrecisionDigits](@input varchar(30))
returns tinyint
as
begin
        declare @dotAt tinyint
        declare @len tinyint
        declare @ret tinyint
        set @input=ltrim(rtrim(@input))
        select @dotAt = charindex('.',@input),@len=len(@input)
        set @ret=@len-@dotAt
        if @dotAt=0
                set @ret=0

        return @ret
END
```

The Function of Getting Result Codes

```
USE [OWML_Index]
GO
/****** Object:  UserDefinedFunction [dbo].[getResultCode]    Script Date: 11/11/2009 14:13:20 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
--Author:		<Yang,Wei>
-- Create date: <April 20, ,>
-- Description: <return result code based on number>
-- =============================================
ALTER FUNCTION [dbo].[getResultCode](@number varchar(30),@minusIntended tinyint =0)
--@minusIntended = 0 minus converted to <
--@minusIntended = 1 minus converted to OK
--@minusIntended = 2 minus converted to ER
RETURNS char(5)
AS
BEGIN
	SET @number =LTRIM(rtrim(@number))
	if @number!=''
	begin
	-- get first character
		declare @RcodeRet char(5)
		declare @firstC tinyint
		set @firstC= ascii(LEFT(@number,1))
		if @minusIntended=2 and @firstC=45
			Set @RcodeRet='ER'
		else
			begin
```

```
        if @minusIntended=0
if @firstC=45 ---(45)
SET @firstC=60 --<(60)
        --checks on first character
if (@firstC >=48 and @firstC <=57)OR @firstC=45-- check if the character is between 0=48 to 9=57
begin
set @RcodeRet='OK'
end
else
if @firstC= 43 OR @firstC= 62  -- check for +(43) or >(62)
begin
set @RcodeRet='>'
end
else
if  @firstC= 60 -- check for -(45) or <(60)
begin
set @RcodeRet='<'
end
else
if @firstC= 46 -- check for .(46)
begin
if len (@number)>1
begin

set @RcodeRet='OK'
end
else
begin
set @RcodeRet='DNQ'
end
end
end
```

```
        end
        else
        begin
                --returning for blank as -999999,ER,0
                set @RcodeRet='ER'
        end
        return @RcodeRet
END
```

Appendix C. Data Retrieval Queries for Index Selection

Query 1:

```
select *
from MasterData
where datetime1 < '2008-12-01 '   and ParamID ='rain'
```

Query 2:

```
select *
from MasterData
where StationID=63
```

Query 3:

```
WITH dailyrain_CTE (DateInt,StationID, TotRain) as (
select date1=cast ( datetime1 as int)+1,
StationID, sum(ValueN)
from MasterData where ParamID='RAIN' and ResultID='OK'
group by StationID, cast ( datetime1 as int)+1)
select year(CAST(DateInt as datetime)) as years,MONTH(CAST(DateInt as datetime)) as months, StationID ,AVG(TotRain) as
dailyAverageRain from dailyrain_CTE
group by cube(year(CAST(DateInt as datetime)), MONTH(CAST(DateInt as datetime)), StationID) order by StationID, years,months
```

Query 4:

```
select datetime1, StationID, ParamID, ValueN, PrecisionDigits, ResultID, SampleQID, MatrixID, DepthFt
```

```
from MasterData join ChemicalAnalysisDepth
on (MasterData.ChemDepthID=ChemicalAnalysisDepth.ChemDepthID)
where ParamID='DO' and ValueN<4 and ResultID='ok'
```

Query 5:

```
Select *
from MasterData
where ParamID='DO' and ValueN<4
```

Query 6:

```
Select *
from MasterData
```

Query 7:

```
select COUNT(TrackID) as numberOfData, ParamID
from MasterData
group by ParamID
```

Query 8:

