MyOceandataSQL
Users' Manual

Oceandata.sql

Oceandata Upload Application (OUA)

Oceandata Web Application (OWA)

Authors (equal contributions):

Damian Castro (project engineer & builder)

damianocastro@hotmail.com

Sebastien Donnet (project architect)

sebastien.donnet@dfo-mpo.gc.ca

Andry Ratsimandresy (project director)

andry.ratsimandresy@dfo-mpo.gc.ca

Contents

Introduction

The present document is a user manual for MyOceandataSQL, a suite of applications to store, manage and retrieve oceanographic data. The suite consists of 2 applications, an Oceandata Upload Application (OUA) and an Oceandata Web Application (OWA) which interface with a MySQL database (Figure 1). Using screenshots, this document will guide a user to install the applications and to upload and download data.

This document contains two sections. The first section is an installation guide of the suite and the second section is a step-by-step usage instruction of the applications. A description of the configuration file (XML language) used by the OUA to transform and convert input datafiles to the database is provided in Appendix 1 for users interested in more details and/or for customization.

MyOceandataSQL was designed to store and manage a large variety of geographically defined data such as profiles and fixed point timeseries as well as moving measurements; CTD profiles, moored ADCP and Drifters, respectively. To do this, a set of key data and metadata needs to be available, in a specific format, from the file to be uploaded: **4 metadata variables (Station, Instrument, Longitude(DecDeg) and Latitude(DecDeg) and 2 measurement variables (Time(ISO8601) and Depth(m))**. The names of those variables should be provided as spelled or will need to be mapped otherwise (see section 'Using the Oceandata Upload Application' for details).

Two main types of installations were tested and are presented in this document: a local installation using an MS Windows server and clients and an installation using a Linux server. Users whishing to install are advised to read the entire 'Installation' section prior to make a choice on the tools to use and to better understand the system dependencies needed. Users already familiar with MySQL and PHP web services will probably be familiar with those or other tools that would work with/for MyOceandataSQL.

All the files needed for an installation and testing (example files) of the suite are available at the following GitHub repository: https://github.com/damiancastro/MyOceanDataSQL1.1 and a working example of the web-application is available at http://www.damiancastro.ca/myoceandatasql/.

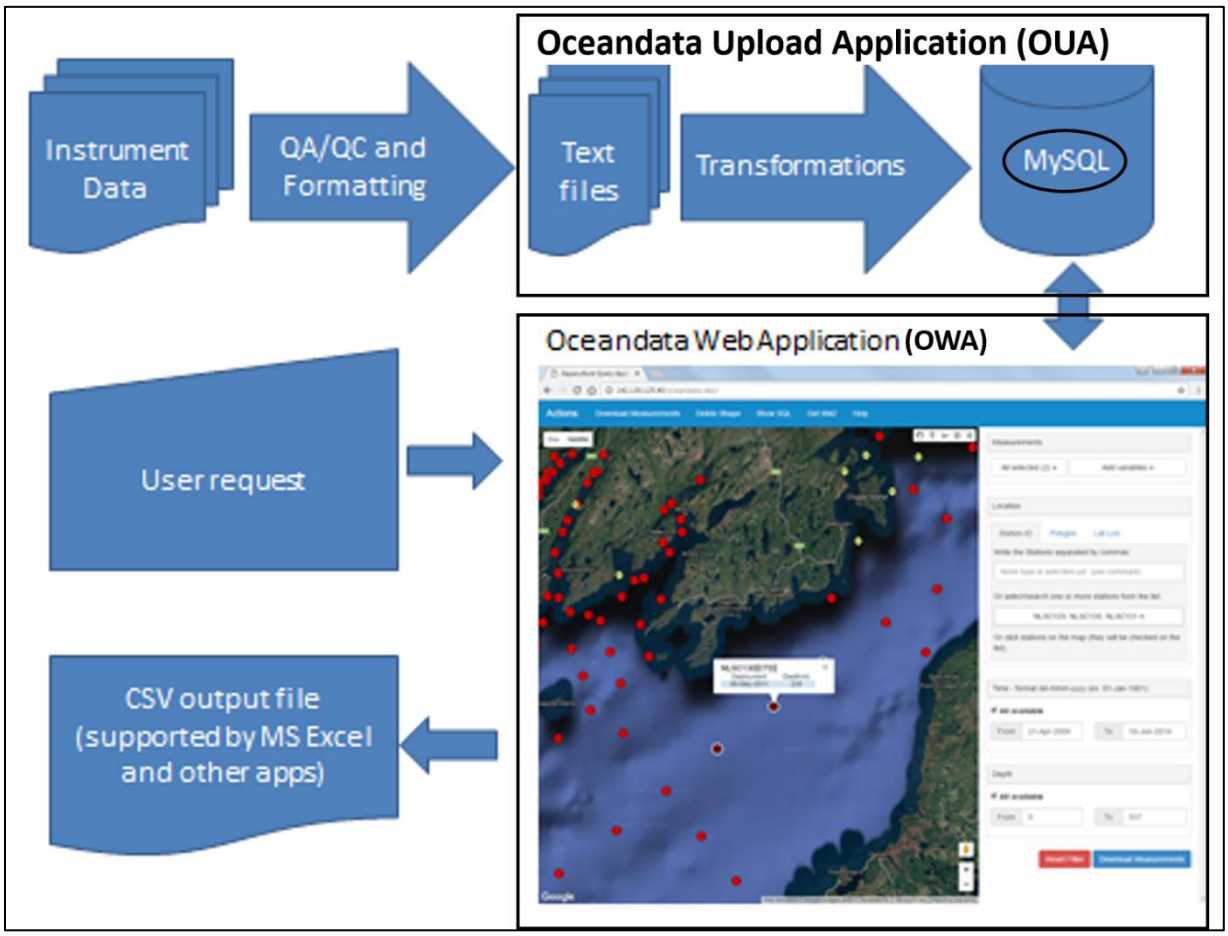


Figure 1: Processes and applications of MyOceandataSQL suite (source: Donnet et al. 2021, in submission to Oceanography)

Installation

MyOceandataSQL suite needs 3 components to be installed for it to work: the Oceandata SQL database, the OWA and the OUA. The database is installed in a MySQL server that can either be Linux or MS Windows-based. The OWA has to be installed in a Web server that can also be based on a MS Windows or Linux box. OWA and MySQL have to be installed in the same server for OWA to have access to the data files created by MySQL. The available current version of OUA only runs on a MS Windows environment and was tested with MS Windows 7 and 10. The installation of the database and OWA were tested on both Linux and MS Windows (7 and 10) systems (see details below). MySQL version needs to be 5.7 or higher and the web server needs to be able to handle PHP. You will need to have root or administrator privileges to be able to install most of the applications.

Creating the database

MySQL (v5.7 or higher) needs to be installed and run before creating the database. The script *oceandata.sql* is used to create all the database objects needed to run OWA and OUA. The script can be opened and should be able to run on any MySQL query tool such as MySQL Workbench or HeidiSQL (see

local MS Windows installation below). It can also be run from the Linux command line (see Linux server installation below).

The creation of the database objects should be done carefully. The scripts will create a database called *oceandata* if the database does not exists. If an '*oceandata*' database exists, all MyOceandataSQL objects (tables, routines, etc.) will be deleted and recreated. Thus, any data tables or any changes done to previous database objects will be lost. It is recommended to make a backup of any existing "oceandata" database before running the ocenadata.sql script (see Appendix 2 "Backup and restore" for details).

Once the script is run, the database should contain 3, empty, tables: a "metadata" table, a "measurement" table and a "files" table. The metadata table will include the mandatory metadata associated with the measurements, i.e. **Station**, **Instrument**, **Longitude(DecDeg)** and **Latitude(DecDeg)**, as well as a few more that are created upon upload and that are mainly used to help the user navigate and search via the OWA (see details in 'Using the Oceandata Web Application' section). Those latter variables are **InstrumentType**, **Position**, **DeployTime(ISO8601)**, **RecoverTime(ISO8601)**, **MaxDepth**, **file** and **UploadTime**. **InstrumentType** is defined by the user during upload via the OUA (details in section 'Using the Oceandata Upload Application'). **Position** is derived from **Longitude(DecDeg)** and **Latitude(DecDeg)**, **DeployTime(ISO8601)** and **RecoverTime(ISO8601)** are derived from the mandatory measurement field Time(ISO8601) while **MaxDepth** is calculated from the field **Depth(m)**. The fields **file** and **UploadTime** are filled with the filename of the uploaded file and date/time of the upload. The measurement table will contain the two mandatory fields **Time(ISO8601) and Depth(m)** as well as any other variables created during upload processes (see details in section 'Using the Oceandata Upload Application'). Finally, the files table will contain the filename (including the full path), folder (of origin) and filename (without path) of each of file uploaded. To link those tables together, an **idEvent** field will be added to the metadata and measurement tables. A field **id** and **idFiles** will also be added to the measurement and files tables, respectively.

NOTES:

- the name of the database to be created can be changed by editing the script "*oceandata.sql*" and changing the database name (e.g. *new_oceandata*), there are only 2 instances located at line 1 and 2 of the file.
- OWA needs MySQL to dump files in an export folder that both OWA and MySQL has read and write access to. By default, an export folder will be created during an installation of MySQL, set in the system variable "*secure_file_priv*". This folder's location can be find by typing **show variables like "secure_file_priv"** from mysql prompt. In our installation on a Linux server (details below) this folder was automatically set to **/srv/www/htdocs/oceandata/files/**. On an MS windows local server using WAMP (details below), this folder was set to **pathtoWAMP\tmp** by default. This is an important security measure since Mysql will not be allowed to write to any other folder in the server.
- **The WAMPserver presented below (local MS Windows install) uses a blank password for the root user. This is not a safe option; only showed and used here for simplicity though we do not**

recommend it. The best way to secure MYSQL is to use a password for root and not use it for both OWA and OUA applications. Then, create a user that has Select, Insert, Update and delete access only to the database's objects. See Mysql documentation on how to create a user and grant the required level of access to objects.

Installation on a local MS Windows computer using WAMPserver and HeidiSQL

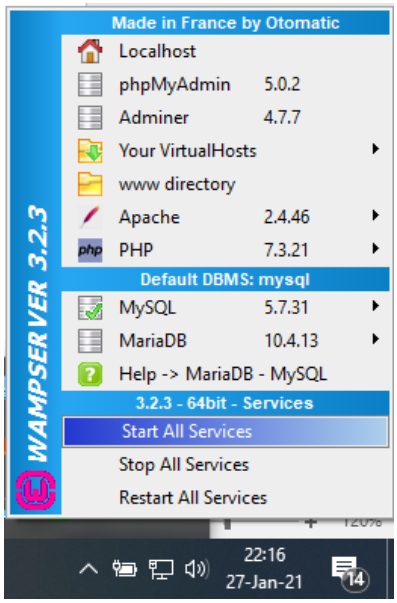
This installation creates a 'local' system, i.e. a database that can only be run and accessed locally.

