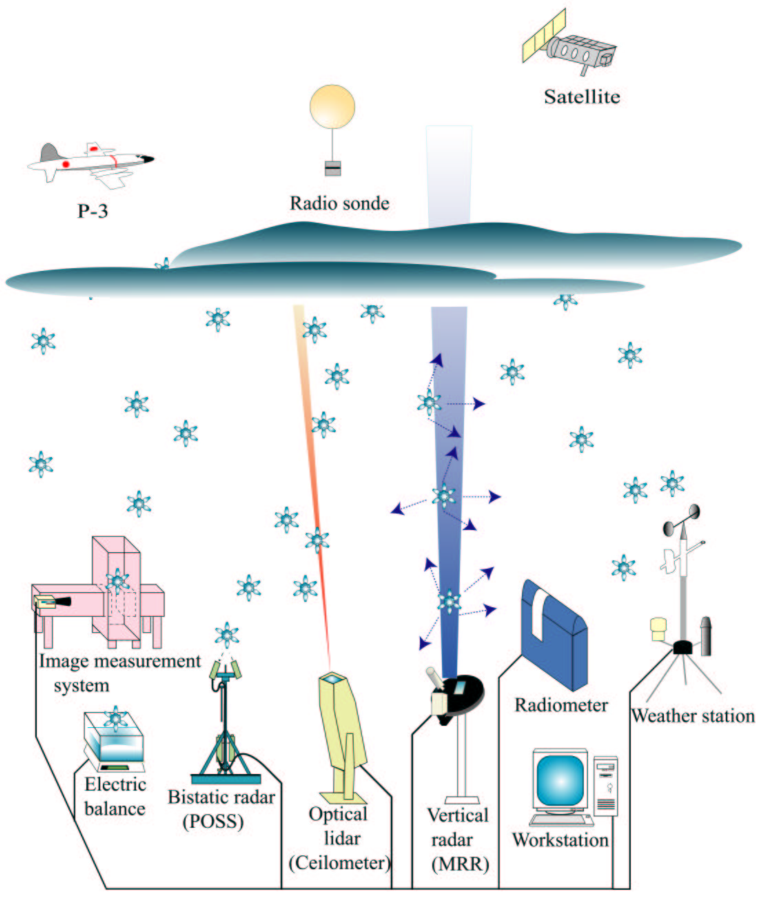
Wakasa Database User's Guide

Arto Teräs ⟨teras@wis.ec.t.kanazawa-u.ac.jp⟩

September 13, 2003



Contents

1 Introduction

This manual is a user's guide for the Wakasa database, which is used for storing weather related observation data, especially snowfall data. This guide is intended for a user who does not need to modify the database contents or take care of other administrative tasks. There is a separate Administrator's Guide which covers such operations. There is also a Developer's Guide which explains the structure and source code of the current tools on a technical level and gives advice on further development, for example adding new instruments and developing analysis programs which access data in the base directly.

The manual is organized as follows. First, the structure of the database is described and methods of accessing the data are explained. The main part of the manual concentrates on using the visalization tools developed for the project. Then, there is a short introduction of using the data in other third party programs. Finally, the data types and descriptions of all columns of the database tables are listed in the appendix.

2 Accessing data in the database

Database systems allow storing large amounts data efficiently. For a user accustomed to standard text files, it may first sound scary that the data in the database cannot be accessed in the same way, using any text editor. However, there are many tools which can be used to access the data, retrieve just the necessary parts of it and even store the result in a text file, if the user wants that.

The data in Wakasa database is measurement values from several instruments, stored in **tables**. Before the database, the same data of the same instruments was stored in text files. In this case, you can consider a table as one huge text file, storing all the data for a long time period, but still allowing very quick and convenient access to any individual value or range of values.

2.1 Wakasa database structure

The Wakasa database contains data from 9 different instruments. The data is organized in tables, including one parameter table and one or more data tables for each instrument. An overview of the structure is shown in figure 1.

Each table contains a number of columns. More detailed figure of the structure, including labels of individual columns, can be found in Appendix A. The data types and descriptions of each column are listed in Appendix B.

Actually, there are two versions of the Wakasa database, **standard version** and **array version**. The structure shown in figure 1 is the standard version. In the array version, tables `ceilo_backscatter`, `mrr_data` and `mrr_raw_data` don't exist. Their contents have been merged to tables `ceilo`, `mrr` and `mrr_raw` as PostgreSQL arrays[1].

[1]The array data type is a PostgreSQL database engine specific extension to standard SQL.