```
select datetime1, StationID, StationName,ParamID, ValueN, PrecisionDigits, ResultID, SampleQID, MatrixID
from MasterData join Stations
on (MasterData.StationID=Stations.StationID)
```

Appendix D. Mapping Interface Code

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3c.org/TR/1999/REC-html401-19991224/loose.dtd">
<!-- saved from url=(0033)http://staging-xx.water.usgs.gov/ -->

<html lang="en" xml:lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
        <title>OccoquanWatershedMonitoringSystem</title>


<script src="http://maps.google.com/maps?file=api&amp;v=2.103&amp;key=ABQIAAAAyMblkCuPBJcrVc4glTbaMhROBMt6iRM_3JT2L3XQ7rcjci6yRhTTGGMLEf-syf8t3WPKRJPsdwhnhw" type="text/javascript"></script>

<script type="text/javascript">

   function load() {
      var map;
      var ovmap;

      if (GBrowserIsCompatible()) {
         map = new GMap2(document.getElementById("map"));

         //Add map controls
         map.addControl(new GScaleControl());
         map.addMapType(G_PHYSICAL_MAP);
         map.addControl(new GLargeMapControl());
         var mapControl = new GHierarchicalMapTypeControl();
```

```
        mapControl.clearRelationships();
        mapControl.addRelationship(G_SATELLITE_MAP, G_HYBRID_MAP, "Labels", true);
        map.addControl(mapControl);

        //The boundary
        var outline = new GGeoXml("http://www.ecolve.com/owml/OCC_Contour.kmz");
        map.addOverlay(outline);


        //Declare the icons. Triangle Icon locates on MSN Space
        var gicons = [];
        gicons['mrk_00'] = new GIcon(G_DEFAULT_ICON, 'https://zcnpwa.bay.livefilestore.com/y1p-
1ue9ocNQCy57Gqe0scxEgqrtwzyM_ULPyeBe040kpydT6QmgVeuLQrI4PE-m4SjDEx9pjLjNz_xIeADG9_ATOK2-
ElS28PT/1.gif');
        gicons['mrk_00'].iconSize = new GSize(13, 13);
        gicons['mrk_00'].shadowSize = new GSize(5, 5);
        gicons['mrk_00'].iconAnchor = new GPoint(5, 5);

        gicons['mrk_01'] = new GIcon(G_DEFAULT_ICON,
'https://zcnpwa.bay.livefilestore.com/y1pvtHhDGFYt4EfnLzsu85lJX3rX9RxSwzu33ULqkzQNetBypEjnVN739wH5U-
UtQZa5HMb7fbKGkWWYUmcPFByuBw-CgDjlJde/2.gif');
        gicons['mrk_01'].iconSize = new GSize(13, 13);
        gicons['mrk_01'].shadowSize = new GSize(5, 5);
        gicons['mrk_01'].iconAnchor = new GPoint(5, 5);

        gicons['mrk_02'] = new GIcon(G_DEFAULT_ICON, 'https://zcnpwa.bay.livefilestore.com/y1poeDH9yad0vADH-
3r3WJjhG_dBF9-6gCQWcF0ej96XGUzGVkNZV6Vcnob_DAEBPXIVtUSUUHY8OQH3O9XaguZ4sbQKg2M08fH/3.gif');
        gicons['mrk_02'].iconSize = new GSize(13, 13);
        gicons['mrk_02'].shadowSize = new GSize(5, 5);
```

```
        gicons['mrk_02'].iconAnchor = new GPoint(5, 5);

        gicons['mrk_03'] = new GIcon(G_DEFAULT_ICON,
'https://zcnpwa.bay.livefilestore.com/y1poeDH9yad0vBMzQKZ2U9-
GAfJRjJqtUmh0pNPji88MSwy_AdmWjBH0esrEoooMxMd4skj289171ni6NO109wDUT11Wwue9XED/4.gif');
        gicons['mrk_03'].iconSize = new GSize(13, 13);
        gicons['mrk_03'].shadowSize = new GSize(5, 5);
        gicons['mrk_03'].iconAnchor = new GPoint(5, 5);

        gicons['mrk_04'] = new GIcon(G_DEFAULT_ICON,
'https://zcnpwa.bay.livefilestore.com/y1pPoK6XmSyJA_AtYMsB8HiS1bj4L7cfpAzZfvpZmnFE5i9D8Zy31au9q8Zxh-
HjhE1bOlGIPpHnz2f5tCcAXiAYsIssERK0kn6/5.gif');
        gicons['mrk_04'].iconSize = new GSize(13, 13);
        gicons['mrk_04'].shadowSize = new GSize(5, 5);
        gicons['mrk_04'].iconAnchor = new GPoint(5, 5);

        gicons['mrk_05'] = new GIcon(G_DEFAULT_ICON,
'https://zcnpwa.bay.livefilestore.com/y1puK1NsylObYtwJF5my8MRl1yMZyIsumlak2DGqw3Cq_nuDkr1soxrbO3c3NlG1luS
MLak-bNW0pC0zBYlqQfWBTN0c2uxVyDX/6.gif');
        gicons['mrk_05'].iconSize = new GSize(13, 13);
        gicons['mrk_05'].shadowSize = new GSize(5, 5);
        gicons['mrk_05'].iconAnchor = new GPoint(5, 5);


        //Set the map bounds
        var bounds = new GLatLngBounds(new GLatLng(38.45, -77.6), new GLatLng(39.05, -77.09));


        //Set the center of the map
```

```
        map.setCenter(new GLatLng(38.72, -77.5), map.getBoundsZoomLevel(bounds), G_PHYSICAL_MAP);


        var icon = new GIcon();
        icon.infoWindowAnchor = new GPoint(5, 1);

        var tooltip = document.createElement("div");
        document.getElementById("map").appendChild(tooltip);
        tooltip.style.visibility = "hidden";

        function createMarker(point, name, html, icontype, zIdx)
        {
            var marker = new GMarker(point, gicons[icontype]);
            GEvent.addListener(marker, "click", function() { marker.openInfoWindowHtml(html); xgpoint =
marker.getPoint(); });

            marker.tooltip = '<div class="sitetip">' + name + '</div>';
            GEvent.addListener(marker, "mouseover", function() { showTooltip(marker); });
            GEvent.addListener(marker, "mouseout", function() { tooltip.style.visibility = "hidden" });


            map.addOverlay(marker);
            return marker;
        }

        // A function to create a tabbed marker and set up the event window
