Journal of Software Engineering and Applications, 2021, 14, 563-574
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

MeteoRead: Client Software for Inserting Observed Atmospheric Data into MySQL™ Database and Downloading them into Different File Format

Beáta Szabó-Takács[1], Tamás Takács[2], Aleš Roček[3]

[1]Department of Atmospheric Matter Fluxes and Long-Range Transport, Global Change Research Institute, Brno, Czech Republic
[2]Faculty of Engineering and Information Technology, University of Pécs, Pécs, Hungary
[3]Institute of Computer Science, Masaryk University, Brno, Czech Republic
Email: szabo.b@czechglobe.cz, takacst7200@gmail.com, rocek@ics.muni.cz

How to cite this paper: Szabó-Takács, B., Takács, T. and Roček, A. (2021) MeteoRead: Client Software for Inserting Observed Atmospheric Data into MySQL™ Database and Downloading them into Different File Format. *Journal of Software Engineering and Applications*, **14**, 563-574.
https://doi.org/10.4236/jsea.2021.1410033

Received: September 16, 2021
Accepted: October 15, 2021
Published: October 18, 2021

Abstract

The investigation of the tendency of climate change and its effects on ecology, economy and sociology is essential for long term policy making. The long-term measurement of the physical and chemical properties of the atmosphere with state-of-the-arts instruments provides high-quality data for these studies. The evaluated data are stored in really special file structures and formats that cannot be inserted in one common database. Moreover, the observed data usually available in ASCII format and the users sometimes need to convert them in other format. The file conversion is usually time consuming procedure and can contribute to the uncertainties. MeteoRead is a client database software that imports the observed atmospheric data e.g. wind direction, wind speed, aerosol particle concentration etc. and makes them available in different file formats, which are most commonly used in climate research. This Java™ based program applies the Structured Query Language (SQL) functions such as table creation on a database server, data or figures insertion into the table and data selection via Graphical User Interface. The selected data can be stored in NetCDF, HDF5, DataBase or TXT file formats and the figures can be available in PNG, JPG, JPNG, PDF or GIF file formats. The program was tested on Linux and Windows platforms with different Java™ Development Kit. The MeteoRead is planned to be developed to visualizing the annual, seasonal, monthly, daily or hourly average value of the selected data and to use the functionality of the SQL database to calculate various mathematical and statistical correlations.

Keywords

Java™ Language, MySQL™, Database, Atmospheric Data, Global Change

1. Introduction

Investigation of the impact of global climate change on the atmosphere requires state-of-art instruments for acquiring high-quality data. The activities of our department are based on provision and exploitation of measurements conducted at the Atmospheric Station Křešín u Pacova, which is part of the National Atmospheric Observatory Košetice (NAOK) and member of some international programs like Integrated Carbon Observation System (ICOS), Global Mercury Observation System (GMOS), Aerosol, Clouds, and Trace Gases Research Infrastructure Network (ACTRIS), European Monitoring and Evaluation Program (EMEP) and Air Quality Information System (ISCO) in the middle of the Czech Republic (49˚35'N, 15˚05'E). A wide range of instruments are used for continuous and hi-precision monitoring of greenhouse gases concentrations, aerosols physical and chemical properties, ozone and mercury concentration long with continuous measurement of meteorological parameters, cloud base height and atmospheric boundary layer structure at several height levels of the high tower (250 m) [1]. The observed atmospheric data were processed and evaluated according to the standard operating procedure from ICOS [2] [3], GMOS [4] and EBAS-EMEP [5]. In climate research the observed atmospheric data are essential. Therefore, the data should be available for climate-related research departments and institutes. Moreover the measured data are applied for toxicological studies e.g. we contribute to the Minamata Convention with the measured atmospheric mercury.

The observed data are available in the above-mentioned systems but to collect the different data the user need connect to multiple databases. Furthermore, the observed data usually available in ASCII format and the users sometimes need to convert them in other format e.g. HDF5, NetCDF or DataBase. The file conversion is usually time consuming procedure and can contribute to the uncertainties. The evaluated data are stored in really special file structures and formats that cannot be inserted in one common database. Although, the Open-source Project for Network Data Access Protocol (OpenDAP) is a freely available software for scientific data networking but the user needs a data analysis tool which has adapted to enabled DAP-based data input [6]. The benefit of OpenDAP is the remote access of the structured scientific data but the user has limitation regarding the usage of the data. According to the authors' best knowledge there is no available any database manager on the market that can import data from different special data files into one common database and that is able also export the selected data from the database to different file formats.