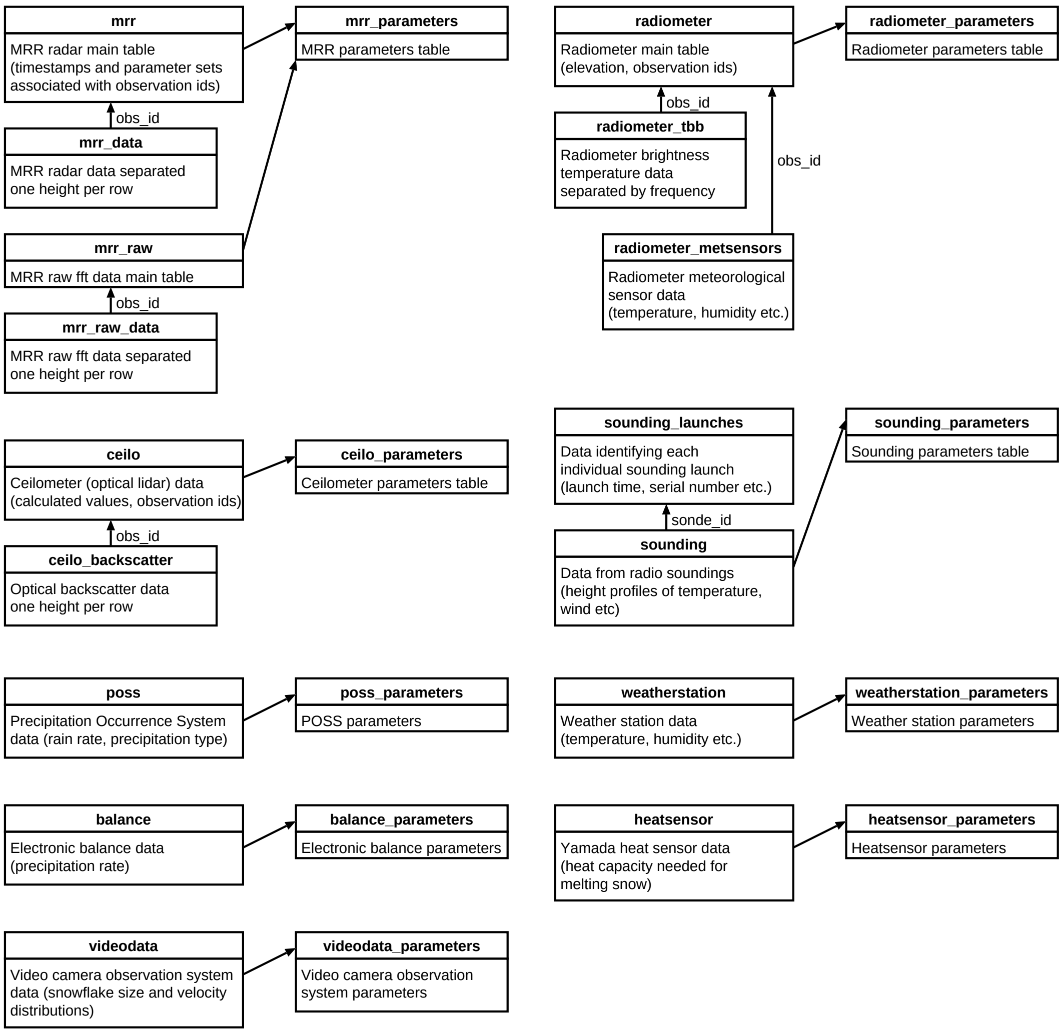

mrr
MRR radar main table (timestamps and parameter sets associated with observation ids)
mrr_parameters
MRR parameters table
obs_id
mrr_data
MRR radar data separated one height per row
mrr_raw
MRR raw fft data main table
obs_id
mrr_raw_data
MRR raw fft data separated one height per row
radiometer
Radiometer main table (elevation, observation ids)
radiometer_parameters
Radiometer parameters table
obs_id
radiometer_tbb
Radiometer brightness temperature data separated by frequency
obs_id
radiometer_metsensors
Radiometer meteorological sensor data (temperature, humidity etc.)
ceilo
Ceilometer (optical lidar) data (calculated values, observation ids)
ceilo_parameters
Ceilometer parameters table
obs_id
ceilo_backscatter
Optical backscatter data one height per row
sounding_launches
Data identifying each individual sounding launch (launch time, serial number etc.)
sounding_parameters
Sounding parameters table
sonde_id
sounding
Data from radio soundings (height profiles of temperature, wind etc)
poss
Precipitation Occurrence System data (rain rate, precipitation type)
poss_parameters
POSS parameters
weatherstation
Weather station data (temperature, humidity etc.)
weatherstation_parameters
Weather station parameters
balance
Electronic balance data (precipitation rate)
balance_parameters
Electronic balance parameters
heatsensor
Yamada heat sensor data (heat capacity needed for melting snow)
heatsensor_parameters
Heatsensor parameters
videodata
Video camera observation system data (snowflake size and velocity distributions)
videodata_parameters
Video camera observation system parameters

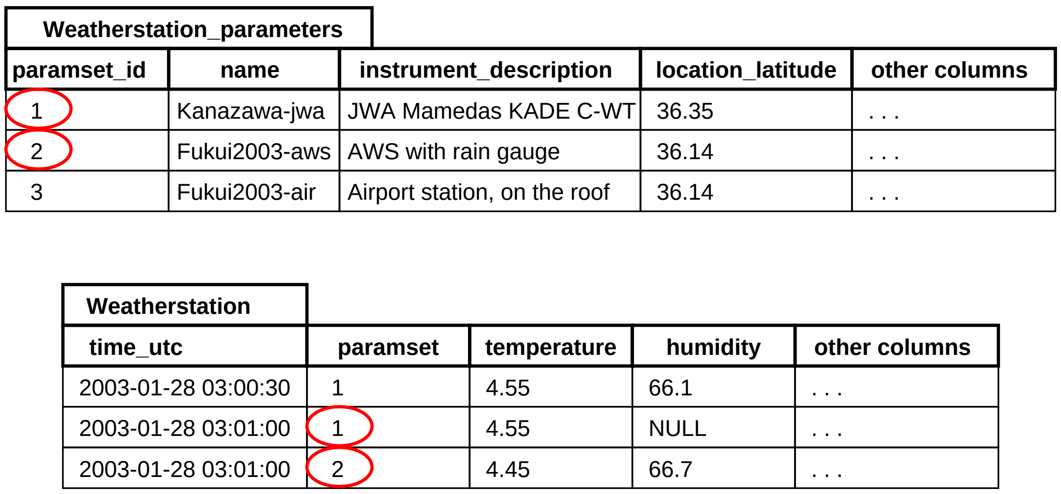
Figure 1: Wakasa database structure

2.2 Data tables and parameter tables

All measurement data in the database is associated with a parameter set. Therefore, there are at least two tables associated with each instrument: one **data table**[2] and one **parameter table**. The actual measurement values are stored in the data tables, while the parameter tables contain values which remain constant during a long period of time. In the data table, one measurement is represented on one row. Figure 2 illustrates how the data is stored.