MyOceandataSQL suite should work with any web server with access to MySQL and PHP. We (most recently) tested the suite using WAMPserver64 3.2.3 (64 bits; August 2020) which comes with MySQL 5.7&8.0 and PHP 5&7 (http://www.wampserver.com/en/). Before the installation of WAMPserver, one needs to ensure that MS Visual C++ runtime libraries (VC) are also installed; they are required dependencies. With WAMPserver64 3.2.3, libraries VC9 (2008), VC10 (2010) and VC11 (2012) are required when using default settings, i.e. PHP v5.6.4 and Apache 2.4.26. If you want to use PHP v.7.x and/or higher version of Apache you will need VC13 (2015) and VC14 (2017) also. **NOTE:** you will need to install both the x86 and x64 versions of VC to run WAMPserver64. To check what is already installed on your system, WAMP also provides a tool "check_vcredist.exe". This tool, along with the above VC dependencies and WAMPserver64 3.2.3 itself are provided from the wampserver website (see details in our README.md including a quick install guide on our GitHub page). We installed WAMPserver under C:\wamp64 to be as close as the root as possible as well as being somewhere with read&write rights, as advised. **NOTE**: We tested other places such as C:\users\ourHome\WAMP and run into (OUA communication) issues.

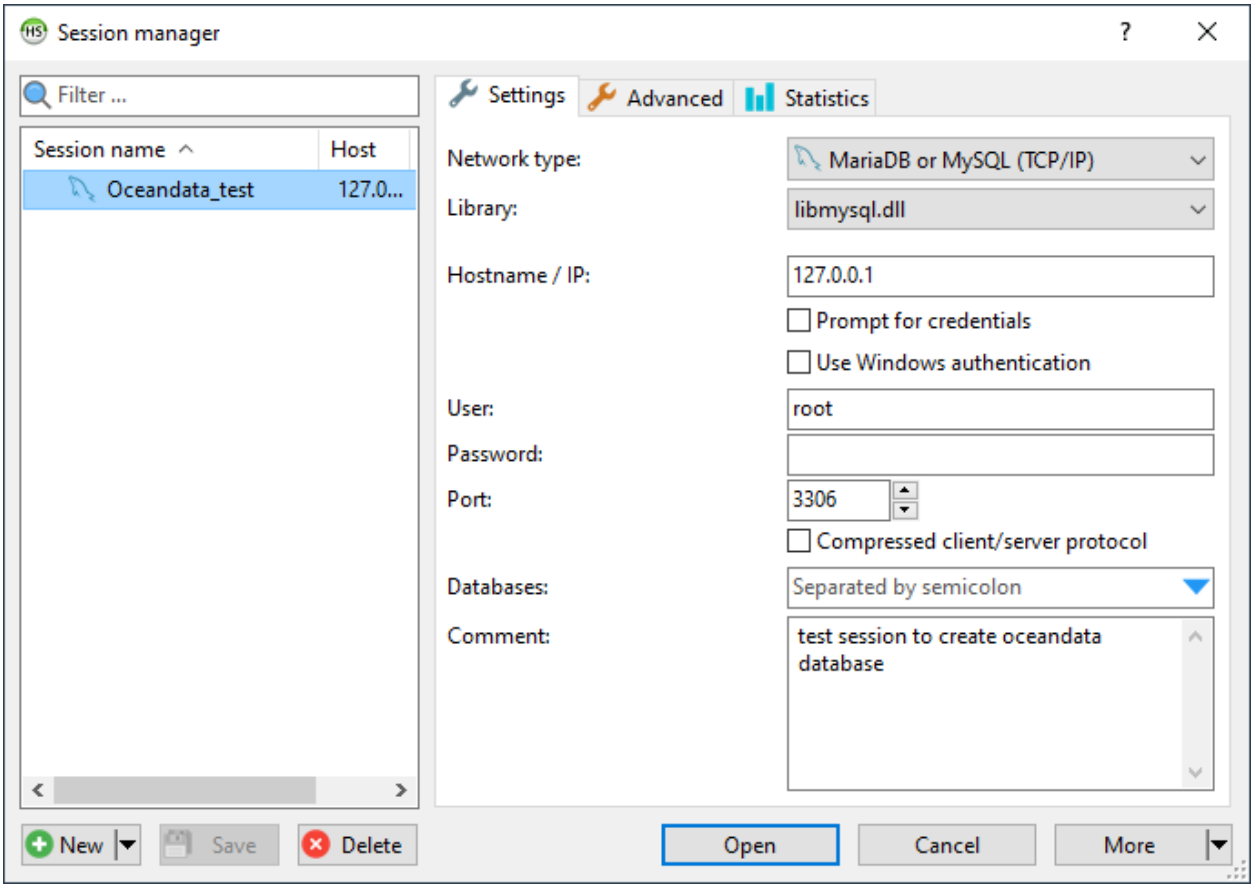
Once WAMPserver is installed, you may install HeidiSQL to communicate more easily with MySQL server. HeidiSQL can be found at https://www.heidisql.com/ (download installer). On our windows 10 test machine, we installed the latest version available at the time (v11.2) and followed all the defaults settings. The link to executable is also available in our README.md. **NOTE:** HeidiSQL is not absolutely necessary, one can install/run oceandata.sql directly from the MySQL prompt (console) available in WAMP (see more details on the commands in "Installation on a Linux server" section, below). However, we found this (fairly light) tool useful not only to run the setup script but also to examine our database going along with uploads.

To setup the server and install oceandata, one needs to:

1) launch WAMPserver (as admin) [desktop icon and/or quicklaunch]. **NOTE:** Icon (in quick launch) goes from red to orange and then green. If it doesn't, then you need to do "start all services" (it may takes a few seconds or a bit more).

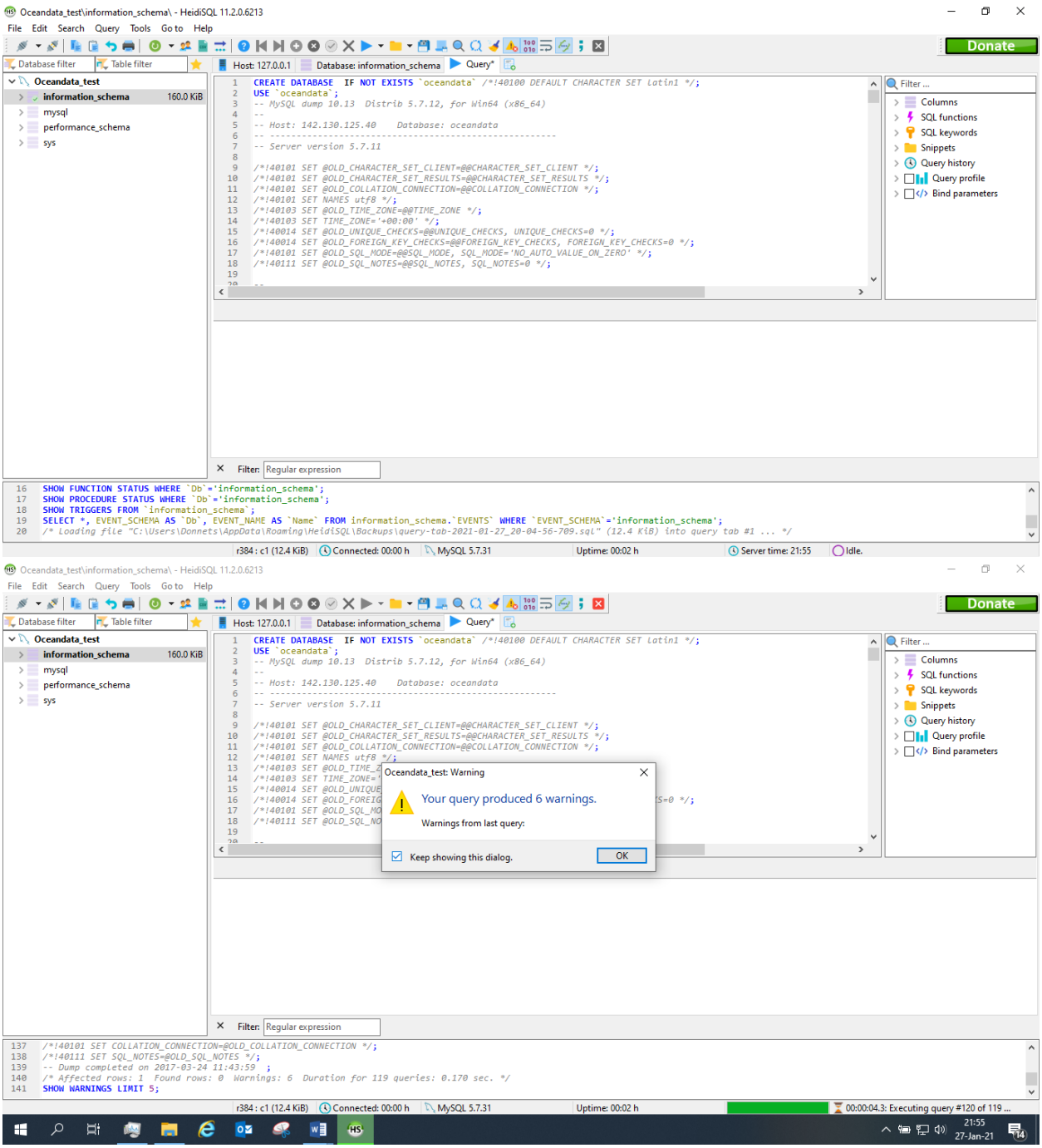


2) Open HeidiSQL and initiate a session; you may enter a name of your choice for it (right-click renamed) but you must leave user as “root” and password blank (default WAMP-MySQL settings). On our system, port was set as “3306” by default (also by WAMP).