The motivation of creating a user-friendly client database software is to share

our observed and qualified atmospheric data with the benefits that allows easily create a database table, insert data from different special file structure and that the data make available in a different file format without the necessity of user experience in database handling.

The MeteoRead is a user-friendly Graphical User Interface (GUI) program written in Java™ language for database management (Figure 1). The exported data can be stored in four different file formats: NetCDF, HDF5, DataBase and TXT. Furthermore, the users can also insert figures (e.g..png,.jpg,.pdf) into the database, and the figures can be downloaded in five different figure formats: PNG, JPG, JPNG, PDF and GIF.

2. Data and Tools

MeteoRead was developed in Java™ language. One of the most significant benefits of Java™ language is its platform independence as the development and execution are available on almost all devices. Java™ is an object-oriented language where the created objects remain available throughout the runtime thanks to its

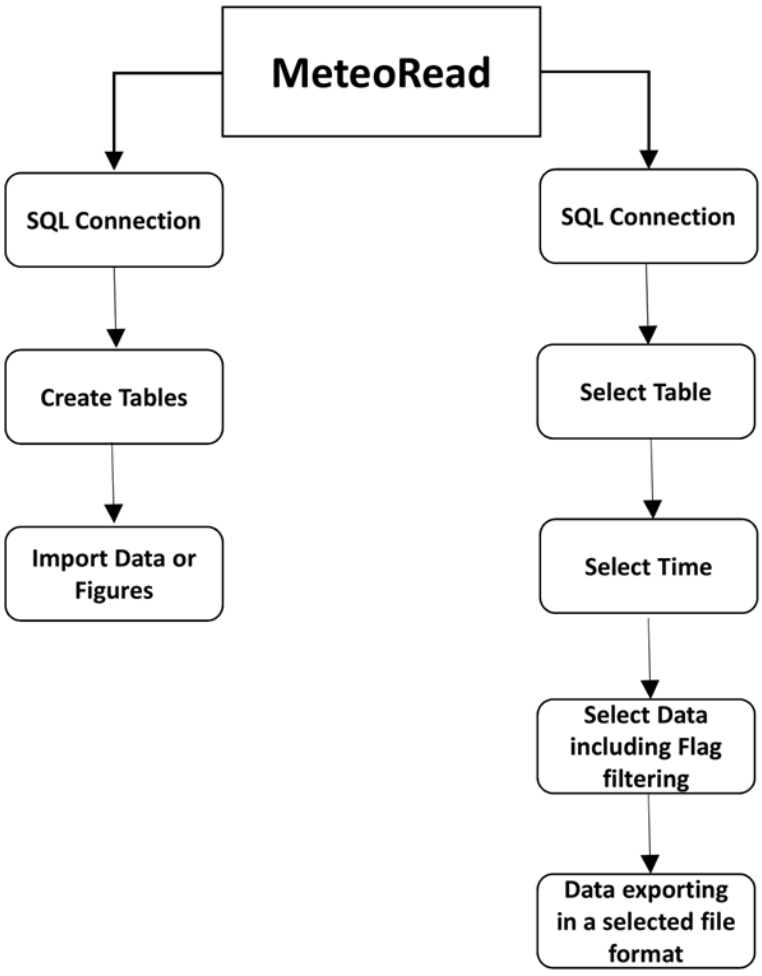

Figure 1. Schematic diagram of the MeteoRead software.

memory handling technique [7]. Moreover, multithreading is also one of the language's best functions, which is used to accelerate the program's runtime by allocating the biggest tasks to separate threads. Several Java™ classes for software development can be found in Java™ Development Kit (JDK). Java™ Database Connectivity (JDBC) is an Application Programming Interface (API) for Java™ language, which is used for connecting to a relational database and acquiring data. JDBC API's classes and interfaces are an excerpt of the general principles and operations used to access any database [8]. The Structured Query Language (SQL) was developed by IBM in 1970 and was standardized in 1986. SQL is a language for implementing the operation of relational algebra based on a relational data model. The MySQL™ is a SQL-based relational database management server. It was acquired by Sun Microsystems in 2008. MySQL™ is an open-source system and one of the most common on the market. The Meteo-Read was developed with the qualified output of the following instruments: 2D anemometer Gill WindObserver, Aethalometer AE33, Barometer Young, Vaisala Ceilometer CL51, Humidity and Temperature sensor Vaisala HMT, Integrating Nephelometer TSI 3563, Picarro G2301 $CO_2/CH_4/H_2O$ Analyser, Tekran 2537B Ambient Mercury Monitor and Thermo 49i UV Photometric O3 Analyser.