Note that one table can contain data from several similar instruments. In this example, there are three weather stations, one located in Kanazawa and two in Fukui, represented by parameter set id's 1, 2 and 3, respectively. For each parameter set, there is also a more descriptive name: `Kanazawa-jwa`, `Fukui2003-aws` and `Fukui2003-aws` in our example. These names can be used when visualizing the data using the plotting tools (see chapter 3).

[2]Depending on the instrument, there may be additional data tables which are linked to the main table.



Weatherstation_parameters

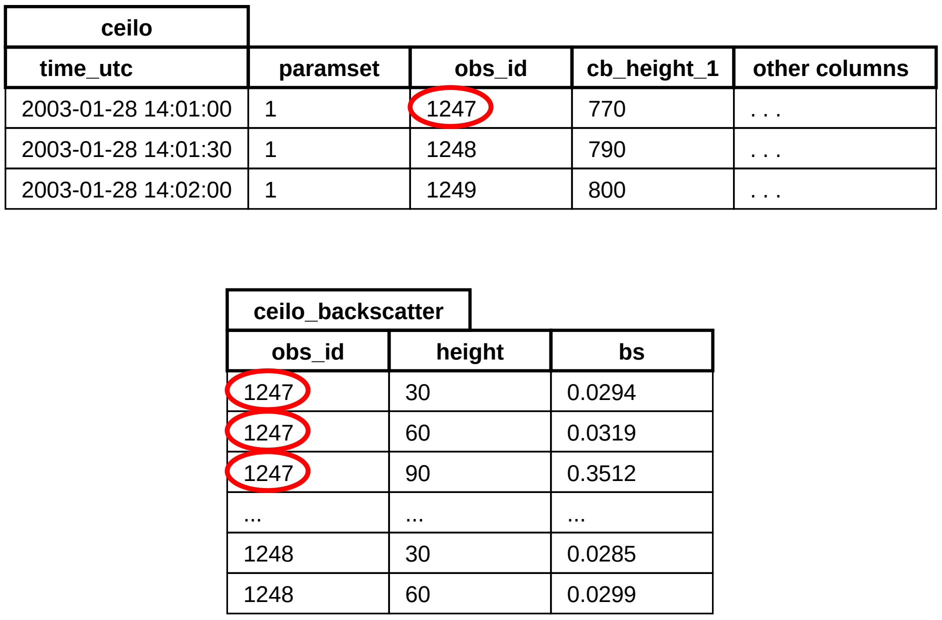
paramset_id	name	instrument_description	location_latitude	other columns
1	Kanazawa-jwa	JWA Mamedas KADE C-WT	36.35	. . .
2	Fukui2003-aws	AWS with rain gauge	36.14	. . .
3	Fukui2003-air	Airport station, on the roof	36.14	. . .

Weatherstation

time_utc	paramset	temperature	humidity	other columns
2003-01-28 03:00:30	1	4.55	66.1	. . .
2003-01-28 03:01:00	1	4.55	NULL	. . .
2003-01-28 03:01:00	2	4.45	66.7	. . .

Figure 2: Weather station data in the database

For some instruments, there are more than one data table. For example, the optical lidar data is divided in tables `ceilo` and `ceilo_backscatter` as shown in figure 3. The `ceilo` table is the main data table and `ceilo_backscatter` contains more detailed backscatter data, separated one height per row. These values are linked to the main data table using observation id numbers, generated automatically by the database. Each observation id refers to one measurement, and the respective time stamps and parameter set id numbers are stored in the main data table.



ceilo

time_utc	paramset	obs_id	cb_height_1	other columns
2003-01-28 14:01:00	1	1247	770	. . .
2003-01-28 14:01:30	1	1248	790	. . .
2003-01-28 14:02:00	1	1249	800	. . .

ceilo_backscatter

obs_id	height	bs
1247	30	0.0294
1247	60	0.0319
1247	90	0.3512
...	...	...
1248	30	0.0285
1248	60	0.0299

Figure 3: Data tables for the optical lidar

2.3 Common columns in all data tables

The column names in data tables naturally vary depending on the instrument. However, there are a few columns which are the same for all instruments. These columns are described below.

time_utc Contains the time stamp of the measurement. All times are stored in coordinated universal time (UTC).

paramset Contains the parameter set id number which allows retrieving the associated data from the parameters table. The parameter sets can be used to separate data between multiple similar instruments, different settings, locations or observation campaigns.

reliability Contains a value describing the reliability of the measurement. At the moment, a policy for the reliability values has not been decided yet, so this field is reserved for the future.

The additional data tables such as `ceilo_backscatter` don't contain these columns. However, they are linked to main tables via observation id's so the same information can be easily retrieved.

2.4 SQL language

Wakasa database is a relational database, at least currently using the PostgreSQL engine [1]. Standard Query Language, SQL in short, is a language used for inserting, deleting and retrieving data from relational databases. It has been widely documented in many different books and other sources [2] [3] [4] [5]. A few examples on how to retrieve data from the Wakasa database are given below.

Retrieving temperatures as measured by weather station 1 on January 27, 2003 (UTC time):

```
  SELECT time_utc, temperature FROM weatherstation
   WHERE paramset = '1'
     AND time_utc >= '2003-01-27 00:00:00'
     AND time_utc < '2003-01-28 00:00:00';
ORDER BY time_utc;
```

The last line (`ORDER BY time_utc`) makes sure that the results are presented in ascending order of timestamps. Without the ORDER BY clause they could be in arbitrary order. The following example is similar to previous one, but retrieves also humidity values and sets an additional condition, temperature lower than 3 degrees celsius:

```
  SELECT time_utc, temperature, humidity FROM weatherstation
   WHERE paramset = '1'
     AND time_utc >= '2003-01-27 00:00:00'
     AND time_utc < '2003-01-28 00:00:00'
     AND temperature < '3';
ORDER BY time_utc;
```