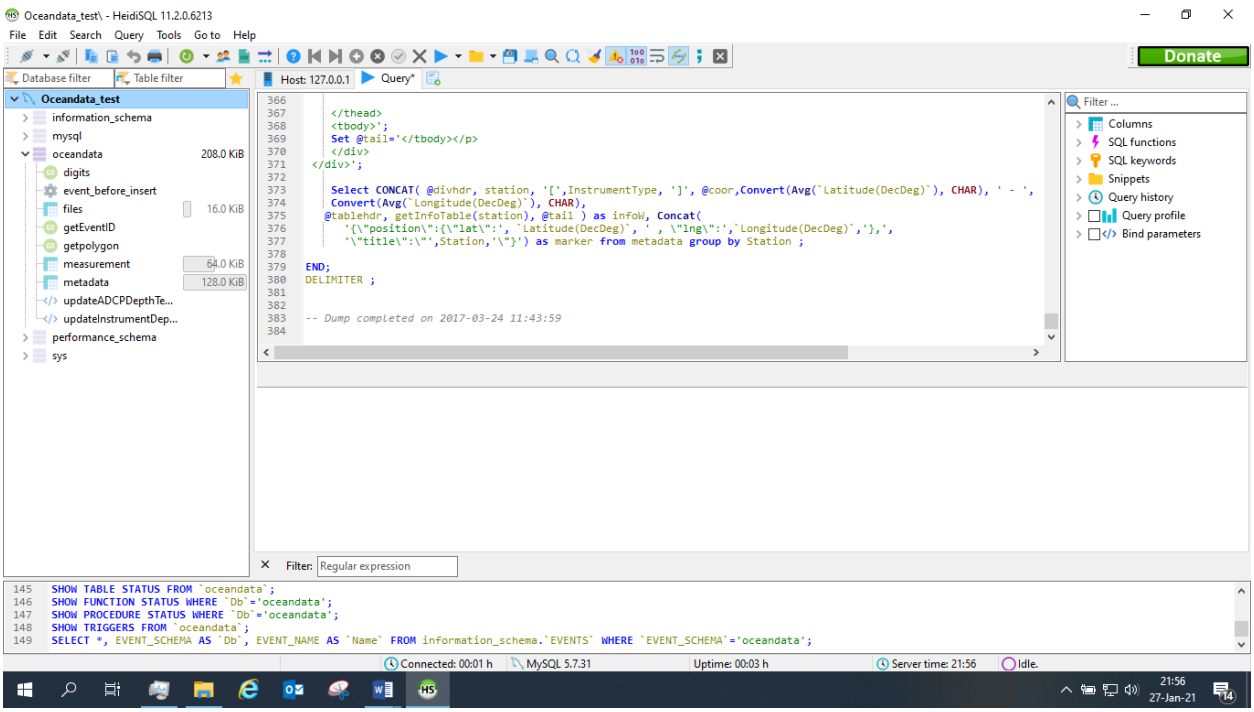


Make sure you don’t have any database name listed in the “Databases” box then click on “Open”

3) Once in HeidiSQL's main window click on query and add the content of your oceandata.sql file within the "Query" subwindow and click on the blue arrow or F9 to run the set of commands (alternatively, you may also do "open" which will then run the script automatically). **NOTE:** a WARNING window might open although the database is created; you can ignore it.



4) Then, you may refresh (green arrow curl or F5) and you should see a database "oceandata" appearing (**NOTE:** we, sometimes, had to close and re-open Heidi for oceandata to appear):



At this point, you are done creating the database shell and can move on to import data via OUA.

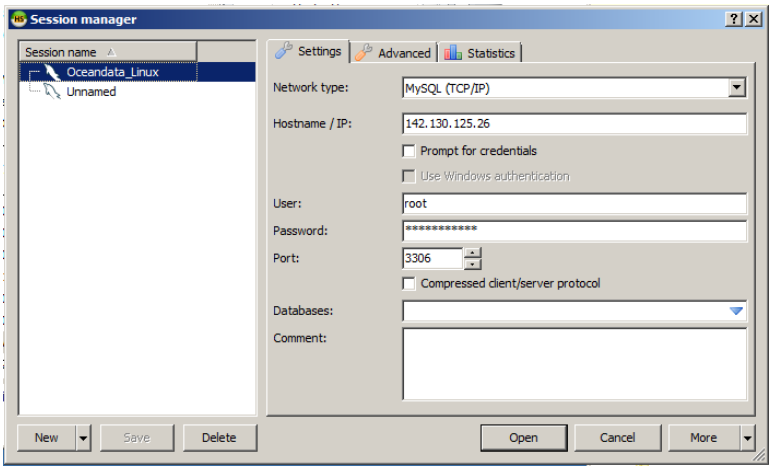
Installation on a Linux server

This installation creates a 'remote' system, i.e. a database that can be accessed remotely.

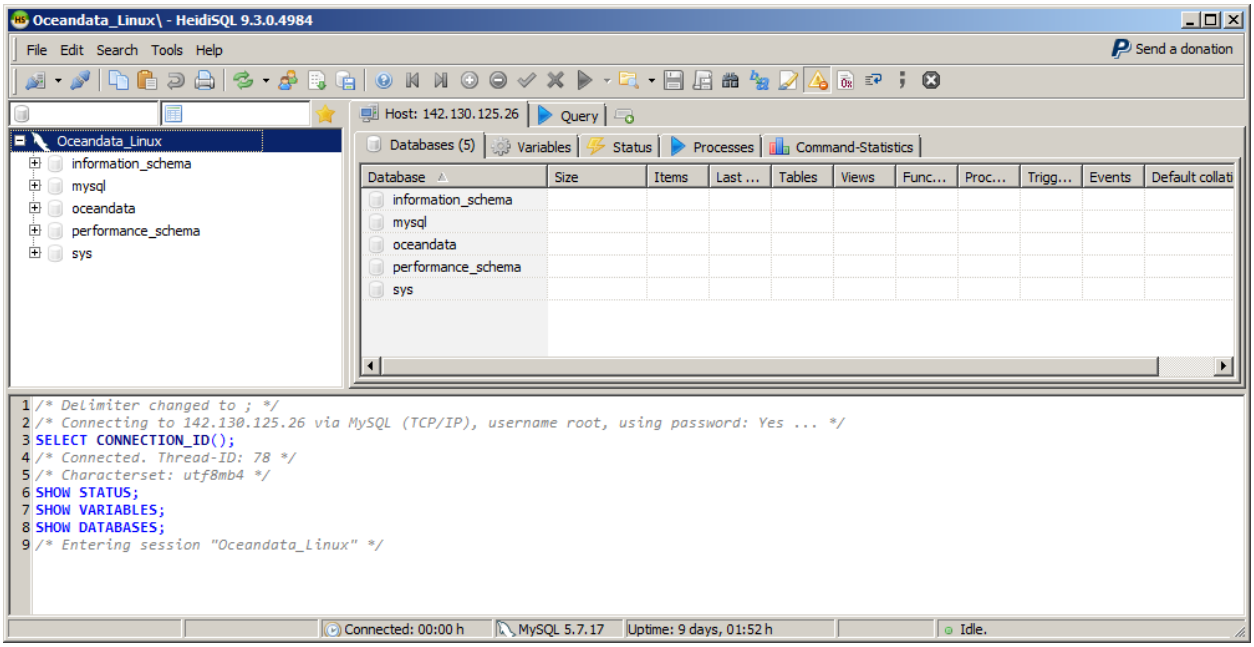
You must first install MySQL on your Linux system and have your computer/server connected to a LAN and/or WAN. On debian-based distribution (e.g. Ubuntu and derivatives) one can install MySQL via the command **apt-get install mysql-server**. **NOTE**: you will need version 5.7 or higher in order to install oceandata (below); you may check this using **mysql --version** from the bash prompt.

Once MySQL installed, one can install the database by running oceandata.sql from the bash prompt: **mysql < oceandata.sql**. **NOTE:** you may need to call on user root to do that such as **mysql –u root –p < oceandata.sql**

Alternatively, you can also run the script from a MS Windows machine connected to the Linux server via HeidiSQL as explained above. Here below a few snapshots illustrating this:



NOTE: this time, user root was assigned a password during MySQL installation



Some useful MySQL commands (i.e. work from any mysql prompt):

show databases;	% list databases
use oceandata;	% select oceandata database to query
show tables;	% show tables of a database
drop database oceandata;	% remove a database
show variables like "secure_file_priv"	% show path of import/export folder

Installing OWA

The installation process is the same for any web servers given that it only requires copying all OWA files to a desired location. Files may be modified at user will to customise the application.

On a local MS Windows box running WAMPserver, the set of OWA files can be copied in *WAMPserver_installation_folder**www**oceandata*\ (where "WAMPServer_installation_folder" is the folder where the server is installed). Access will then be at

http://localhost/oceandata/

On a Linux openSUSE 3.2 running Apache web server, for example, all the files can go to a subfolder oceandata in */srv/www/htdocs/oceandata/*. Thus, the application will be accessed at

http://server-ip-or-name/oceandata/

In order to connect to the database, OWA uses the information included in the **configuration.php** and **index.php** files. These files are located in the root of the OWA folder.

The name of the database needs to be the same in both **configuration.php** and **index.php** files (two instances of "oceandata.measurement" in **index.php** and one instance of "oceandata" at "$database" line in the **configuration.php** file).

The **configuration.php** file contains a few parameters to set location and the credentials of the database, e.g. (WAMP):

```
<?php
    //Database connection variables. Change them when changing database and/or server.
    $servername = "localhost";
    $username = "root";
    $password = "";
    $database = "oceandata";
    $mode='intranet';
?>
```

e.g. (Linux):

```
    $servername = "142.130.125.26";
    $username = "query";
    $password = "YourPassword";
```

$database = "oceandata";

Server name can be the ip address or the server name of a network (for example the dns name). Username is the user that will log on MySQL server to download data. This user needs 'select and execute rights' to create the file to be downloaded. This needs to be done from mysql (as mysql root user) using the 'information_schema' database:

mysql> CREATE USER 'query'@'localhost' IDENTIFIED BY '***Your_Password_in_configuration.php***';

mysql> use information_schema;

mysql> GRANT SELECT, EXECUTE ON `oceandata`.* TO 'query'@'%';

In the above case, 'query' user has been given permission only from 'localhost' and "***Your_Password_in_configuration.php***" is the password you would have set in the **configuration.php** file. For the database to be accessed from other machines, one may need to change 'localhost' with other syntax.

NOTE: if user "query" already exists in the system, do (from mysql prompt):

mysql> Select * from mysql.user; % to check and run

If 'query' is present it has to be dropped using the following set of commands:

mysql> DROP user 'query'@'localhost';

mysql> FLUSH privileges;

NOTE: on WAMP, we could not get the user 'query' to write data; to make it work we set the root as the user (**configuration.php**) though not being the safest option (**i.e. use at own risk**):

To make download work, user needs to also edit the **download.php** script to indicate the right path where data will be going:

$file="C:/wamp64/tmp/data".TIME().".csv"; // e.g. using WAMP server

$file="/srv/www/htdocs/oceandata/files/data".TIME().".csv"; // e.g. using Linux server

Installing OUA

To install OUA, one needs to create a folder with read, write, and execute access and copy the executable **OceanData+.exe** with its configuration file (**config.xml**). E.g. on our local MS Windows install, we placed them in C:\Users\Username\Download\OUA. Some dependencies are required: Visual C++ Redistributable for Visual Studio 2015 (VC13 x86 ; 32 bits) and MySQL ODBC 5.3 **32 bits** driver. A copy of

VC13 x86 is provided in the OUA folder of our GitHub repository (https://github.com/damiancastro/MyOceanDataSQL1.1) and it can also be downloaded from Microsoft's website: https://www.microsoft.com/en-ca/download/details.aspx?id=48145 (download the **x86 32 bits version**). **NOTE:** higher version of VC (e.g. 14 & 15) are backward compatible with VC13 and the application has been tested on different machine running Visual Studio 2015 or newer.

A copy of MySQL ODBC 5.3 32 bits driver is provided in the OUA folder of our GitHub repository and can also be downloaded from https://dev.mysql.com/downloads/connector/odbc/. In most cases, you should install this version which will work by default with the OUA's executable provided in the repository. During install, you may select a "Typical" installation (default) which will work with OceanData+.exe.