3. Structure of the Software

3.1. Login Part of the Software

The Java™ Database Connection (JDBC) driver supports the connection to MySQL™ database server. The privileges, *i.e.* Administrative roles have to be set for the users on the MySQL™ server. The users, who have privileges (Administrative roles) as Create Tables and Insert, can create SQL Table(s) and import observed data to the created tables.

The Select role gives permission for data export from the database for every user. The purpose of this setting privileges process is to allow the table creation and data insertion only for the users who have experience in the measurement and data evaluation process, the file structure and the data attributes.

Each process (table writing, data or figure inserting and export) is made via a graphical interface. The program starts with login part which gives the input parameters for the JDBC connection to the MySQL™ server (**Figure 2**).

The advantage of this Login part is that the client is able to connect to multiple MySQL™ databases.

3.2. Creating SQL Tables and Importing Data

The observed and evaluated data can be stored in different file formats (e.g. NASA Ames;.dat;.txt;.hist;.csv) what the software is able to read for SQL table creation and data insertion. These input files usually contain header which are the name of the observed data. Owing to the fact that the observed data are timeseries each header include date and time information in different formats (e.g. "Date", "DATE", "stime"). The program can find the data header based on

the date and time information and distinguish it from the observed data.

SQL tables can be created with adding the name, the altitude and the location of the instrument (Figure 3). One string is created from this information which is the SQL table name. The purpose of this table name creation process is to distinguish the data measured with same instrument in different stations and or in different height levels. In this part of the program the header of the input files

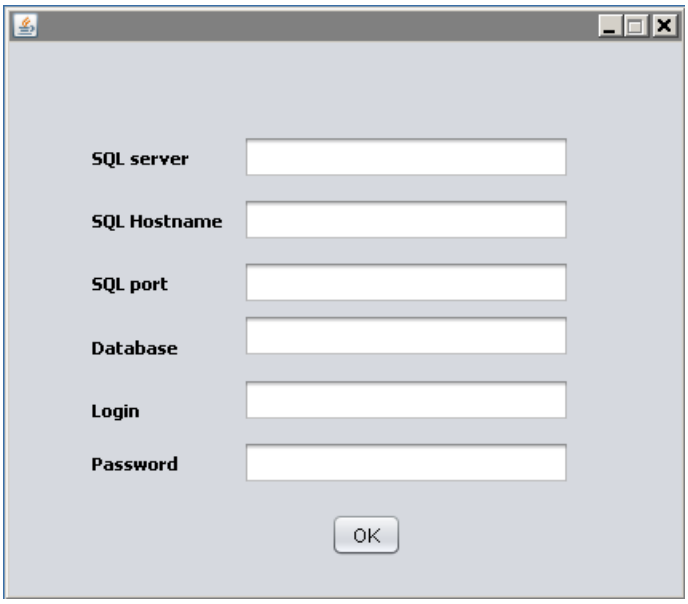

Figure 2. Login GUI part of the program.