Retrieving the id number of videodata parameter set called `wakasa2003`:

```
  SELECT paramset_id FROM videodata_parameters
   WHERE name = 'wakasa2003';
```

Retrieving number of detected snowflakes at each measurement interval and their diameter distribution as measured by the image processing system on January 28, 2003 at 18:20:00-18:30:00, Japanese time. Note that the timestamps will have to be converted to UTC time for the query:

```
  SELECT time_utc, flakes_diameter, diameter_distribution FROM videodata
   WHERE paramset = '1'
     AND time_utc >= '2003-01-28 09:20:00'
     AND time_utc <= '2003-01-28 09:30:00'
ORDER BY time_utc;
```

Combining two previous queries so that the parameter set name is used directly. Note that the parameter set name and the retrieved data are in different tables, so we have to tell the database how the data in these tables can be joined. In this case, `paramset` column in `videodata` must have the same value than `paramset_id` column in `videodata_parameters`:

```
  SELECT time_utc, flakes_diameter, diameter_distribution
    FROM videodata, videodata_parameters
   WHERE videodata_parameters.name = 'wakasa2003'
     AND videodata_parameters.paramset_id = videodata.paramset
     AND time_utc >= '2003-01-28 09:20:00'
     AND time_utc <= '2003-01-28 09:30:00'
ORDER BY time_utc;
```

Retrieving optical lidar lowest cloudbase height value and backscatter profile between heights 0 and 2000 m on January 28, 2003 at 18:00:00-19:00:00 Japanese time. The backscatter profile is stored in a separate table `ceilo_backscatter` so we must use the `obs_id` fields to link these values to timestamps stored in the `ceilo` table. Results are sorted primarily by timestamp and secondarily by backscatter table height column:

```
  SELECT time_utc, cb_height_1, ceilo_backscatter.height, ceilo_backscatter.bs
    FROM ceilo, ceilo_backscatter
   WHERE paramset = '1'
     AND time_utc >= '2003-01-28 09:00:00'
     AND time_utc <= '2003-01-28 10:00:00'
     AND ceilo.obs_id = ceilo_backscatter.obs_id
     AND ceilo_backscatter.height < '2000'
ORDER BY time_utc, ceilo_backscatter.height;
```

The user is encouraged to try the given examples and experiment with more SQL commands.

2.5 Differences between the standard and array version

As mentioned previously, there are two versions of the database: standard and array version. The array version takes advantage of a PostgreSQL specific extension to standard SQL. The array datatype allows to store several values inside one table cell but still access any element of the array individually.

The only differences are related to optical lidar (Ceilometer) and MRR radar data, tables for all other instruments are identical. In the array version, tables `ceilo_backscatter`, `mrr_data` and `mrr_raw_data` don't exist. Their contents have been merged to tables `ceilo`, `mrr` and `mrr_raw` as arrays. An example of storing ceilometer data in the array version is shown in figure 4.

ceilo							
time_utc	**paramset**	**cb_height_1**	**. . .**	**bs**			
2003-01-28 14:01:00	1	770	. . .	0.0294	0.0319	. . .	0.0526
2003-01-28 14:01:30	1	790	. . .	0.0285	0.0299	. . .	0.0135
2003-01-28 14:02:00	1	800	. . .	0.0330	0.0389	. . .	0.0013

Figure 4: Optical lidar data table, array version.

Compare this figure with the figure 3 which shows the table structure of the standard version. The backscatter values have been moved from the separate table `ceilo_backscatter` to the last column of the main table.

The advantage of the array version is increased performance. It is especially significant when inserting or retrieving long time ranges of data using scripting languages. The main problem is not the time consumed by the database for finding the right section of the table but rather the large number of rows to be processed. Using arrays permits to reduce the number of rows significantly. It would be also possible to encode same data in the cell using a text string, but then it would no longer be possible to retrieve individual values.

A limitation of the PostgreSQL array extension is that NULL values cannot be stored as array elements. In all other tables and columns, NULL is used to signify a missing value. However, in the array version missing values inside arrays are noted with the number `-9999`. Therefore, this value should be discarded or treated specially when analyzing optical lidar or MRR radar data. It was selected on the basis that large negative values never occur in normal MRR radar and ceilometer backscatter data. Note that zero values are perfectly possible in actual measurement data and therefore zero would not have been a good choice for marking missing data.

The array version doesn't contain the height data for each value as the separate tables in the standard version do. Therefore it is necessary for the user to calculate the correct array indices when formulating the SQL query. A comparison of retrieving optical lidar backscatter profile between heights 0 and 2000 m on January 28, 2003 at 18:00:00-19:00:00 Japanese time is shown below.

Standard version:

```
  SELECT time_utc, ceilo_backscatter.height, ceilo_backscatter.bs
    FROM ceilo, ceilo_backscatter
   WHERE paramset = '1'
     AND time_utc >= '2003-01-28 09:00:00'
     AND time_utc <= '2003-01-28 10:00:00'
     AND ceilo.obs_id = ceilo_backscatter.obs_id
     AND ceilo_backscatter.height <= '2000'
ORDER BY time_utc, ceilo_backscatter.height;
```

Array version:

```
  SELECT time_utc, bs[1:66]
    FROM ceilo
   WHERE paramset = '1'
     AND time_utc >= '2003-01-28 09:00:00'
     AND time_utc <= '2003-01-28 10:00:00'
ORDER BY time_utc;
```

The array version is shorter, but selecting the correct height range is less intuitive. Also, the results are presented a bit differently. In the first query each backscatter value is accompanied with a height value, one value per row. In the second query all backscatter values for each timestamp are displayed on the same row.