The exact name of the above driver has to be specified in the configuration file config.xml to match what the user has installed in the computer running OUA. OUA was tested with MySQL ODBC 5.3 and Microsoft.ACE.OLEDB.12.0 (provided by default MS Office applications). To check which version of ODBC you have, you may simply used native MS Windows application program "Apps & Feature":

Checking Microsoft.ACE.OLEDB version is more tricky, one may try this page for help https://www.codeproject.com/Tips/508868/Programmatically-check-Access-Database-Engine-for for help (we ended-up not having to try on any of our computers). Though not tested with others, it is expected to work with other versions of the same drivers. **NOTE:** an external user reported us the need to install MS Office Access engine (AccessDatabaseEngine.exe; available from Microsoft's website, https://www.microsoft.com/en-ca/download/details.aspx?id=13255 for 2010 version) for OUA to work; maybe due to variant MS Office installation and/or version.

By default, the OUA comes with the following values in the configuration file:

Destination Database (WAMP install example):

```
<odbcparams driver="{MySQL ODBC 5.3 Unicode Driver}"
server="localhost" database="oceandata" user="root"
password=""></odbcparams>

<mysqlConn server="localhost" database="oceandata" user="root"
password="">Default</mysqlConn>
```

Source file (WAMP install example):

```
<oledbparams>Provider=Microsoft.ACE.OLEDB.12.0;Extended
Properties='Text;HDR=Yes;FMT=Tabdelimited;';Data Source=</oledbparams>
```

Using the Oceandata Upload Application (OUA)

NOTE: to get a new user started, example files are provided in our GitHub repository (https://github.com/damiancastro/MyOceanDataSQL1.1; examples folder)

OUA has one window with four tabs: **Database**, **File**, **VarMap**, [***source data filename***] and **log**. When the application starts, only two tabs are shown: **Database** and **File** (Figure 2).

The database tab is used to set the database connection parameters: the driver, server, database user and password (all of those specified in the config.xml file).

Driver: This document assumes that the database server is MySQL 5.7 and that the ODBC driver information matches that MySQL version (see Installing OUA section and config.xml for more details).

Server: This is the network address of the database application. It can be an IP or a DNS name.

Database: This is the database or schema to connect to.

User and **Password**: These are credentials to access the database server.

Once OUA connects to the database it will list the variable existing in the database in the metadata and measurement tables (Figure 2).

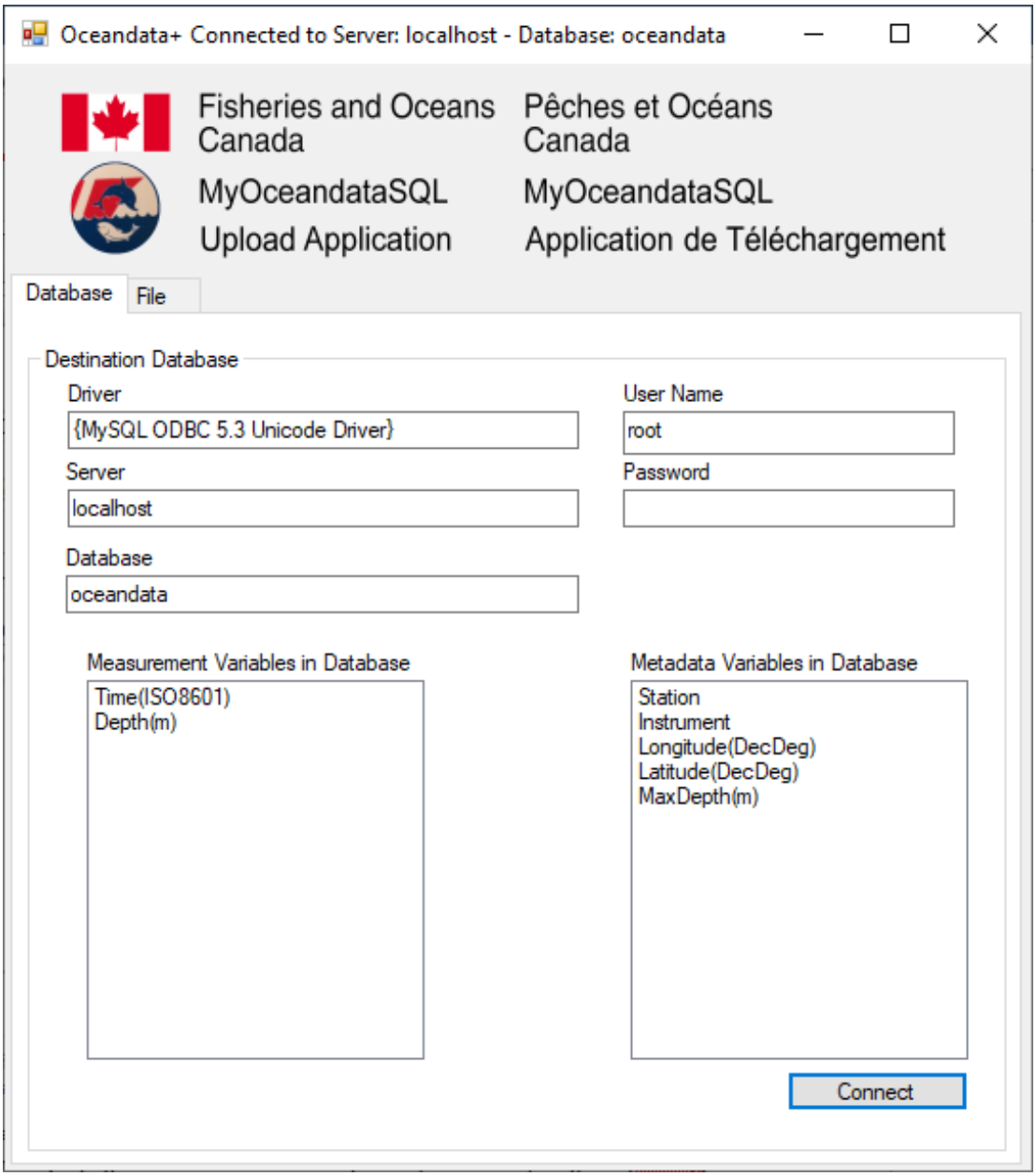


Figure 2: Database Tab (from MS Windows WAMP system)

There are 6 mandatory variables, set in the default config.xml file and database, to be provided in any file to upload: 2 measurement variables (Time(ISO8601) and Depth(m)) and 4 metadata variables (Station, Instrument, Longitude(DecDeg) and Latitude(DecDeg). If those key variables are not available in the file, the OUA will not proceed. NOTE: variable mapping (see details below) can be used if you chose not to use the same exact names (e.g. Time vs. Time(ISO8601) or Lat vs Latitude(DecDeg)) but their format must be as expected (details there after). "Station" can be used to store an institute or cruise site/station identifier (string, 30 characters max) and "Instrument" to store the make, model and serial number (e.g. SBE19#6524, string also; 50 characters max). "Longitude(DecDeg)" and "Latitude(DecDeg)" are deployment positions where data were collected, it must be given in decimal

degrees (DecDeg). “Time(ISO8601)” is the measurement time and must be given in yyyy-MM-ddThh:mm:ss ISO8601 format (ISO, 2004) with yyyy being the 4-digit year, MM the 2-digit month, dd the 2-digit day, hh the 2-digit hour, mm the 2-digit minute and ss the 2-digit second. Depth can be given in any unit (e.g. meters or feet) but will need to have its field name edited in the config.xml and oceandata.sql if using something-else than meters (e.g. Depth(ft) if/when using feet) and, needless to say, it *must* stay consistent among all data provided. In addition to these mandatory variables any other additional variables can be added such as Temperature, Salinity and Dissolved Oxygen as measurement variables and/or any other metadata desired (e.g. cruise name/ID). **As a good practice, though not mandatory, it is recommended that corresponding units be indicated in bracket with the variable name (e.g. Temperature(oC)) as this will help the user downloading the data.**

The **File** tab contains information on the file to upload and the initial parameters for uploading: the instrument type and the file location (Figure 3). An ‘**Autodetect Instrument Type**’ box can also be used/checked if the instrument type information is available in the input file. **NOTE:** this feature is obsolete and will/should be removed in the future. . The file location can be typed, copied and pasted, or selected using the browse button. The **Load** button will open the file, check all the variables that are included in it and compare them with the variables listed in the database. **NOTE:** this may take a few seconds to proceed (the larger the file, the longer). All the variables included in the file will appear in the “**Variables in File**” panel. Any new variable(s), which is a variable that is in the file but not yet in the database will appear in the “**New Variables**” panel. Any new variable can be either created or mapped to existing database fields (see next section). The tab also provides an “**Update Batch**” feature which allows the upload of a batch of files contained in a folder. When choosing this option, you will need to load a file first, e.g. first one of the series to be uploaded, and you will then be prompted to choose the folder and all the files listed in that folder will be processed upon hitting “**Update Batch**”. **NOTE: use this option to upload file of the same type only (i.e. a bunch of CTD or ADCP files that have been processed/exported the exact same way with the same number, and name, of variables). NOTE also that you should check the VarMap tab BEFORE hitting the Update Batch button, otherwise any necessary mapping or variable creation will not occur.** The format of files to upload must be tab-separated text files though different format can be used by editing the config.xml file accordingly (see details in Appendix 1). The files **must not** have dots in the file name. First raw **must have** variables names as an header followed by the data (from second raw).