        // This version accepts a variable number of tabs, passed in the arrays htmls[] and labels[]
        function createTabbedMarker(point, tabhtmls, tablabels) {
            var mkr = new GIcon();
```

```
mkr.image = "/kml/images/data_sites.gif";
mkr.iconSize = new GSize(20, 20);
mkr.shadowSize = new GSize(5, 5);
mkr.iconAnchor = new GPoint(6, 20);
mkr.infoWindowAnchor = new GPoint(5, 1);
var marker = new GMarker(point, mkr);


/* add next line for tooltip */
mkr.title = '<div class="tooltip">' + point + '</div>';

GEvent.addListener(marker, "click", function() {
   // adjust the width so that the info window is large enough for this many tabs
   if (tabhtmls.length > 1) {
      tabhtmls[0] = '<div style="height:200px;width:' + tabhtmls.length * 80 + 'px">' + tabhtmls[0] + '</div>';
   }
   var tabs = [];

   for (var i = 0; i < tabhtmls.length; i++) {
      tabs.push(new GInfoWindowTab(tablabels[i], tabhtmls[i]));
   }
   marker.openInfoWindowTabsHtml(tabs);
});

/* Events added to show the tool tips (mouse over event) */
GEvent.addListener(mkr, "mouseover", function() { showTooltip(cmmarker); });
GEvent.addListener(mkr, "mouseout", function() { tooltip.style.visibility = "hidden" });

return marker;
```

```
        }


        function showTooltip(marker) {
            tooltip.innerHTML = marker.tooltip;
            var point = map.getCurrentMapType().getProjection().fromLatLngToPixel(map.getBounds().getSouthWest(),
map.getZoom());
            var offset = map.getCurrentMapType().getProjection().fromLatLngToPixel(marker.getPoint(), map.getZoom());
            var anchor = marker.getIcon().iconAnchor;
            var width = marker.getIcon().iconSize.width;
            var pos = new GControlPosition(G_ANCHOR_BOTTOM_LEFT, new GSize(offset.x - point.x - anchor.x + width,
-offset.y + point.y + anchor.y));
            pos.apply(tooltip);
            tooltip.style.visibility = "visible";
        }


        // add stations as point files onto map
        point = new GPoint(-77.674, 38.755); marker = createMarker(
        point, 'BR02', "<a href='DataExplore.aspx?StID=3'>StationID: 3</a> <br> Station Name: BR02 <br>
Location:South Run Fauquier County Vint Hill Farms Military Res northern corner<br> ", "mrk_03", 1000)
        map.addOverlay(marker)

        point = new GPoint(-77.666, 38.770); marker = createMarker
        (point, 'BR03', "<a href='DataExplore.aspx?StID=4'>Station ID: 4</a>  <br/> StationName: BR03 <br/>
Location:South Run,Prince William Co., at bridge on State Hwy. 684 (Buckland Mill) <br/> ", "mrk_03", 1000);
        map.addOverlay(marker);

        point = new GPoint(-77.625, 38.792); marker = createMarker
```

```
        (point, 'BR04', "<a href='DataExplore.aspx?StID=5'>Station ID: 5</a>   <br/> StationName: BR04 <br/>
Location:North Fork Creek,Prince William Co., at bridge on US Hwy. 29, south of Linton Hall <br/> ", "mrk_03", 1000);
        map.addOverlay(marker);

        point = new GPoint(-77.658, 38.784); marker = createMarker
        (point, 'BR05', "<a href='DataExplore.aspx?StID=6'>Station ID: 6</a>  <br/> StationName: BR05 <br/>
Location:Unnamed Stream,Prince William Co., culvert on northbound US 29 <br/> ", "mrk_03", 1000);
        map.addOverlay(marker);

        point = new GPoint(-77.659, 38.752); marker = createMarker
        (point, 'BR06', "<a href='DataExplore.aspx?StID=7'>Station ID: 7</a>  <br/> StationName: BR06 <br/>
Location:Unnamed Stream,Prince William Co., culvert on north side State Hwy. 215 <br/>", "mrk_03", 1000);
        map.addOverlay(marker);

        point = new GPoint(-77.689, 38.749); marker = createMarker
        (point, 'BR07', " <a href='DataExplore.aspx?StID=8'>Station ID: 8</a>   <br/> StationName: BR07 <br/> Location:
South Run,Fauquier Co., culvert on St. Hwy. 793, 0.9 mi. S of St. Hwy. 600 <br/>  </a> ", "mrk_03", 1000);
        map.addOverlay(marker);




        point = new GPoint(-77.674, 38.765); marker = createMarker
        (point, 'LM01', " <a href='DataExplore.aspx?StID=10'>Station ID: 10</a> <br/> StationName: LM01 <br/> Location:
Lake Manassas,western Prince William County, station nearest dam <br/>  </a> ", "mrk_00", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.628, 38.775); marker = createMarker
        (point, 'LM02', "<a href='DataExplore.aspx?StID=11'>Station ID: 11</a> <br/> StationName: LM02 <br/>
Location:Lake Manassas,western Prince William County <br/> ", "mrk_00", 1000);
```

```
        map.addOverlay(marker);
        point = new GPoint(-77.624, 38.783); marker = createMarker
        (point, 'LM03', "<a href='DataExplore.aspx?StID=12'>Station ID: 12</a> <br/> StationName: LM03 <br/>
Location:Lake Manassas,western Prince William County <br/> ", "mrk_00", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.634, 38.770); marker = createMarker
        (point, 'LM04', "<a href='DataExplore.aspx?StID=13'>Station ID: 13</a> <br/> StationName: LM04 <br/>