2.6 PostgreSQL interactive terminal psql

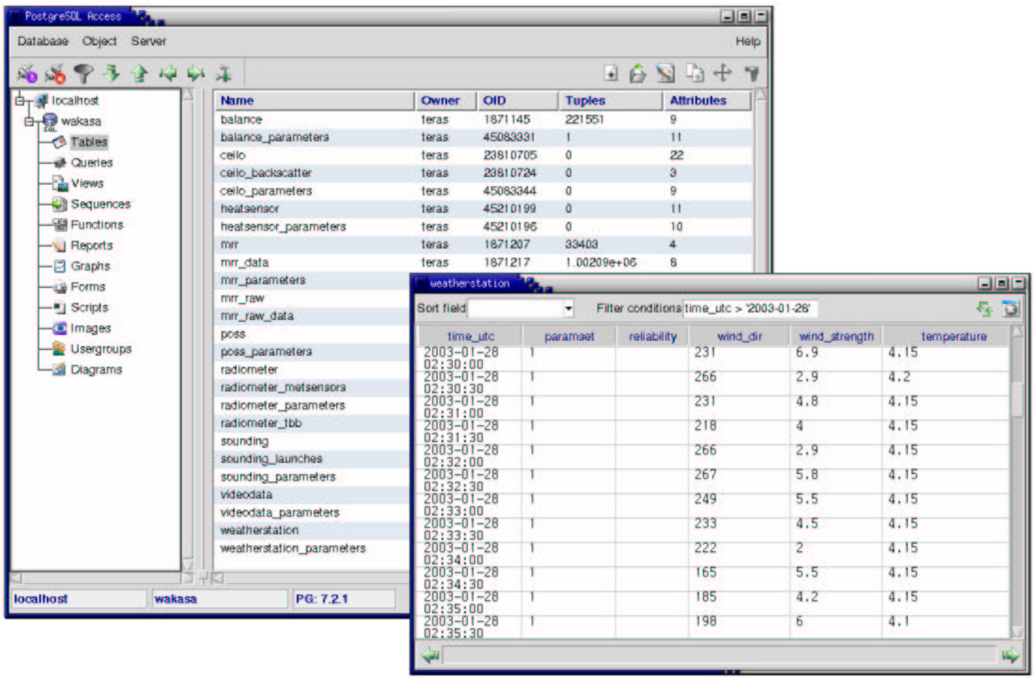
PostgreSQL interactive terminal `psql` is a command-line tool for PostgreSQL database management. Most of the functionality is necessary only for database administrators, but it can be useful also for users who are just browsing the database. For example, it can be used to give SQL commands to the database and exporting the resulting data to external files. For more information, see PostgreSQL documentation [6].

2.7 Graphical database browsers

Programs with a graphical user interface are the easiest tool for getting an overview of the database and browsing the values. There are dozens of suitable programs from various manufacturers for all major operating systems. Many of the programs also contain database management functionality.

Some of these programs are specific to a database of one certain manufacturer, but many take a generic approach and can be used with almost any relational database. The disadvantage of the generic programs is that they often don't support special features of each database engine.

Two graphical database browsing / management tools were used during the Wakasa database development: Pgaccess [7] and Phppgadmin [8]. Figure 5 shows an example screenshot of Pgaccess.



Name	Owner	OID	Tuples	Attributes
balance	teras	1871145	221551	9
balance_parameters	teras	45083331	1	11
ceilo	teras	23810705	0	22
ceilo_backscatter	teras	23810724	0	3
ceilo_parameters	teras	45083344	0	9
heatsensor	teras	45210199	0	11
heatsensor_parameters	teras	45210196	0	10
mrr	teras	1871207	33403	4
mrr_data	teras	1871217	1.00209e+06	8
mrr_parameters				
mrr_raw				
mrr_raw_data				
poss				
poss_parameters				
radiometer				
radiometer_metsensors				
radiometer_parameters				
radiometer_tbb				
sounding				
sounding_launches				
sounding_parameters				
videodata				
videodata_parameters				
weatherstation				
weatherstation_parameters				

Sort field

Filter conditions time_utc > '2003-01-28'

time_utc	paramset	reliability	wind_dir	wind_strength	temperature
2003-01-28 02:30:00	1		231	6.9	4.15
2003-01-28 02:30:30	1		266	2.9	4.2
2003-01-28 02:31:00	1		231	4.8	4.15
2003-01-28 02:31:30	1		218	4	4.15
2003-01-28 02:32:00	1		266	2.9	4.15
2003-01-28 02:32:30	1		267	5.8	4.15
2003-01-28 02:33:00	1		249	5.5	4.15
2003-01-28 02:33:30	1		233	4.5	4.15
2003-01-28 02:34:00	1		222	2	4.15
2003-01-28 02:34:30	1		165	5.5	4.15
2003-01-28 02:35:00	1		185	4.2	4.15
2003-01-28 02:35:30	1		198	6	4.1

Figure 5: Browsing database tables with Pgaccess

3 The visualization tools

3.1 Overview

A set of visualization tools are available to quickly see graphically the data of each supported instrument. The tools have been written using the Python [9] programming language and use Gnuplot [10] for output. They can be installed either on the same computer than the database or on another computer which connects to the database server via network. See the Administrator's Guide for more information about installation and system requirements.

There is one plotting program per instrument and the programs are operated from the command line. This makes it easy to write scripts which generate large sets of graphs automatically for a long time period. Advanced users can also directly modify the source code of the plotting scripts. The available plotters are listed below.