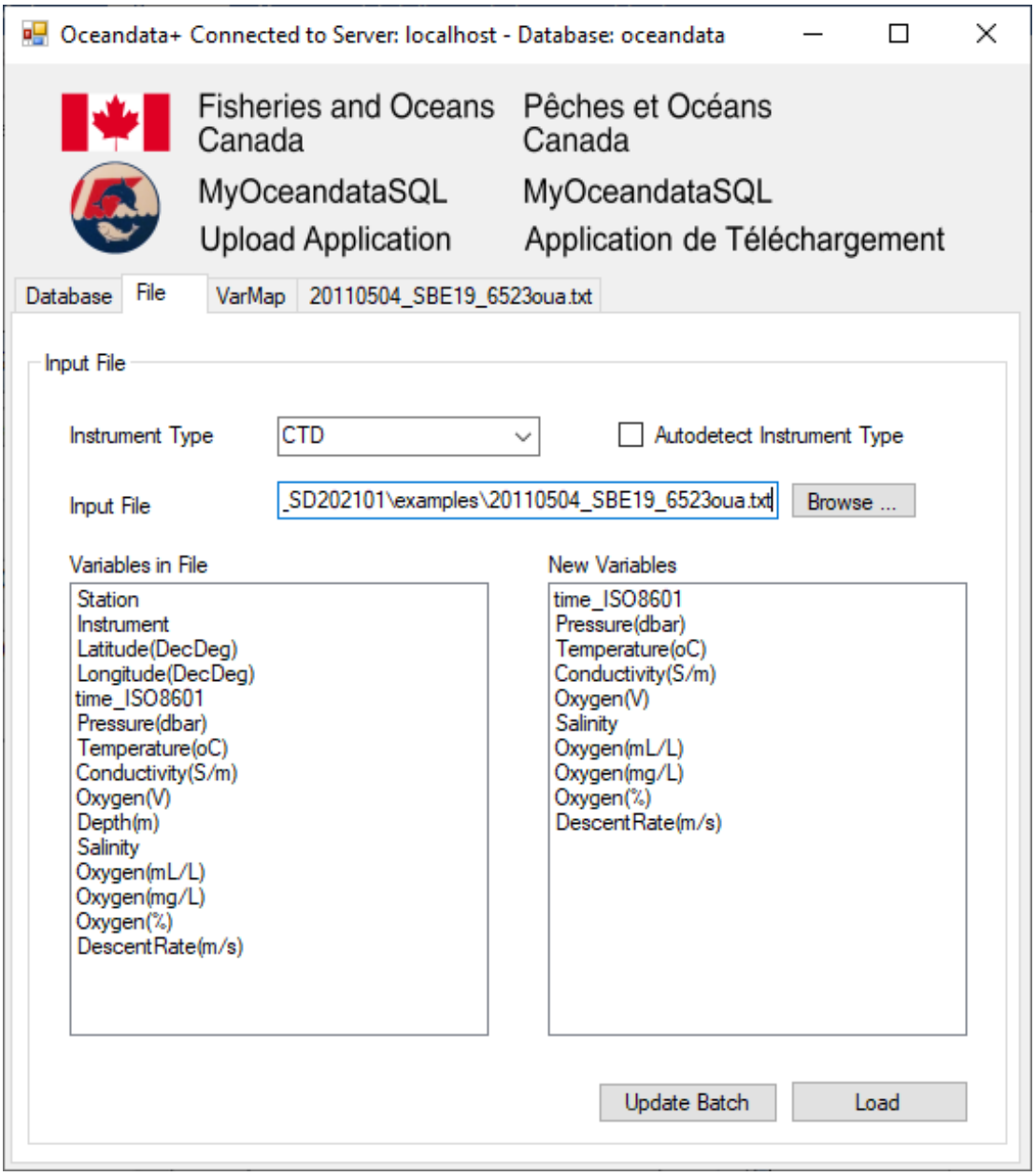


Figure 3: File tab (note: in this example, the field time was labeled as time_ISO8601 which will require mapping)

The **VarMap** tab contains mapping information/parameters to create and/or map variables from uploaded file(s) to the database (Figure 4). Mapping information/parameters can also be accessed (and edited) in the configuration file (config.xml). In order to be uploaded, each file variable has to be either set as "**New Variable**", mapped to a database variable from one of the two database main tables (metadata or measurement) or **ignored** (default). New variables will have to be provided with a "**Data type**" which can either be "**Text**" (30 characters max), "**Number**" or "**Date Time**" ("%Y-%m-%dT%H:%i:%s" format where Y is a four digits year, m is a two digits month, d is a two digits day, the letter T indicating that time follows, H is 24hr format hour, i is a two digits minutes and s is a two digits seconds; e.g. 2018-12-31T12:35:45). When a new variable is being requested it will be created both in

the configuration file and in the database. **Once edited as wanted, user must hit "Save mapping" button to have it processed. NOTE: this make takes a few seconds to take effect; particularly if the file is large and/or many new variables have to be created**

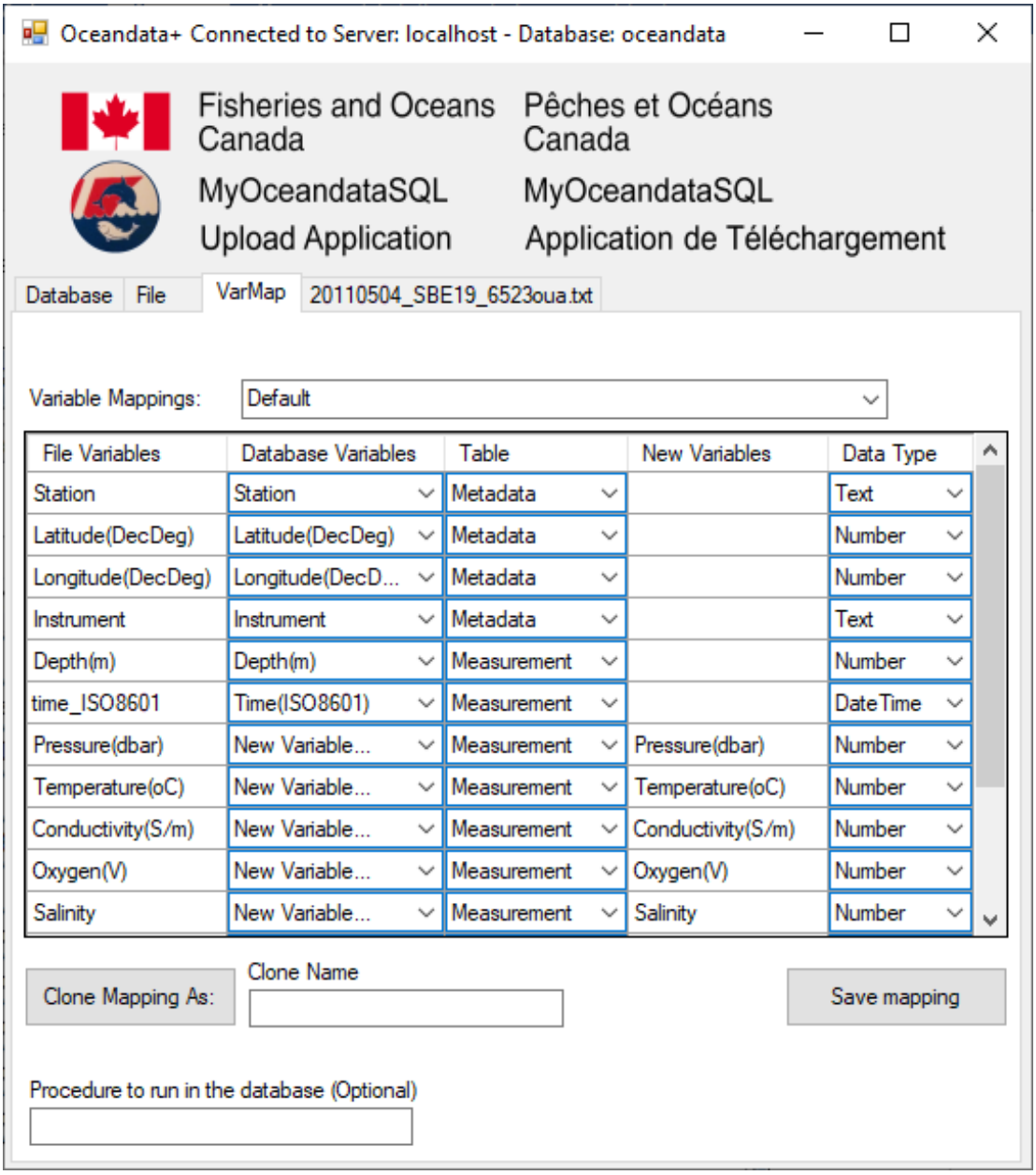


Figure 4: Variable Map Tab with time_ISO8601 mapped as Time(ISO8601) and new variables created

The source data file tab will display the input data using the filename as tab name (Figure 5). It allows inspecting the file that is going to be loaded. Thus the database updating process sequence has two steps. First, the file is loaded to the application where it can be inspected. Second, the data is appended to the database by hitting the "**Update**" button. When update batch is chosen (see above) the two steps happen at once.

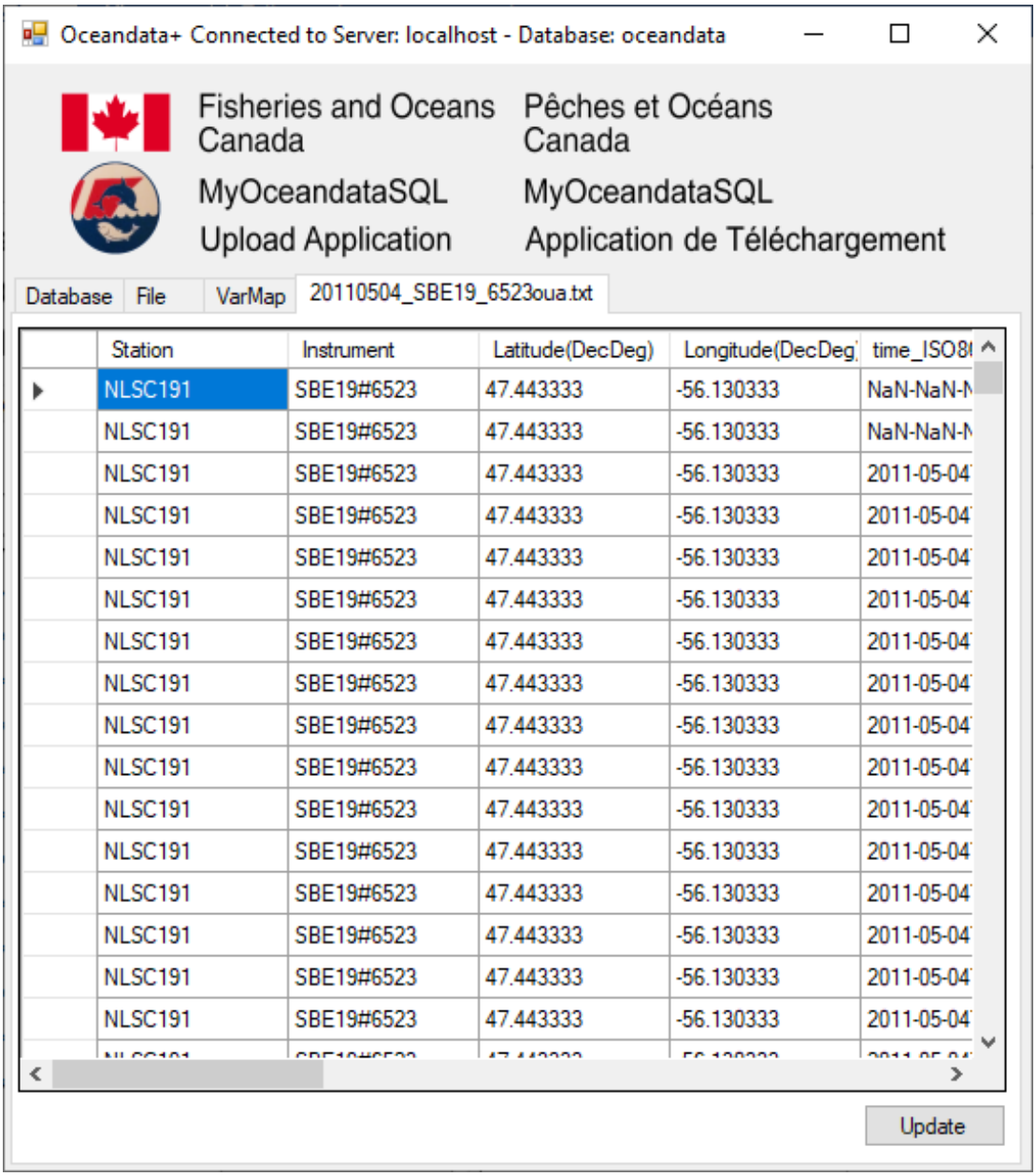


Figure 5: Data source file

Once the update or the update batch is launched, the **Log** tab (Figure 6) is displayed in the screen and OUA will log all the messages that are produce while uploading. These messages include time, rows updated in the database's metadata table, rows updated in the database's measurement table.