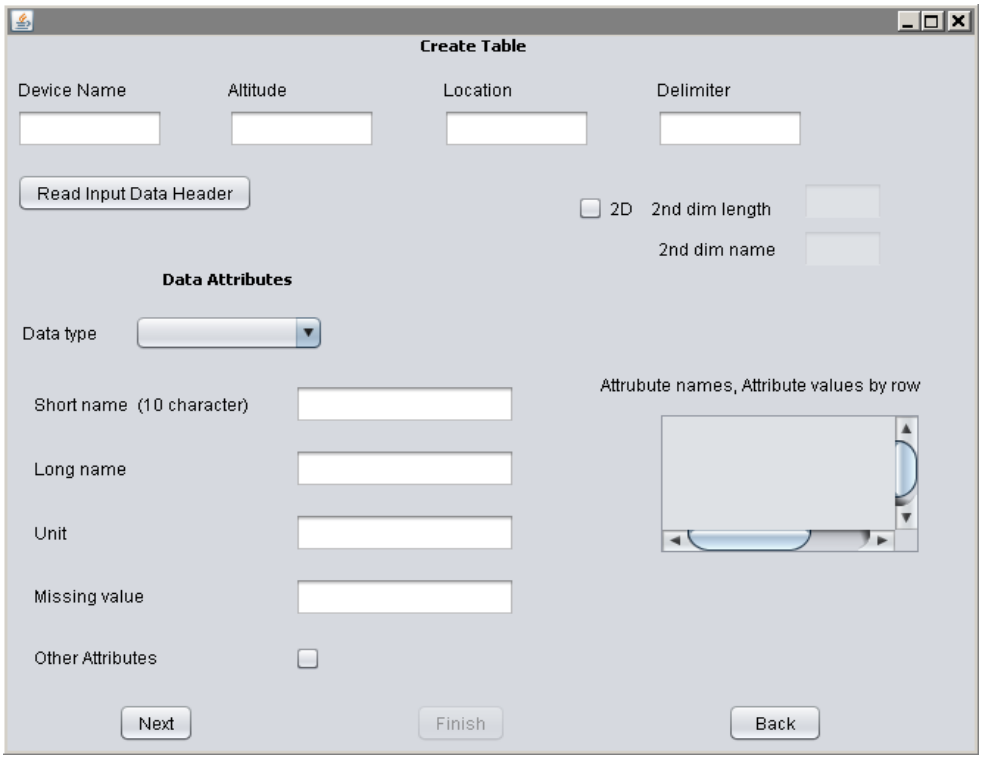

Figure 3. Creating Table with Header GUI part of the program.

are read and set the SQL column names based on it. The recognition of the SQL column type is not automatic. The user has to set it according to the data type, manually. The following data types can be chosen: Date, String or Character, 20-bit Hex ASCII, Integer and Double. The used date formats are various therefore one extra column "Time_ID" is added to the table where the date and time variables are set in a uniform date format: YYYY-mm-dd hh:MM:ss. The 20-bit Hex ASCII data, that is the ceilometer profile data, are stored as BLOB in the SQL table. Furthermore, the attributes of the data such as "short name", "unit", "missing value" or any other one can be set. These attributes need for the NetCDF and HDF5 format writing. Due to the fact that NetCDF [9] and HDF5 [10] writers have unique Java functions the NetCDF and the HDF5 writer's java codes were stored in SQL tables in Strings (LONGTEXT) for NetCDF and HDF5 file writing, respectively (**Figure 4**). This method was chosen to make the software more flexible and independent on the chosen data. Thanks to this method we did not have to write one NetCDF and one HDF5 file writer Java class for each instrument, respectively.

Four SQL tables are created with this part of the software. One SQL table contains the data, two SQL tables contain the NetCDF and HDF5 Java functions in strings, respectively based on the read name and the given attribute information according to the data type (**Figure 4**). Moreover, one additional SQL table stores the short names of the data for DataBase file writer [11] because it has a limitation to write a header which is longer than 10 characters. In some case, however, the variables are stored without header e.g. Vaisala CL51 ceilometer messages [12]. In this case, the SQL table is created with "Create SQL Table without Header" part of the program without reading the header of the input file. The user has to know the order of the variables and give the number of the variables (number of table columns). The structure of this file is same as the "Create SQL

AT010
Variable AT010; AT010 = ncfile.addVariable(null, "AT010", DataType.DOUBLE, "time");
AT010.addAttribute(new Attribute("long_name", "Temperature")); AT010.addAttribute(new Attribute("unit","Celsius degree")); AT010.addAttribute(new Attribute("_FillValue",-999.9));
ArrayDouble.D1 AT010Data = new ArrayDouble.D1(countLinesResult);
AT010Data.set(ima.set(timeIdx),Double.parseDouble(Value("AT010").get(timeIdx)));
ncfile.write(AT010,AT010Data);

Figure 4. NetCDF java code in SQL table of temperature at 10 m. The countLinesResult at the third row indicates the number of the selected row in the SQL table where the temperature data are stored which is defined in the time selecting part of the program. These strings are stored in the "Create SQL Table with Header" part where the AT010 is the name of the data from the header and the "Temperature", "Celsius degree" and the "–999.9" were given by the user via GUI as "longname", "unit" and "missing value" attributes. The long name and the unit are set originally as string but the missing or fill value is casted according to the data type in the file writer.