Table 1: Available plotting scripts and supported plot types

Instrument	Executable name	Supported plots
Electronic balance	plotbalance.py	Precipitation rate
Optical lidar	plotceilo.py	Lowest cloud base height Integrated backscatter Backscatter height profile

(Continued on next page)

(Continued from previous page)

Instrument	Executable name	Supported plots
Heat sensor	plotheatsensor.py	Air temperature Heat needed to melt snow Heat needed to keep the sensor board from freezing Number of snow particles detected Water detection signal Snow detection signal Snow accretion detection signal Freezing detection signal
MRR-2 radar	plotmrr.py	Integrated reflectivity Reflectivity height profile Average rain rate in the height range Rain rate height profile Average liquid water content Liquid water content height profile Average group velocity of precipitation particles Group velocity height profile
POSS radar	plotposs.py	Reflectivity
Radiometer	plotradiometer.py	Brightness temperatures of available frequencies Temperature (meteorological sensors) Humidity (meteorological sensors) Air pressure (meteorological sensors) Rain flage (meteorological sensors)
Radio sounding	plotsonde.py	Temperature Humidity Dew point Air pressure Wind direction Wind speed
Video based observation system (Kanazawa U.)	plotvideo.py	Snowflake number concentration Average diameter Average velocity Diameter distribution during a given time period Velocity distribution during a given time period Diameter distribution map Velocity distribution map
Weather station	plotweather.py	Temperature Humidity Wind direction Wind speed Air pressure Solar radiation Rain rate

3.2 Configuration file plotter.conf

All the plotting scripts use a common configuration file which contains necessary information on how to connect to the database. The default name of the configuration file is `plotter.conf` but it is also possible to specify an alternative file using a command line parameter (see section 3.3). An example configuration file is given below.

```
# Configuration file for the plotting scripts. Syntax similar as
# in Windows INI files, parsable using ConfigParser class in Python.
# Note that the entries are case sensitive.

[Database]
host: localhost
port: 5432
name: wakasa
username: teras
passwd:

[Postgresql]
use_arrays: 0

[Gnuplot]
path: /usr/local/bin/gnuplot
```

The configuration file can be modified using a normal text editor. There are three sections enclosed in brackets — `Database`, `Postgresql` and `Gnuplot` — and a number of key-value pairs under each section. Lines beginning with the '`#`' character are treated as comments. The key-value pairs are described in the following table.

Table 2: Configuration file entries

Database section	
host	Host name or IP address of the database server.
port	TCP/IP port number on the database server accepting connections from clients.
name	Name of the database to connect to. One database server can have multiple databases.
username	User name to use for the database connection. Note that database user accounts are separate from the system user accounts (accounts which are used to log in to the computer).
password	Password for the database user account.
Postgresql section	
use_arrays	Specifies the Wakasa database version in use: 0 for standard version, 1 for the version using PostgreSQL array extensions. This setting must match the actual database layout, or some of the plotters will not work.

Gnuplot section	
path	Specifies the location of the gnuplot plotting tool. This entry is not obligatory, if gnuplot can be found in one of the directories specified in the PATH environment variable.

3.3 Syntax

The command line syntax of all the plotting scripts is of the following form (replace XXX by the actual name of the plotting script):

```
plotXXX.py [-s START_TIME] [-e END_TIME] <optional parameters>
```

or

```
plotXXX.py [-d DATA_TYPE] [-s START_TIME] [-e END_TIME] <optional parameters>
```

That is, the user must supply at least the start and end time of the plot. Most plotters can visualize several types of data produced by the same instrument, in these cases it is necessary to also select the data type to plot using the `-d` (`--data`) parameter. The order of the parameters is not significant, any order is accepted.

All plotters also accept a large number of optional parameters. Most of them are shared, some are specific to each plotter. The most common parameters have both a one-letter shortcut and a longer name, for example `-s "2003-01-28 15:00:00+09"` and `--starttime="2003-01-28 15:00:00+09"` are equivalent ways of specifying the start time of the plot. More rarely needed parameters have the longer form only. A list of available parameters can be always obtained using the `-h` (`--help`) parameter, for example `plotvideo.py --help`. The common set of parameters is described in the following two sections.

3.3.1 Obligatory command line parameters

Table 3: Obligatory command line parameters common to all plotters

Parameter		Description
`-d`	`--data=STRING`	Specify the data type to plot. Available choices depend on the instrument. Example: `-d temperature`
`-s`	`--starttime=TIME`	Specify the start time of the plot. Syntax: YYYY-MM-DD HH:MM:SSZZZ, where ZZZ is the time zone information (ISO 8601 notation). Example: `-s "2003-01-27 21:02:50+09"`

(Continued on next page)

(Continued from previous page)

Parameter		**Description**
`-e`	`--endtime=TIME`	Specify the end time of the plot. Syntax: as starttime. Example: `-e "2003-01-27 21:02:50+09"`

3.3.2 Optional command line parameters

Table 4: Optional command line parameters common to all plotters

Parameter		**Description**
`-h`	`--help`	Show the help screen
`-c`	`--conffile=STRING`	Specify an alternative configuration file (default `plotter.conf`). See section 3.2 for configuration file syntax. Example: `-c otherdb.conf`
`-p`	`--paramset=NAME`	Use the given paramset when retrieving data from the database. If data from several devices or settings is in the same table, this parameter can be used to select the right dataset. The NAME should correspond to the value in the name column of the device parameters table. By default, the first available parameter set is used. To avoid errors, it is a good idea to always specify the parameter set explicitly. Examples: `-p wakasa2003`, `-p kanazawa2002`
`-v`	`--verbose`	Verbose output. This option can be given several times to reach the desired verbosity level. By default, the level is 0 and only errors are printed during processing. Level 1 prints some status information and level 2 is useful for development and debugging, as all SQL queries and other detailed information is printed. Examples: `-v` (verbosity level 1), `-v -v` (verbosity level 2)
`-o`	`--outputfile=NAME`	Direct plot output to an external file instead of screen. Portable network graphics (png) and encapsulated postscript (eps) formats are supported. To select the format, use a file name which ends either in `.png` or `.eps`. Examples: `-o graph1.png`, `--outputfile=graph2.eps`
	`--gpcmdfile=NAME`	Direct Gnuplot commands to an external file. This allows the user to see which Gnuplot commands are used to produce the plot. Example: `--gpcmdfile=graph1.gnuplot`

(Continued on next page)

(Continued from previous page)