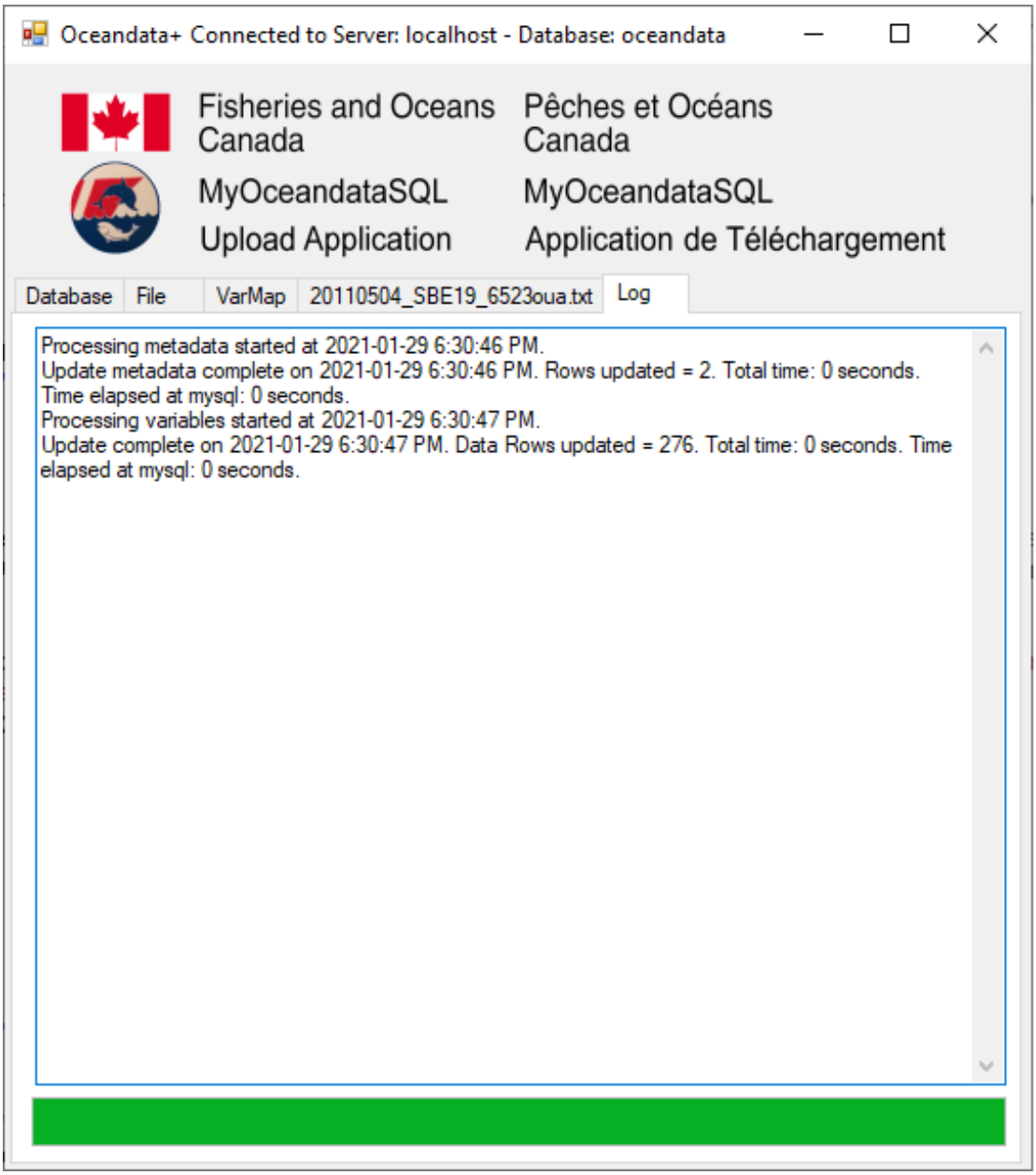


Figure 6: Log tab

Expected behaviour

1. An error happens when the file does not have one of the mandatory fields. The error will contain the name of field. If many are missing, only the first will be shown:

   ERROR [HY000] [MySQL][ODBC 5.3(w) Driver][mysqld-5.7.11]Column 'Station' cannot be null

2. If right after the process starts no file name is logged it is likely that file has not the right format or is empty. The following error also might appear:

   “Object reference not set to an instance of an object.”

3. The application will send error message indicating that it cannot access the log file if <odbcparams … </odbcparams> and <mysqlConn … ></mysqlConn> differs in settings. This error might comes out as well when WAMP server is down or after a windows update messing up with WAMP. In the latter case, you'll probably need to re-install WAMP to make it work again
4. An error message "Instrument Type cannot be blank" if the Input File box (File tab) is left blank before hitting the "Update Batch" button.
5. We found best to close the application before doing another upload; thus effectively resetting the input information

Using the Oceandata Web Application (OWA)

Once data is in the database, the Oceandata Web Application (OWA) can be used to select, filter and download measurements in comma-separated files. OWA consists of an interactive map (Google Earth API) on the left, several input controls on the right, and a menu on top (see Figure 7). These tools are the interface to filter and select the data to download.

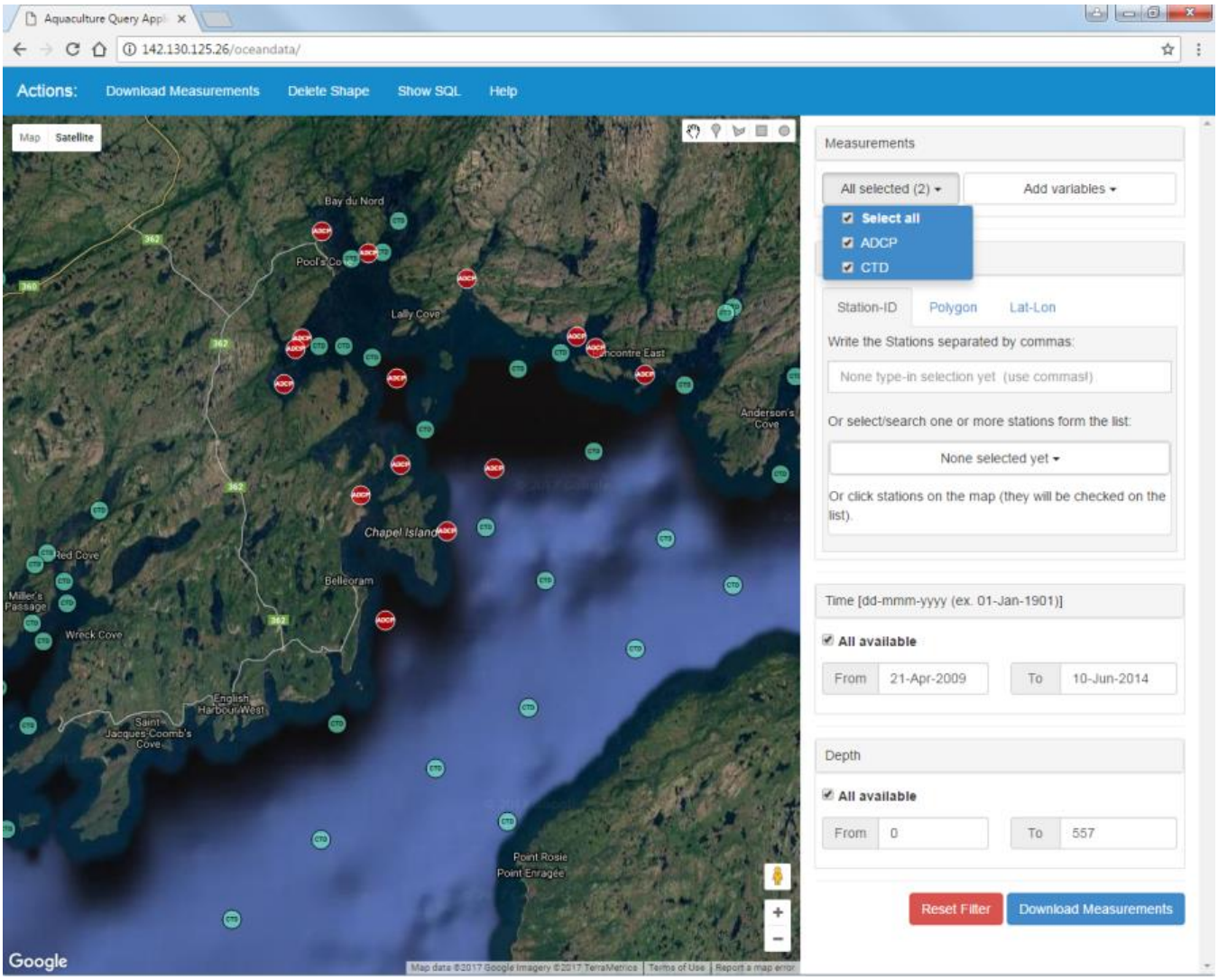


Figure 7: Oceandata Web Application

On the right side of OWA there are four panes that are used to select and filter data. The first one is *Measurements*. In the Measurements pane the user can select instruments (as shown in Figure 7) and measurement variables (Figure 8). The selected instruments and variables will be included in the downloaded file.