Location:Lake Manassas,western Prince William County <br/> ", "mrk_00", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.650, 38.769); marker = createMarker
        (point, 'LM05', "<a href='DataExplore.aspx?StID=14'>Station ID: 14</a>  <br/> StationName: LM05 <br/>
Location:Lake Manassas,western Prince William County <br/> ", "mrk_00", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.663, 38.773); marker = createMarker
        (point, 'LM06', "<a href='DataExplore.aspx?StID=15'>Station ID: 15</a><br/> StationName: LM06 <br/>
Location:Lake Manassas,western Prince William County <br/> ", "mrk_00", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.658, 38.775); marker = createMarker
        (point, 'LM07', " <a href='DataExplore.aspx?StID=16'>Station ID: 16</a> <br/> StationName: LM07 <br/>
Location:Lake Manassas,western Prince William County <br/> ", "mrk_00", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.643, 38.778); marker = createMarker
        (point, 'LM08', "<a href='DataExplore.aspx?StID=17'>Station ID: 17</a> <br/> StationName: LM08 <br/>
Location:Lake Manassas,western Prince William County <br/> ", "mrk_00", 1000);
        map.addOverlay(marker);

        point = new GPoint(-77.278, 38.696); marker = createMarker
        (point, 'RE01', " <a href='DataExplore.aspx?StID=32'>Station ID: 32</a> <br/> StationName: RE01 <br/> Location:
Occoquan Reservoir,Occoquan Dam <br/>  </a> ", "mrk_01", 1000);
```

```
        map.addOverlay(marker);
        point = new GPoint(-77.283, 38.696); marker = createMarker
        (point, 'RE02', "<a href='DataExplore.aspx?StID=33'>Station ID: 33</a> <br/> StationName: RE02 <br/> Location:
Occoquan Reservoi,second power line, 0.3 mi. above dam<br/>  </a> ", "mrk_01", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.306, 38.708); marker = createMarker
        (point, 'RE05', "<a href='DataExplore.aspx?StID=35'>Station ID: 35</a> <br/> StationName: RE05 <br/> Location:
Occoquan Reservoir,below Sandy Run, 1.8 mi. above dam <br/>  </a> ", "mrk_01", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.320, 38.712); marker = createMarker
        (point, 'RE10', "<a href='DataExplore.aspx?StID=36'>Station ID: 36</a> <br/> StationName: RE10 <br/> Location:
Occoquan Reservoir,Jacob's Rock, 4.0 mi. above dam <br/>  </a> ", "mrk_01", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.351, 38.721); marker = createMarker
        (point, 'RE15', "<a href='DataExplore.aspx?StID=37'>Station ID: 37</a> <br/> StationName: RE15 <br/> Location:
Occoquan Reservoir,Ryan's Dam, 6.1 mi. above dam <br/>  </a> ", "mrk_01", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.379, 38.719); marker = createMarker
        (point, 'RE20', "<a href='DataExplore.aspx?StID=43'>Station ID: 43</a>  <br/> StationName: RE20 <br/> Location:
Occoquan Reservoir,below confluence of Bull Run and Occoquan Creek <br/>  </a> ", "mrk_01", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.386, 38.723); marker = createMarker
        (point, 'RE25', "<a href='DataExplore.aspx?StID=45'>Station ID: 45</a> <br/> StationName: RE25 <br/> Location:
Occoquan Reservoir,Occoquan Creek above confluence with Bull Run <br/>  </a> ", "mrk_01", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.388, 38.742); marker = createMarker
        (point, 'RE30', "<a href='DataExplore.aspx?StID=47'>Station ID: 47</a> <br/> StationName: RE30 <br/> Location:
Occoquan Reservoir,Bull Run Marina, 10.5 mi. above dam <br/>  </a> ", "mrk_01", 1000);
        map.addOverlay(marker);
```

```
        point = new GPoint(-77.394, 38.718); marker = createMarker
        (point, 'RE35', "<a href='DataExplore.aspx?StID=49'>Station ID: 49</a> <br/> StationName: RE35 <br/> Location:
Occoquan Reservoir,Ravenwood Bridge, 11.2 mi. above dam <br/>  </a> ", "mrk_01", 1000);
        map.addOverlay(marker);

        point = new GPoint(-77.802, 38.780); marker = createMarker
        (point, 'AIR', " <a href='DataExplore.aspx?StID=64'>Station ID: 64</a><br/> StationName: AIR <br/> Location:
Airlie station <br/>  </a> ", "mrk_02", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.556, 38.620); marker = createMarker
        (point, 'CRW', "<a href='DataExplore.aspx?StID=66'>Station ID: 66</a>  <br/> StationName: CRW <br/> Location:
Cedar Run Wetlands<br/>  </a> ", "mrk_02", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.696, 38.760); marker = createMarker
        (point, 'CHRE', "<a href='DataExplore.aspx?StID=67'>Station ID: 67</a>  <br/> StationName: CHRE <br/>
Location: C.Hunter Ritchie Elem. School <br/>  </a> ", "mrk_02", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.398, 38.784); marker = createMarker
        (point, 'CE', "<a href='DataExplore.aspx?StID=68'>Station ID: 68</a>  <br/> StationName: CE <br/> Location:
Clifton Elem. School <br/>  </a> ", "mrk_02", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.564, 38.788); marker = createMarker