Parameter		Description
	`--gpdatafile=NAME`	Direct plot data to an external file, in Gnuplot format. By default, a temporary file is used and deleted when the plot is ready. This allows the user to save the data for further inspection. It is also very practical together with `--gpcmdfile` parameter — the commands can be edited and the data replotted using Gnuplot without retrieving it again from the database. Note that the output file may not contain exactly the same data as in the database, for example `--timeres` and `--avg` options modify the data points before plotting. Example: `--gpcmdfile=graph1.data`
`-l`	`--localtime`	Use local time when plotting. By default, the time axis in the plots is in UTC time, independent of the format in which the start and end times are given. Note that the local time refers to the local time at the moment the instrument was used for measurements (the offset is retrieved from the parameters table), which may not be the same than local time when doing analysis. For example, the data may have been recorded in China, and the user may be doing the analysis in Japan. Example: `-l`
	`--timeres=N`	Time resolution of the plot in seconds. If the instrument recorded data at a better resolution, samples during N seconds are averaged to produce each data point. If the data resolution is not sufficient (instrument recorded the data in longer intervals), this parameter will be ignored. Example: `--timeres=60` (one minute time resolution)
	`--avg=N`	Use moving average of N values. N successive measurements are averaged to produce each data point. The number of total data points does not change, as when using the `--timeres` option. The moving average is processed after the time resolution adjustment, so these two options can also be used together. Examples: `--avg=5` (moving average of 5 values, time resolution unknown) `--timeres=60 --avg=3` (time resolution 1 minute, each data point replaced by the average value during 3 minutes)
	`--xrange=START:END`	Set the x axis range of the plot. By default, the range is chosen automatically. START and END are numbers and separated by a colon, same format as in Gnuplot `set xrange` command, except without the enclosing brackets. The given value is passed directly to Gnuplot, see Gnuplot documentation for more information. Example: `--xrange=-5:5`
	`--yrange=START:END`	Set the y axis range of the plot. Syntax as in `--xrange`.

(Continued on next page)

(Continued from previous page)

Parameter		**Description**
	`--zrange=START:END`	Set the z axis range of the plot, in three dimensional plots. Syntax as in `--xrange`.
	`--cbrange=START:END`	Set the color map range of the plot, in 2D color map plots. Syntax as in `--xrange`.
	`--title=STRING`	Set the title of the plot. By default, a title is chosen by the plotter depending on the type of values being plotted. This option is passed directly to Gnuplot. Example: `--title="Temperature on Jan 28, 2003"`
	`--xlabel=STRING`	Set the x axis label of the plot. Syntax as in `--title`.
	`--ylabel=STRING`	Set the y axis label of the plot. Syntax as in `--title`.
	`--zlabel=STRING`	Set the z axis label of the plot. Syntax as in `--title`.
	`--cblabel=STRING`	Set the color map label of the plot. Syntax as in `--title`.
	`--font=STRING`	Set the font of titles and labels. This sets the default font for all titles and labels, they can be also specified individually using the `--titlefont`, `--xfont`, `--yfont`, `--zfont` and `--cbfont` parameters. Both font family and size can be specified, see Gnuplot documentation [11] for more information. Example: `--font="Helvetica, 24"`
	`--titlefont=STRING`	Set the font of the plot title. Syntax as in `--font`.
	`--xfont=STRING`	Set the font of the x axis label. Syntax as in `--font`.
	`--yfont=STRING`	Set the font of the y axis label. Syntax as in `--font`.
	`--zfont=STRING`	Set the font of the z axis label. Syntax as in `--font`.
	`--cbfont=STRING`	Set the font of the color map label. Syntax as in `--font`.

	`--plottype=STRING`	Plot type, directly passed to Gnuplot as the "with" argument in the Gnuplot plot command. See Gnuplot documentation for syntax.
		Examples: `--plottype="lines"`,
		`--plottype="linespoints"`
		`--plottype="boxes"`

	`--pointsize=N`	Set the point size used in plots. N is an integer or floating point number, and directly passed to Gnuplot, see Gnuplot documentation for more information. Default point size is 1.0. This option is especially useful in 2D color map plots — if the plot contains white stripes increase the pointsize to produce a smooth surface. Example: `--pointsize=1.5`
	`--palette=STRING`	Adjust the palette in color map plots. This value is directly passed to Gnuplot, see Gnuplot documentation for more information. Examples: `--palette="negative"` (inverted colors), `--palette="gray negative"` (inverted colors, grayscale)

3.4 Details of each plotter

3.4.1 Plotbalance.py

`Plotbalance.py` can be used to plot precipitation rate measured using an electronic balance. There are no special plotter specific parameters.

The precipitation rate is measured using the weight difference between successive measurements of the balance. Therefore the rate can also be negative, if the weather is fine and water or snow inside the measurement box (placed on top of the balance) is evaporating. Also, when the measurement box is temporarely taken away (to empty it from snow), the precipitation rate graph shows a high positive and negative peak.

At least in the configuration used in Kanazawa university, the weight is recorded using very high time resolution and alternates rapidly due to wind and other factors. Therefore the optional parameters `--timeres` and `--avg` are especially useful with this plotter.

3.4.2 Plotceilo.py

`Plotceilo.py` is a plotter for the Vaisala Ceilometer CT25K optical lidar data. It can be used to plot the lowest cloud base height, integrated optical backscatter for a chosen height range or backscatter height profile. The plotter specific parameters accepted by `plotceilo.py` are listed in the following table.