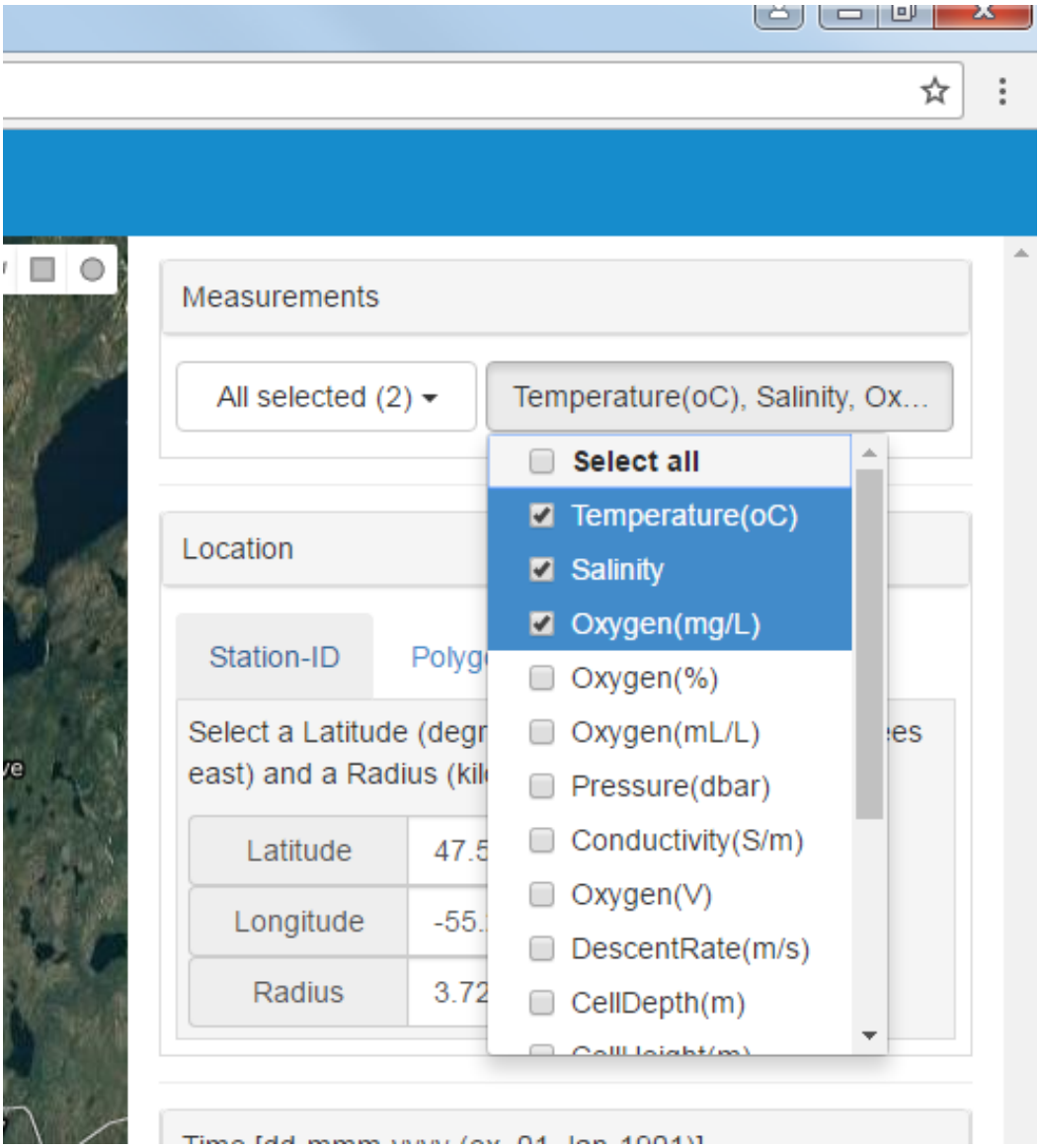


Figure 8: Selecting variables

The next pane is called “Location”. This pane has three tabs where the user can specify the location of the data to download in three different ways to select station to download data from. First, if Station IDs or their location on the map is known, they can be selected and will appear with a different color on the map (Figure 9).

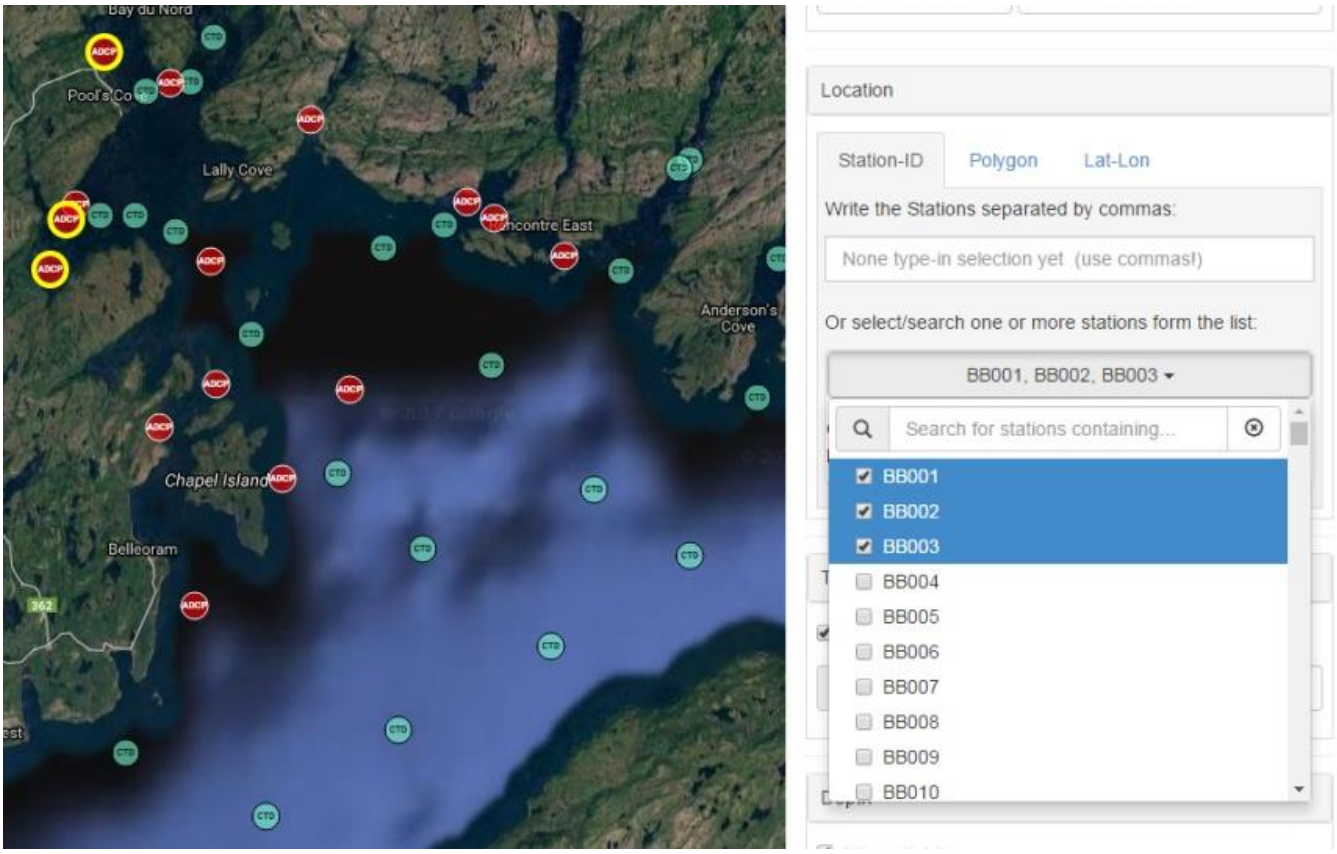


Figure 9: Station Ids selected.

Second, a set of Stations can be selected by drawing the corners of a polygon (. All stations inside the polygon will be included in the downloaded file.

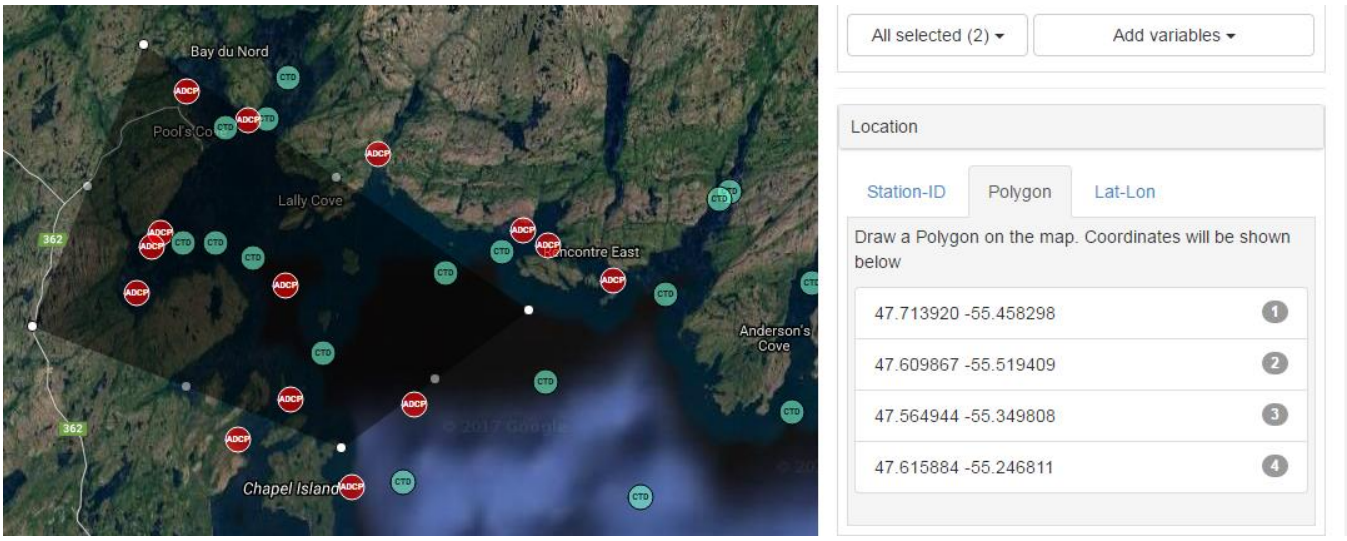


Figure 10: Using a polygon to select stations.

Third, by picking a point on the map and drawing a circle centered in it, all the station inside the circle will be included on the downloaded file (Figure 11).

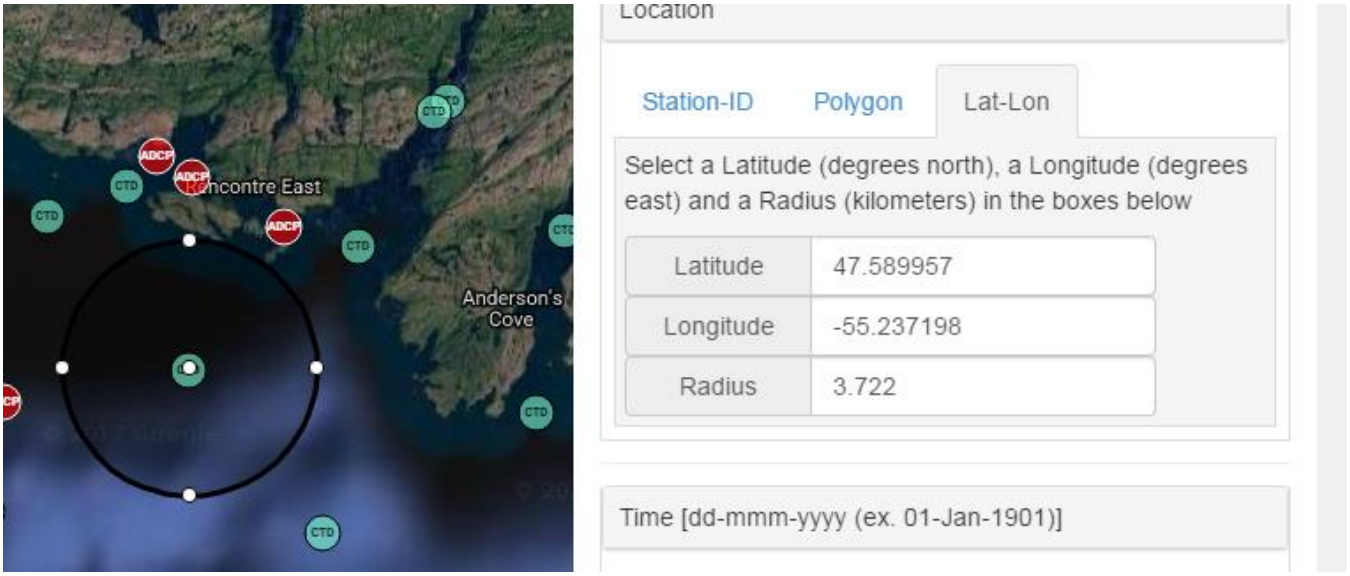


Figure 11: Using a circle to select stations

The last two panes on the right hand of OWA are used to filter time and depth. Time is represented uising the dd-mmm-yyyy (ex. 01-Jan-1901) requires filling the time from and to. Depth is the distance from the sea-surface (i.e. positive downward).