        (point, 'BF', "<a href='DataExplore.aspx?StID=69'>Station ID: 69</a>  <br/> StationName: BF <br/> Location: Balls
Ford Rd Yardwaste Facility<br/>  </a> ", "mrk_02", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.622, 38.762); marker = createMarker
        (point, 'LM', " <a href='DataExplore.aspx?StID=70'>Station ID: 70</a> <br/> StationName: LM <br/> Location:
Lake Manassas Water Treatment Plant <br/>  </a> ", "mrk_02", 1000);
        map.addOverlay(marker);
```

```
        point = new GPoint(-77.428, 38.637); marker = createMarker
        (point, 'PWLF', "<a href='DataExplore.aspx?StID=71'>Station ID: 71</a>  <br/> StationName: PWLF <br/>
Location: Prince William County Regional Landfill<br/>  </a> ", "mrk_02", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.634, 38.882); marker = createMarker
        (point, 'EVR', "<a href='DataExplore.aspx?StID=72'>Station ID: 72</a>  <br/> StationName: EVR <br/> Location:
Evergreen Fire Dept. <br/>  </a> ", "mrk_02", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.723, 38.621); marker = createMarker
        (point, 'CPK', "<a href='DataExplore.aspx?StID=73'>Station ID: 73</a>  <br/> StationName: CPK <br/> Location:
Crockette Park <br/>  </a> ", "mrk_02", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.370, 38.872); marker = createMarker
        (point, 'FOP', "<a href='DataExplore.aspx?StID=74'>Station ID: 74</a>  <br/> StationName: FOP <br/> Location:
Fair Oaks Police Dept. <br/>  </a> ", "mrk_02", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.481, 38.749); marker = createMarker
        (point, 'OWML', "<a href='DataExplore.aspx?StID=78'>Station ID: 78</a>  <br/> StationName: OWML <br/>
Location: Occoquan Watershed Monitoring Laboratory <br/>  </a> ", "mrk_02", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.259, 38.691); marker = createMarker
        (point, 'LORT', "<a href='DataExplore.aspx?StID=79'>Station ID: 79</a>  <br/> StationName: LORT <br/>
Location: Lorton Water Treatment Plant <br/>  </a> ", "mrk_02", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.448, 38.705); marker = createMarker
        (point, 'LJ', "<a href='DataExplore.aspx?StID=80'>Station ID: 80</a> <br/> StationName: LJ <br/> Location: Lack
Jackson Dam <br/>  </a> ", "mrk_02", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.448, 38.951); marker = createMarker
```

```
        (point, 'DULLES', "<a href='DataExplore.aspx?StID=81'>Station ID: 81</a> <br/> StationName: DULLES <br/>
Location: Dulles International Airport <br/>  </a> ", "mrk_02", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.622, 38.762); marker = createMarker
        (point, 'LMDAM', " <a href='DataExplore.aspx?StID=85'>Station ID: 85</a> <br/> StationName: LMDAM <br/>
Location: Lake Manassas Dam<br/>  </a> ", "mrk_02", 1000);
        map.addOverlay(marker);

        point = new GPoint(-77.277, 38.694); marker = createMarker
        (point, 'ST01', " <a href='DataExplore.aspx?StID=52'>Station ID: 52</a> <br/> StationName: ST01 <br/> Location:
Occoquan Dam,northeastern Prince William County, north of Town of Occoquan <br/>  </a> ", "mrk_04", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.290, 38.680); marker = createMarker
        (point, 'ST05', "<a href='DataExplore.aspx?StID=53'>Station ID: 53</a> <br/> StationName: ST05 <br/> Location:
Hooes Run near Occoquan Dam,near Occoquan Dam<br/>  </a> ", "mrk_04", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.446, 38.705); marker = createMarker
        (point, 'ST10', "<a href='DataExplore.aspx?StID=54'>Station ID: 54</a> <br/> StationName: ST10 <br/> Location:
Lake Jackson,Occoquan River near Manassas<br/>  </a> ", "mrk_04", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.554, 38.616); marker = createMarker
        (point, 'ST20', "<a href='DataExplore.aspx?StID=55'>Station ID: 55</a> <br/> StationName: ST20 <br/> Location:
Cedar Run near Aden,southern Prince William County  (replaced by ST25, 8/20/91)<br/>  </a> ", "mrk_04", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.554, 38.615); marker = createMarker
        (point, 'ST25', "<a href='DataExplore.aspx?StID=56'>Station ID: 56</a>  <br/> StationName: ST25 <br/> Location:
Cedar Run Near Aden,southern Prince William County  (500 ft downstream of ST20) <br/>  </a> ", "mrk_04", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.564, 38.749); marker = createMarker
```

```
        (point, 'ST30', "<a href='DataExplore.aspx?StID=57'>Station ID: 57</a> <br/> StationName: ST30 <br/> Location:
Broad Run near Bristow,central Prince William County<br/>  </a> ", "mrk_04", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.415, 38.766); marker = createMarker
        (point, 'ST40', "<a href='DataExplore.aspx?StID=58'>Station ID: 58</a> <br/> StationName: ST40 <br/> Location:
Yates Ford,Bull Run near Clifton <br/>  </a> ", "mrk_04", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.450, 38.803); marker = createMarker
        (point, 'ST45', "<a href='DataExplore.aspx?StID=59'>Station ID: 59</a> <br/> StationName: ST45 <br/> Location:
Yorkshire,southwest Fairfax County<br/>  </a> ", "mrk_04", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.467, 38.821); marker = createMarker
        (point, 'ST50', "<a href='DataExplore.aspx?StID=60'>Station ID: 60</a> <br/> StationName: ST50 <br/> Location:
Cub Run near Bull Run,southwest Fairfax County<br/>  </a> ", "mrk_04", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.571, 38.889); marker = createMarker
        (point, 'ST60', "<a href='DataExplore.aspx?StID=61'>Station ID: 61</a> <br/> StationName: ST60 <br/> Location:
Bull Run near Catharpin,northern Prince William County<br/>  </a> ", "mrk_04", 1000);
        map.addOverlay(marker);
        point = new GPoint(-77.672, 38.780); marker = createMarker
        (point, 'ST70', "<a href='DataExplore.aspx?StID=62'>Station ID: 62</a> <br/> StationName: ST70 <br/> Location:
Broad Run at Buckland,northern Prince William County<br/>  </a> ", "mrk_04", 1000);
        map.addOverlay(marker);

        point = new GPoint(-77.117, 38.929); marker = createMarker
        (point, 'PR01', "<a href='DataExplore.aspx?StID=182'>Station ID: 182</a> <br/> StationName: PR01 <br/>
Location:Chain Bridge, Va 123, Potomac River, Arlington Co. VA.<br/>  </a> ", "mrk_05", 1000);
        map.addOverlay(marker);
```

```
      }
    }

</script>
</head>


<body id="body1" onload="load()" onunload="GUnload()">
<style type=text/css>
 .sitetip {
   border: #006699 1px solid;
   font-size: 12px;
   font-weight: bold;
   BACKGROUND-COLOR: #ffffff;
}
</style>
<!-- BEGIN OWML Header Template -->
<DIV id=usgscolorband>
<DIV id=usgsbanner>
<DIV id=usgsidentifier>
<img alt="bannerimage" src="image/bannerimage.gif"
  style="width: 100%; height: 125px" /></DIV>
<DIV id=usgsccsabox>
<DIV id=usgsccsa>
<BR>
<A href="http://www.owml.vt.edu/index.htm">OWML Home</A>
<A href="http://www.owml.vt.edu/index.htm">Stakeholders</A>                    <A
href="http://www.owml.vt.edu/index.htm">Publications</A>                    <A
href="http://www.owml.vt.edu/index.htm">Project</A>
```

```
<A href="http://www.owml.vt.edu/index.htm">Links</A>
<A href="http://www.owml.vt.edu/aboutowml.htm">About OWML</A>
<A href="http://www.owml.vt.edu/contactus.htm">Contact US</A>

                                    </DIV>
                                </DIV>
                            </DIV>

                        </DIV>



<!-- END OWML Header Template -->

<TABLE class=mainlayout>
  <TBODY>
  <TR style="VERTICAL-ALIGN: top">
    <TD class="sidepane"><P class="leftheadlight"></P><br>

              <p><font size="2"><b><br>
                <br>
     </TD>
    <TD class=rightpane valign="top" width="100%">

<!-- main body -->
<div id="bodyDIV">
          <H2>Monitoring Stations in Occoquan Watershed
 </H2>
          <!-- Google Maps Section -->
```

```
        <div id="map" style="width: 100%; height:700px; border: 3px black solid;"></div>
        <!-- End of Google Maps Section -->

        <!-- Map Explanation -->
        <table>
                <tr><td colspan="2" align="left"><b>Explanation</b></td></tr>
                <tr><td><img
src="https://zcnpwa.bay.livefilestore.com/y1pvtHhDGFYt4EfnLzsu85lJX3rX9RxSwzu33ULqkzQNetBypEjnVN739wH5U-
UtQZa5HMb7fbKGkWWYUmcPFByuBw-CgDjlJde/2.gif" alt="Reservior Stations (RE) and Stream Stations (ST)
"></td><td>Reservior Stations (RE)</td></tr>
                <tr><td><img
src="https://zcnpwa.bay.livefilestore.com/y1pPoK6XmSyJA_AtYMsB8HiS1bj4L7cfpAzZfvpZmnFE5i9D8Zy31au9q8Zxh-
HjhE1bOlGIPpHnz2f5tCcAXiAYsIssERK0kn6/5.gif" alt="Reservior Stations (RE) and Stream Stations (ST)
"></td><td>Stream Stations (ST)</td></tr>
                <tr><td><img src="https://zcnpwa.bay.livefilestore.com/y1p-
1ue9ocNQCy57Gqe0scxEgqrtwzyM_ULPyeBe040kpydT6QmgVeuLQrI4PE-m4SjDEx9pjLjNz_xIeADG9_ATOK2-
ElS28PT/1.gif" alt="Lake Manassas Stations"></td><td>Lake Manassas Stations</td></tr>
                <tr><td><img src="https://zcnpwa.bay.livefilestore.com/y1poeDH9yad0vADH-3r3WJjhG_dBF9-
6gCQWcF0ej96XGUzGVkNZV6Vcnob_DAEBPXIVtUSUUHY8OQH3O9XaguZ4sbQKg2M08fH/3.gif" alt="Weather
Stations"></td><td>Weather and Rain Stations</td></tr>
                <tr><td><img src="https://zcnpwa.bay.livefilestore.com/y1poeDH9yad0vBMzQKZ2U9-
GAfJRjJqtUmh0pNPji88MSwy_AdmWjBH0esrEoooMxMd4skj289171ni6NO109wDUT11Wwue9XED/4.gif" alt="Bull Run
Stations"></td><td>Bull Run Stations(BR) </td></tr>
                <tr><td><img
src="https://zcnpwa.bay.livefilestore.com/y1puK1NsylObYtwJF5my8MRl1yMZyIsumlak2DGqw3Cq_nuDkr1soxrbO3c3NlG
1luSMLak-bNW0pC0zBYlqQfWBTN0c2uxVyDX/6.gif" alt="Other Stations"></td><td>Other Stations </td></tr>
                <tr><td></td><td> <br></td></tr>

        </table>
```

```
</div>
</td></tr></table>
</body>
</html>
```