Table with Header" with the exception that the given "short name" is the name of the column.

In the case of figures only the instrument name, altitude and location have to be given for SQL Table creating which includes three columns: the Time_ID as varchar, the image as BLOB and the ID as auto-increment integer.

The columns are separated based on the predefined delimiter which needs to be specified by the user based on the input data source format (Figure 5).

The data are read from the input file line by line and the data are stored in String array by splitting the current line with the chosen delimiters. In the "OK" button Action Performed the chosen table is checked if it is empty or not. If the table is not empty the existing "TIME_ID" values stored to a String array for checking if the read data are still imported. One method was developed to compare the inserted time and the time data in the table for preventing duplication. If the time data exists in the table, a message warns the user.

The SQL inserting statement is set according to the name and number of SQL table columns and the read data are posted and casted according to the type of the column with a for loop. In the case of figures importing the user has to add the figure's time in the time format as yyyy.MM.dd HH:mm:ss e.g. 2015.03.12 00:00:00.

3.3. Data Selecting and Export

The data can be exported in NetCDF, HDF5, DBF or TXT format and the figures can be stored in PNG, JPG, JPNG, PDF or GIF file from the SQL database (Figure 6).

The tables are listed in the same way as in the data importing Java Class. After the SQL table is selected the begin and end time of the time series have to be specified (Figure 7).

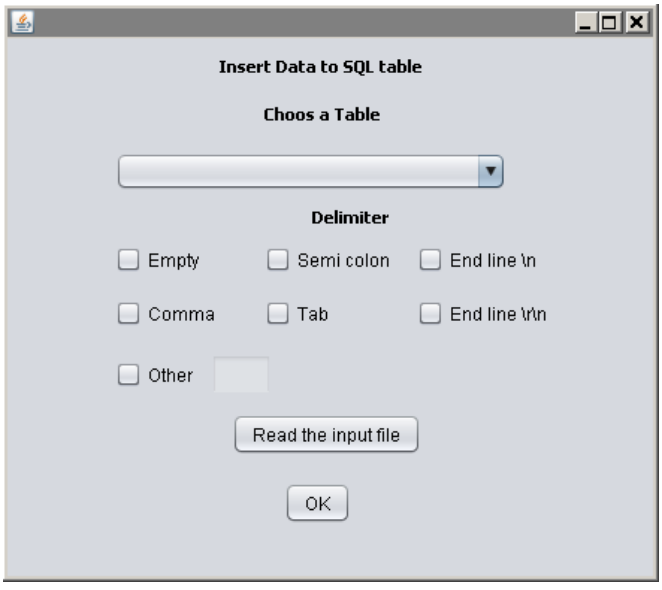

Figure 5. Data Importing GUI part of the program.

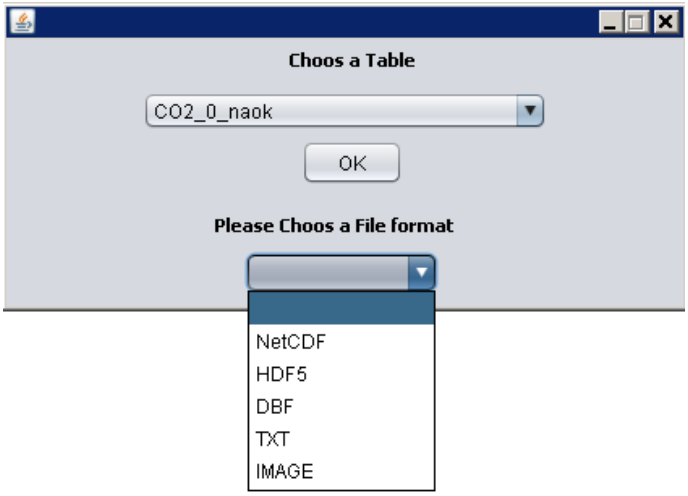

Figure 6. Data Exporting GUI part of the program.