Known issues and room for improvements

- Attempt was made to install oceandata using MySQL workbench on an MS Windows computer (part of MySQL Installer Community suite) but it failed due to “error calling Python module function SQLIDEUtils.runSQLScript” (error during “Run SQL Script”). It is believed to be a MySQL Workbench specific issue which we did not know how to solve.
- OUA error message’s “Instrument Type cannot be blank” should be changed to “Input File cannot be blank” when the Input File box (File tab) is left blank before hitting the “Update Batch” button.
- OUA “Autodetect Instrument Type” box/option should be removed.
- OUA “Update Batch” button should be moved to the source data file tab for more consistency.
- OWA’s Google Earth yellow highlight/border of selected station does not work anymore (since sometime in 2019). It is thought to be due to Google Earth API update or something of the like.
- OWA’s consistency between graphical selection (Google Earth) and right panel with respect to station and/or polygons. Data visually selected should be the ones the user gets data from. As of now, the selected tab in right panel (e.g., “Station-ID”, “Polygon” or “Lat-Lon”) determine the choice. i.e. if “Polygon” is selected then only data from the polygon is extracted even if the user has also selected a station outside that polygon. One would need to be able to select anything, i.e., station&polygon&lat-lon, and be able to download accordingly.

- OWA metadata displayed in GE should be chronological (presently seems to be in order of upload in database)
- If you select “All available” in OWA for time be aware that it actually stops at T=00:00:00 of that last day data are available; thus generally missing some data (up to a day). To be sure not to miss anything, user should extend the search by 1 day (of that indicated).
- Deployment and recovery dates seem not correct for the drifters. This seems to be an issue with indexing. Index for extracting meta-data should be made on the “Station-ID” only so that: first Lat&Lon are extracted and min and max time of that file/deployment is being extracted.
- Drifters have, inherently, Latitudes and Longitudes which varies. The database handles this by expanding the ‘metadata’ table automatically in the case of same Instrument and StationID but varying Lat&Lon. Problem is that once a user upload such a file (Drifter) the database continues to expand the metadata table for every file that follows, even for a ‘fixed’ dataset such as coming from a moored ADCP. The tools (MySQL database, OUA and OWA) will still work fine but this behaviour increases the database space uselessly. This issue can be avoided by using specific variables to Drifter positions other than the mandatory (deployment) Latitude and Longitude, e.g. X_position(DecDeg) and Y_position(DecDeg), and have deployment coordinates as Latitude(decdeg) and Longitude(decdeg). Doing this way, however, will systematically download all the data of any given drifter deployed/present in a polygon selected in OWA (as opposed to data located within that polygon only).
- As designed currently, while the database can virtually handle an infinite number of data (i.e. hardware limited) the OWA, however, is very much limited by the number of stations that can be displayed. Over ~10,000 stations, the application crashes and handling more than ~5,000 stations can render the user experience frustrating (slow). This issue is thought to be due to the metadata ‘fly-by’ capability of the OWA, i.e. it displays key metadata by hovering with the mouse and could be addressed by using an ‘on-demand information query’ feature instead.
- Internal (to the SQL database) “MaxDepth(m)” field/variable should not be visible nor made available to the user other than in Google Earth (as “Depth” field in the ‘fly-by’ box). It should be removed from the extracted/downloaded .csv data which extracts all variables available from the SQL database “metadata” table (by default).
- when extracting data via the OWA, one can end up with many NaNs. e.g., when downloading temperature of a given ADCP without specifying depth range (default is all). Would be best if the application filters out the “no-data” raws. i.e. for any given request, when user is requesting any number of variables (e.g. temperature, current speed, etc) the app should only provide a file that contains at least 1 value of a say variable per raw (i.e. per location, time & depth).authorship details should be provided in both the OUA and OWA. In OUA, it could be in the form of a tab “About” and in the OWA it could either be appended in the help (so no additional work on your part) or as another tab/button “About” placed in the top blue ribbon (e.g., right next to “Help”).
- add a mySQL logo (available in GitHub rep., OWA folder) to OWA’s top right headband.

Appendix 1 : OUA configuration File (config.xml)

This configuration file is an XML (Extended Mark-Up Language) file where all the information necessary to process data files to be uploaded is stored. The information includes the destination database, the source and the schemas for each file type. The schemas contain the source fields names and their destination fields names as well as potential functions necessary to process them.

NOTE: other than making sure, and editing if needs be, that the "**<destination>**" and "**<source>**" sections correspond to the system being installed, a regular user should not have to edit this file extensively. i.e. all the other sections (schemas) get automatically updated as user(s) uses and upload files through the OUA.

Sections and parameters

The configuration file is structured in sections, each one identified by one xml element (e.g. parent <oceandata>). Each section should be closed by a closing xml element (e.g. </oceandata>). If the section has no child sections, that is sections included between the section's opening and closing, they can be closed in the opening element.

Parameters are put inside the section opening element. For example *<date type="string">*.

Values are always between double quotes.

Oceandata section <oceandata>

This is the main section where all the configuration parameters and sub-sections are. This section has no parameters.

Destination section <destination>

This section contains the information to connect to the destination database. It contains three subsections:

```
<destination>
      <odbcparams> Driver…
      <mysqlConn
      <batchsize>2500</batchsize>
      <filedoubleproc>0</filedoubleproc>
</destination>
```

Odbcparams gives the parameter to connect to the database. The odbc driver name needs to be the exact name. Go to the ODBC driver manager on the MS windows control panel (add or remove programs) to get the exact name. Default provided is "**MySQL ODBC 5.3 Unicode Driver**".

Batchsize indicates the number of rows to be sent to the destination in each batch. The larger the faster. 2500 is a good number to deal with ADCP data.

Filedoubleproc controls whether to allow process each file more than once. Use one (1) only for testing purposes. Otherwise use 0.

Source section <source>

It defines the parameters for input files to be processed. It contains only one subsection:

```
<source>
      <odbcparams>Driver…</odbcparams>
      <oledbparams> Provider=… </oledbparams>
</source>
```

Odbcparams indicates the parameters to connect to the database where the data come from.

Olebdparams are Microsoft.ACE.OLEDB parameters; indicate version you have on your system

Schemas section <schemas>

This section contains all the information for each file structure containing measurements which is described in separate subsections.

```
<schemas>
<schema name="Default">
      <InstrumentTypes>
            <InstrumentType name="ADCP" />
            <InstrumentType name="CTD" />
      </InstrumentTypes>
      <metadata table="metadata">
            <field name="Station" type="string" />
            <field name="Latitude(DecDeg)"></field>
            <field name="Longitude(DecDeg)" />
            <field name="Instrument" type="string" />
      </metadata>
      <fields>
            <field name="idEvent" idEventFunction="'getEventID"  />
            <field name="Time_ISO8601" destfunction="str_to_date"
      destparameterstr="%Y-%m-%dT%H:%i:%s" type="datetime"
      function="str_to_date" destination="Time(ISO8601)" />
              <field name="Depth(m)" />
      </fields>
      <afterProcedure></afterProcedure>
</schema>
</schemas>
```

InstrumentTypes subsection list all the instrument type available in the database. It increases as new data type are being uploaded via the OUA. By default (as provided in our GitHub repository), it is blank: <InstrumentTypes></InstrumentTypes>

metadata subsection provides the (mandatory) fields going to the database “metadata” table. Each of the field’s subsection (<field>) may contain the following parameters:

name is the name of the field in the source file.

type sets the field type. E.g. “string” means that the data is handled as text.

value (optional) makes the value of the filed constant rather than taking it from the source file. All the rows inserted in the destination database would have the same value.

FieldName (optional) is the field name in the destination database. If not present, it uses the subsection name. If several field subsections has the same destination they are concatenated.

function (optional) is a function used in the source. It has to be a function supported by the odbc driver

parameter (optional) is the parameter to send to the function in the destination database.

parameterstr (optional) is the string or text parameter to send to the function after the field value in the destination database.

datatable (optional) is the destination table if it is different from “metadata”

afterProcedure (optional) indicates what procedure to run after loading the data.

destination (optional; used in mapping) is the field name in the database. If not present, it uses the subsection “name” (see above). If several fields have the same destination they are concatenated.

fields subsection provides the (mandatory) fields going to the database “measurement” table. Each of the fields’ subsection may contain the following parameters:

name is the name of the field on the source file.

type sets the field type. E.g. “string” means that the data is handled as text; “number” as a floating point number and “datetime” is a date

idEventFunction invokes a function in the server to bring the idEvent from the Event table so that data rows are linked with their metadata.

destination (optional; used in mapping) is the field name in the database. If not present, it uses the subsection “name” (see above). If several fields have the same destination they are concatenated.

sufix (optional) adds a suffix to separate fields if fields are to be concatenated (into one destination field)

value (optional) makes the value of the filed constant rather than taking it from the source file. All the rows inserted in the destination database would have the same value.

destFunction is a function used on the source file to process the value, such as when a unit conversion is needed. It has to be a function supported by odbc driver. It is used by default for the date/time field (str_to_date function).

destparameter (optional) is a parameter to send to the function. It will be added after the field value in the destination database.

destparameterstr is a string or text parameter to send to the function. It will be added after the field value in the destination database. It is used by default for the date/time field (%Y-%m-%dT%H:%i:%s parameter)

parameter (optional) is a parameter to send to the function after the field “value” (see above).

parameterstr (optional) is a string or text parameter to send to the function after the field “value” (see above).

afterProcedure subsection provides any mysql function a user would like to run during an upload. These custom function are entered in the “Procedure to run in the database (optional)” box of OUA’s VarMap tab. It is empty by default.

Appendix 2: Backup and restore

Backup

NOTE: never run Oceamdata+ (i.e. OUA) while backing up.

Run one command for each table from the shell:

```
mysqldump -u root -p oceandata metadata >/srv/data/mysql-backup/metadata.sql
```

where "oceandata" is the database and "metadata" is the table to backup; you can also specify the full path where you want to store the backup files (/srv/....).

```
mysqldump -u root -p oceandata measurement >/srv/data/mysql-backup/measurement.sql
```

Restore

Run one command for each table from the shell:

```
mysql -u root -p oceandata  </srv/data/mysql-backup/metadata.sql
```

where "oceandata" is the database where you want to upload the data to. This database can be changed to any other database available within the system (e.g. oceandata_test).

```
mysql -u root -p oceandata  </srv/data/mysql-backup/measurement.sql
```

NOTES:

It is very important to specify the database, otherwise the upload will be done onto the default database when one logs into mysql.

The order of the restore command is important, i.e. 1. metadata.sql, 2. measurement.sql).