Appendix E. Data Exploration Interface Code

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="DataExplore.aspx.vb" Inherits="_Default" %>

<%@ Register assembly="Microsoft.ReportViewer.WebForms, Version=9.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a" namespace="Microsoft.Reporting.WebForms" tagprefix="rsweb" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>OccoquanWatershedMonitoringSystem</title>
    <style type="text/css">
        .style1
        {
            font-style: italic;
            color: #FF6600;
        }
        .style2
        {
            color: #000066;
        }
        .style3
```

```
        {
            color: #000000;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server" dir="ltr">
    <div>

        <h1 class="style1">
            <img alt="bannerimage" src="image/bannerimage.gif"
                style="width: 850px; height: 150px" /></h1>
        <h1 class="style1">
        Occoquan Watershed Monitoring System</h1>

    </div>
    <p>
         </p>
    <p class="style2">
        <span class="style3">Monitoring Station ID Number   </span>
        <asp:TextBox ID="StationName" runat="server" ></asp:TextBox>
        <span class="style3">    </span>
        <asp:Button ID="Button1" runat="server" Text="Submit" />
    </p>
```

```
<p class="style3">
    Monitoring Station Name          
    <asp:DropDownList ID="NameOfStation" runat="server" DataSourceID="StationsName"
        DataTextField="StationName" DataValueField="StationName">
    </asp:DropDownList>
    <asp:SqlDataSource ID="StationsName" runat="server"
        ConnectionString="<%$ ConnectionStrings:OWML_IndexConnectionString2 %>"
        SelectCommand="SELECT [StationName] FROM [Stations] WHERE ([StationID] = @StationID)">
        <SelectParameters>
            <asp:ControlParameter ControlID="StationName" Name="StationID"
                PropertyName="Text" Type="Int32" />
        </SelectParameters>
    </asp:SqlDataSource>
</p>
<p>
    Available Parameters 
    <asp:DropDownList ID="Param" runat="server" AutoPostBack="True"
        DataSourceID="Parameter" DataTextField="ParamID" DataValueField="ParamID">
    </asp:DropDownList>
    <asp:SqlDataSource ID="Parameter" runat="server"
        ConnectionString="<%$ ConnectionStrings:OWML_IndexConnectionString2 %>"

        SelectCommand="SELECT DISTINCT [ParamID] FROM [MasterData] WHERE ([StationID] = @StationID) order by ParamID">
        <SelectParameters>
```

```
            <asp:ControlParameter ControlID="StationName" Name="StationID"
                PropertyName="Text" Type="Int32" />
        </SelectParameters>
    </asp:SqlDataSource>
  </p>
  Begin Date 
  <asp:DropDownList ID="BeginDate" runat="server" AutoPostBack="True"
    DataSourceID="date" DataTextField="datetime1" DataValueField="datetime1"
    Width="170px" CausesValidation="True" CssClass="style3" Height="20px">
  </asp:DropDownList>
   End Date 
  <asp:DropDownList ID="EndDate" runat="server" AutoPostBack="True"
    DataSourceID="date" DataTextField="datetime1" DataValueField="datetime1"
    Height="20px" Width="171px">
  </asp:DropDownList>
  
   <asp:SqlDataSource ID="date" runat="server"
    ConnectionString="<%$ ConnectionStrings:OWML_IndexConnectionString2 %>"

    SelectCommand="SELECT DISTINCT [datetime1] FROM [MasterData] WHERE (([StationID] = @StationID) AND ([ParamID] =
@ParamID)) ORDER BY [datetime1]">
    <SelectParameters>
      <asp:ControlParameter ControlID="StationName" Name="StationID"
        PropertyName="Text" Type="Int32" />
```

```
            <asp:ControlParameter ControlID="Param" Name="ParamID"
                PropertyName="SelectedValue" Type="String" />
        </SelectParameters>
    </asp:SqlDataSource>
    <br />
    <br />
    <asp:GridView ID="GridView2" runat="server" AllowPaging="True"
        AllowSorting="True" AutoGenerateColumns="False" BackColor="#DEBA84"
        BorderColor="#DEBA84" BorderStyle="None" BorderWidth="1px" CellPadding="3"
        CellSpacing="2" DataSourceID="GridView"
        EmptyDataText="There are no data records to display."
        style="text-align: center" Width="654px">
        <RowStyle BackColor="#FFF7E7" ForeColor="#8C4510" />
        <Columns>
            <asp:BoundField DataField="StationID" HeaderText="StationID"
                SortExpression="StationID" />
            <asp:BoundField DataField="StationName" HeaderText="StationName"
                SortExpression="StationName" />
            <asp:BoundField DataField="ParamID" HeaderText="ParamID"
                SortExpression="ParamID" />
<asp:BoundField DataField="Units" HeaderText="Units" SortExpression="Units"></asp:BoundField>
            <asp:BoundField DataField="datetime1" HeaderText="datetime1"
                SortExpression="datetime1" />
            <asp:BoundField DataField="ValueN" HeaderText="ValueN"
```