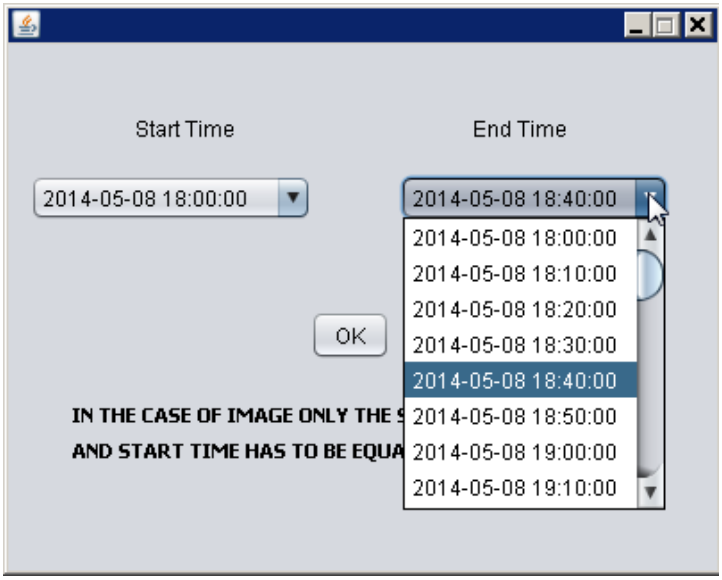

Figure 7. Time Selecting GUI part of the program.

The selected "Start Time" and "End Time" are set as public strings which are used for the data exporting. The program gets the row numbers of the selected times. After the time filtering the data selection is available.

In SelectData Class the names of the columns in the chosen SQL table are selected as represent the name of the variables (**Figure 8**). The selected data names are stored into a string array by a for loop if the "isSelected" condition is true. The items of this string array are added into an empty public string where the items are delimited with comma. This string is called for the SQL selecting statement in the file writers. Some data are labeled with flags. In that case the user can filter the data with multiple flag values. The count of the selected rows are determined with "SELECT COUNT (*)" SQL statement according to the filtering and the result is stored in the "countLinesResult" public static integer which is used for NetCDF file writing (**Figure 4**).

After these filtering the program set the SQL select statement according to the selected information. The functions for NetCDF or HDF5 file writing and short names according to the selected data are selected from the SQL tables and stored to public static strings which are called in the Java Classes where the file writers are built. For instance, public static string name "Variable" is contains the NetCDF Java functions for variable declaration what are selected from the first row (*i.e.* ID = 1) of the SQL table for NetCDF writing (Figure 4).

The NetCDF and HDF5 file writers are abstract java classes where the java functions are written in string builder and compiled with InMemoryJavaCompiler [13]. The observed data are accessed from the SQL table by the "Value (column name)" where the input parameter is the SQL column *i.e.* the variable name and the return value is the value of the data stored in String ArrayList which is casted regarding the type of the data. For instance, the "Value (AT010)" at the fourth row in Figure 4 is a String ArrayList which accesses the values of temperature by the method is shown in (Figure 9.)

The two-way attenuated backscatter profile of CL51 ceilometer is coded with

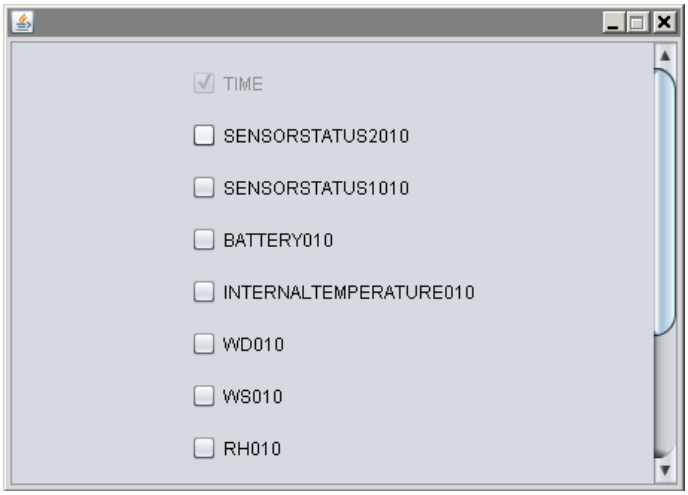

Figure 8. Data Selecting GUI part of the program.

```
sourceCode.append("public ArrayList<String> Value(String colName)throws
Exception{\n" +
      "ArrayList<String> array = new ArrayList<String>();\n" +
          "try{\n" +
             "result.first();\n" +
             "result.previous();\n" +
                   "while(result.next()){\n" +
                         "array.add(result.getString(colName));\n" +
                      "}\n" +
                "}catch(Exception e){\n" +
                "e.printStackTrace();}\n" +
       "return array;}\n");
```