```
            SortExpression="ValueN" />
          <asp:BoundField DataField="ResultID" HeaderText="ResultID"
            SortExpression="ResultID" />
        </Columns>
        <FooterStyle BackColor="#F7DFB5" ForeColor="#8C4510" />
        <PagerStyle ForeColor="#8C4510" HorizontalAlign="Center" />
        <SelectedRowStyle BackColor="#738A9C" Font-Bold="True" ForeColor="White" />
        <HeaderStyle BackColor="#A55129" Font-Bold="True" ForeColor="White" />
    </asp:GridView>
    <asp:SqlDataSource ID="GridView" runat="server"
 ConnectionString="<%$ ConnectionStrings:OWML_IndexConnectionString2 %>"
        ProviderName="<%$ ConnectionStrings:OWML_IndexConnectionString.ProviderName %>"

        SelectCommand="SELECT MasterData.StationID, MasterData.ParamID, MasterData.datetime1, MasterData.ValueN,
MasterData.ResultID, Stations.StationName, ParameterCodes.Units FROM MasterData INNER JOIN Stations ON
MasterData.StationID = Stations.StationID INNER JOIN ParameterCodes ON MasterData.ParamID = ParameterCodes.ParamID
WHERE (MasterData.StationID = @StationID) AND (MasterData.ParamID = @ParamID) AND (MasterData.datetime1 &gt;=
@datetime1) AND (MasterData.datetime1 &lt;= @datetime12) AND (MasterData.ResultID = @ResultID)">
        <SelectParameters>
          <asp:ControlParameter ControlID="StationName" Name="StationID"
            PropertyName="Text" Type="Int32" />
          <asp:ControlParameter ControlID="Param" Name="ParamID"
            PropertyName="SelectedValue" Type="String" />
          <asp:ControlParameter ControlID="BeginDate" Name="datetime1"
```

```
            PropertyName="SelectedValue" Type="DateTime" />
        <asp:ControlParameter ControlID="EndDate" Name="datetime12"
            PropertyName="SelectedValue" Type="DateTime" />
        <asp:Parameter DefaultValue="ok" Name="ResultID" />
    </SelectParameters>
</asp:SqlDataSource>
<br />
<br />
    Click Refresh Button to Get New Chart and Table<br />
<rsweb:ReportViewer ID="ReportViewer1" runat="server" BackColor="#FFFF99"
    BorderColor="White" DocumentMapCollapsed="True"
    Font-Names="Verdana" Font-Size="8pt" Height="586px" ShowBackButton="True"
    SizeToReportContent="True" Width="800px" CssClass="style1"
    DocumentMapWidth="50%" ExportContentDisposition="AlwaysInline"
    PromptAreaCollapsed="True">
    <LocalReport ReportPath="Report.rdlc">
        <DataSources>
            <rsweb:ReportDataSource DataSourceId="ObjectDataSource1"
                Name="DataSet2_MasterData" />
        </DataSources>
    </LocalReport>
</rsweb:ReportViewer>
<asp:ObjectDataSource ID="ObjectDataSource2" runat="server"
    OldValuesParameterFormatString="original_{0}" SelectMethod="GetData"
```

```
    TypeName="DataSet2TableAdapters.MasterDataTableAdapter">
    <SelectParameters>
      <asp:ControlParameter ControlID="StationName" DefaultValue="12"
        Name="StationID" PropertyName="Text" Type="Int32" />
      <asp:ControlParameter ControlID="Param" DefaultValue="do" Name="Parameter"
        PropertyName="SelectedValue" Type="String" />
      <asp:ControlParameter ControlID="BeginDate" DefaultValue="09/1/2000"
        Name="BeginDate" PropertyName="SelectedValue" Type="DateTime" />
      <asp:ControlParameter ControlID="EndDate" DefaultValue="12/1/2000"
        Name="EndDate" PropertyName="SelectedValue" Type="DateTime" />
    </SelectParameters>
</asp:ObjectDataSource>
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
    OldValuesParameterFormatString="original_{0}" SelectMethod="GetData"
    TypeName="DataSet2TableAdapters.MasterDataTableAdapter">
    <SelectParameters>
      <asp:ControlParameter ControlID="StationName" DefaultValue="12"
        Name="StationID" PropertyName="Text" Type="Int32" />
      <asp:ControlParameter ControlID="Param" DefaultValue="do" Name="Parameter"
        PropertyName="SelectedValue" Type="String" />
      <asp:ControlParameter ControlID="BeginDate" DefaultValue="09/1/2000"
        Name="BeginDate" PropertyName="SelectedValue" Type="DateTime" />
      <asp:ControlParameter ControlID="EndDate" DefaultValue="11/1/2000"
        Name="EndDate" PropertyName="SelectedValue" Type="DateTime" />
```

```
        </SelectParameters>
    </asp:ObjectDataSource>
    <br />
    <br />
    <a href="MapPage.htm">Back to MapPage</a>
    </form>
</body>
</html>
```