Figure 9. Data receiving method of the NetCDF and the HDF5 writer where the "result" is the ResultSet object which points to the current row of data on the data SQL table.

20-bit Hex ASCII character set, which length is equal to 5 times the length of the profile. This character set is decoded to digit in NetCDF and in HDF5 writer with the method in Figure 10.

The structure of the TXT and the DataBase file writer is more simple and flexible than the above-mentioned writers therefore they were built in a Java Class, respectively. In the case of TXT writer the header was defined based on the string what contained the name of the variables delimited with tab (\t). After the header determination the selected data from the SQL table were stored in a string with tab delimiter. In the DataBase writer database fields were declared and set according to the data properties (e.g. character, numeric) [11] and the selected data were stored into an object which was added to the writer. The TXT writer applies FileWriter Java Class while the DataBase writer uses FileOutputStream Java Class.

The imported figures e.g. BLView software of Vaisala Ceilometer produced snapshots about the boundary layer status, can be exported into PDF applying with Apache PDFBox® library and into JPG, JPNG, GIF or PNG format using with FileOutputStream Java Class in the image writer.

When the file format is chosen the filechooser is launched where the user set the path where the file is saved and the instance of the file writer is implemented. Because processing of the large files, especially the NetCDF and the HDF5 building is time consuming the file writers contain two threads. One thread launches a panel that includes a progress bar to show the downloading process in percent, the other thread select the data from SQL database and build the file. The progress bar value is the data receiving speed from the database. These threads are implemented simultaneously. When the SQL connection and the file are closed

```
public static ArrayList<Double> backscatter(String s){
    int digit1 = 0;
    double digit =0.0;
    String ss;
    double digit2;
    ArrayList<Double> bcdigit = new ArrayList<Double>();

        for(int i=0; i < s.length()-5; i=i+5){
          ss = s.substring(i,i+5);
          digit1 = Integer.parseInt(ss,16);     /cast the hexadecimal to inte-
ger
                    digit = digit1 * 1.0;       /convert integer to decimal

                if(digit > Math.pow(2.0, 19.0)){
                    digit = -(Math.pow(2.0, 20.0) - digit);
                    }
                digit2 = digit* 0.00001;         /convert the unit to km-1str-1
                bcdigit.add(digit2);
          }
          return bcdigit;
     }
```

Figure 10. Method for converting the 20-bit Hex ASCII character set to decimal.

a pop-up window informs the user, if the file export was successful or not.

3.4. Conclusions and Future Plan

MeteoRead is a client database software for importing observed atmospheric data and figures, and for also filtering and exporting the data in the most commonly applied file formats such as NetCDF, HDF5, DataBase, and TXT and the figures in PNG, JPG, JPNG, PDF and GIF formats. The software was developed in Java™ language by applying the JDBC API for MySQL™ server. The benefit of the program is that the data handling is ensured by GUI, and there is no need for any other client software. The data can be filtered according to their date, variables and flags.

The exceptions were handled with try-catch blocks in each part of the code, and during the development of the code a basic method of debugging was used to check each result. To find the dependence on the environment, MeteoRead was tested on the following platforms:

Debian GNU/Linux 10 (buster) with openJDK 11.0.9.1,

Linux 3.10.0-1127.19.1.e17.x86_64 with openjdk-1.8.0.262.b10-0.el7_8.x86_64,

Windows 10 with JDK 1.8.0 271-b09 and

Windows 7 with JDK 1.8.0_151.

MeteoRead is planned to be developed further. In the present state of the software is applicable for data transfer. We also realized the importance of the data analysis and the software is planned to be developed with data visualization and analytical part. This will contain the possibility of displaying of the yearly, monthly, seasonal, daily and hourly average value of the selected variables in a chart applying with JFreeChart API and saving the chart and the averaged data in selected formats. During the development we also plan to use the functionality of the SQL database to calculate different mathematical and statistical correlations. This should open new possibilities for different groups to use the measured data more for their analysis. The MeteoRead is currently able to connect with MySQL™ servers, but it will be developed to be able to connect with PostgreSQL server as well.

Acknowledgements

This work is based on the use of Large Research Infrastructure CzeCOS supported by the Ministry of Education, Youth and Sports of CR within the CzeCOS program, grant number LM2018123. The authors would like to thank Saliou Mbengue, Gabriela Vítková, Kateřina Komínková for supporting qualified observed data and also thank to Petr Vrána for technical supporting for software testing. Furthermore the authors would like to thank the reviewers' comments.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Dvorská, A., Sedlák, P., Schwarz, J., Fusek, M., Hanuš, V., Vodička, P. and Trusina, J. (2015) Atmospheric Station Křešín u Pacova, Czech Republic—A Central European Research Infrastructure for Studying Greenhouse Gases, Aerosols and Air Quality. *Advances in Science and Research*, **12**, 79-83. https://doi.org/10.5194/asr-12-79-2015

[2] Hazan, L., Tarniewicz, J., Ramonet, M. Laurent, O. and Abbaris, A. (2016) Automatic Processing of Atmospheric CO_2 and CH_4 Mole Fractions at the ICOS Atmosphere Thematic Centre. *Atmospheric Measurement Techniques*, **9**, 4719-4736. https://doi.org/10.5194/amt-9-4719-2016

[3] Yver-Kwok, C., Philippon, C., Bergamaschi, P., Biermann, T., Calzolari, F., Chen, H., Conil, S., Cristofanelli, P., Delmotte, M., Hatakka, J., Heliasz, M., Hermansen, O., Komínková, K., Kubistin, D., Kumps, N., Laurent, O., Laurila, T., Lehner, I., Levula, J., Lindauer, M., Lopez, M., Mammarella, I., Manca, G., Marklund, P., Metzger, J-M., Mölder, M., Platt, S.M., Ra-monet, M., Rivier, L., Scheeren, B., Sha, M.K., Smith, P., Steinbacher, M., Vítková, G. and Wyss, S. (2021) Evaluation and Optimization of ICOS Atmospheric Station Data as Part of the Labeling Process. *Atmospheric Measurement Techniques*, **14**, 89-116. https://doi.org/10.5194/amt-14-89-2021

[4] D'Amore, F., Bencardino, M., Cinnirella, S., Sprovieri, F. and Pirrone, N. (2015) Data Quality through a Web-Based QA/QC System: Implementation for Atmospheric Mercury Data from the Global Mercury Observation System. *Environmental Science: Processes & Impacts*, **17**, 1482-1491. https://doi.org/10.1039/C5EM00205B

[5] European Monitoring and Evaluation Programe. http://ebassubmit.nilu.no/

[6] Open-Source Project for Network Data Access Protocol. http://www.opendap.org/

[7] Sierra, K. and Bathes, B. (2003) Head First Java. 2nd Edition, O'Reilly Media, Inc., Sebastopol, CA.

[8] Reese, G., Yarger, R. J., King, T. and Williams, H. E. (2002) Managing and Using MySQL. 2nd Edition, O'Reilly and Associates, INC., Sebastopol, CA.

[9] Unidata (2020) NetCDF Java version 5.4.1 [software]. Boulder, CO: UCAR/Unidata Program Center. https://www.unidata.ucar.edu/software/netcdf-java/

[10] Rinn, B. (2018) JHDF5 (HDF5 for Java) 14.12 [software]. Zurich, ETH Zürich, Hauptgebäude, Rämistrasse 101. https://sissource.ethz.ch/sispub/jhdf5

[11] Kumar, A. (2004) JavaDBF ver. 0.4.0 and above [software]. Linuxense Information Systems Pvt. Ltd., Trivandrum. http://priede.bf.lu.lv/ftp/pub/DatuBazes/DBF/javadbf/javadbf-tutorial.html

[12] Vaisala (2010) Vaisala Ceilometer CL51. http://www.vaisala.com/en/products/ceilometers/Pages/CL51.aspx

[13] https://mvnrepository.com/artifact/org.mdkt.compiler/InMemoryJavaCompiler