Table 5: `plotceilo.py` specific command line parameters

Parameter		Description
`-d`	`--data=STRING`	Specify the data type to plot. Accepted values are: `lowestcb` lowest cloud base height `bsint` integrated optical backscatter in the chosen height range (default: from 0 to 7680 m) `bsmap` optical backscatter map of the chosen height range (default: from 0 to 7680 m) Example: `-d lowestcb`
	`--height=STRING`	Height range to use in backscatter plots, in meters. The device range is from 0 to 7680 meters, at 30 meter height resolution. Example: `--height=0:1000` (use data between 0 m and 1000 m above the ground)
	`--centerheights`	Shift the height values retrieved from the database so that the dots in the resulting plot are at the centers of the range gates. (default: dots at the top of each range gate)

Retrieving a long time period of the backscatter height profile data from the database is quite slow, especially in the standard (no arrays) version of the system. Therefore it is a good idea to limit the height range using the `--height` parameter, there is rarely anything interesting to be seen above 2000 meters of altitude. On the other hand, plotting integrated backscatter of the whole height range is fast, because the sum of backscatter is stored as single value in the main `ceilo` table of the database.

3.4.3 Plotheatsensor.py

`Plotheatsensor.py` is a plotter for the Yamada Giken snow heat capacity measuring device data. It can be used to see how much energy was needed to melt snow falling on the sensor, and also plot the state of various on-off type status signals provided by the device. The plotter specific parameters accepted by `plotheatsensor.py` are listed in the following table.

Table 6: `plotheatsensor.py` specific command line parameters

Parameter		**Description**	
`-d`	`--data=STRING`	Specify the data type to plot. Accepted values are:	
		`temperature`	Air temperature
		`heat_melting`	Amount of heat needed to melt snow falling on the sensor
		`heat_freezing`	Amoung of heat needed to keep the sensor board from freezing
		`snowparticles`	Number of snow particles detected
		`water_flag`	Water detection signal
		`snow_flag`	Snow detection signal
		`snow_accretion_flag`	Snow accretion detection signal
		`freezing_flag`	Freezing detection signal
		Example: `-d heat_melting`	

3.4.4 Plotmrr.py

`Plotmrr.py` is a plotter for the Metek MRR-2 Micro Range Radar data. It can be used to plot the reflectivity, rain rate, liquid water content and group velocity data provided by the radar. The plotter specific parameters accepted by `plotmrr.py` are listed in the following table.

Table 7: `plotmrr.py` specific command line parameters

Parameter		Description
`-d`	`--data=STRING`	Specify the data type to plot. Accepted values are: `zint` Integrated reflectivity in the given height range (default height range is from 0 m to maximum height measured by the radar, the maximum depends on radar settings). `zmap` Reflectivity height profile `rravg` Average rain rate in the height range `rrmap` Rain rate height profile `lwcavg` Average liquid water content in the height range `lwcmap` Liquid water content height profile `wavg` Average group velocity in the height range `wmap` Group velocity height profile Example: `-d zmap`
	`--height=STRING`	Height range to use in backscatter plots, in meters. The available range depends on the radar settings, starting from ground level (0 m) and going up to 6000 m. Example: `--height=0:1000` (use data between 0 m and 1000 m above the ground)
	`--centerheights`	Shift the height values retrieved from the database so that the dots in the resulting plot are at the centers of the range gates. (default: dots at the top of each range gate)

Note that the rain rate and liquid water content values provided by the MRR are calculated and based on the assumption that the precipitation type is water. Therefore they cannot be directly used for snowfall. Practical experience has shown that even in the case of rain the instrument may give clearly wrong values for the rain rate, and have gaps in the data. The reflectivity is also affected by many factors, including snow falling on the device, which must be taken account in interpreting the results.

The MRR-2 radar can be set to operate using different height ranges and resolutions, resolution decreasing as the range increases. The radar also has a mode of quickly alternating between several height range / resolution settings. Therefore, even during the same time period there may be data of several height resolutions from a single radar. In the database, these are distinguished using different parameter sets, and can be selected using the `--paramset` option of `plotmrr.py`.

Using default values, the height profile plots often contain blank area between the height steps. This is a limitation of Gnuplot, it cannot (at least not in current version) automatically produce a filled map. However, it is possible to produce nice looking plots by manually increasing the point size, using the `--pointsize` command line parameter.

3.4.5 Plotposs.py

`Plotposs.py` is a plotter for the Precipitation Occurrence Sensor System (POSS) data. It can be used to plot the reflectivity values reported by POSS. The plotter specific parameters accepted by `plotposs.py` are listed in the following table.

Table 8: `plotposs.py` specific command line parameters

Parameter		Description
`-d`	`--data=STRING`	Specify the data type to plot. Accepted values are: `reflectivity` Reflectivity value Example: `-d reflectivity` (As `reflectivity` is currently the only supported data type, this parameter is optional.)

3.4.6 Plotradiometer.py

`Plotradiometer.py` is a plotter for radiometer data. Radiometers of several manufacturers are supported. It can be used to plot the brightness temperatures of frequencies measured by the radiometer (currently only single frequency in one plot), and weather related data as measured by meteorological sensors attached to some radiometers. The plotter specific parameters accepted by `plotradiometer.py` are listed in the following table.

Table 9: `plotradiometer.py` specific command line parameters

Parameter		Description
`-d`	`--data=STRING`	Specify the data type to plot. Accepted values are: `bt(X)` Brightness temperature of frequency X (in GHz). `temperature` Air temperature (meteorological sensors) `humidity` Relative humidity (meteorological sensors) `air_pres` Air pressure (meteorological sensors) `rain_flag` Rain flag (meteorological sensors) Example: `-d bt(23.8)`

The frequency X in data type `bt(X)` must be a frequency which is found in the database for the given radiometer model. If some other frequency is given, the query will return no data and no plot will be produced. Meteorological sensors are present in only certain radiometers, so the weather data plots are available only for such models.

3.4.7 Plotsonde.py

`Plotsonde.py` is a plotter for radio sounding data. It can be used to plot various weather related values as reported by radio soundings. Unlike the weather station plotter `plotweather.py`, the values are plotted against the height of the sounding, not time. The plotter specific parameters accepted by `plotsonde.py` are listed in the following table.

Table 10: `plotsonde.py` specific command line parameters

Parameter		Description	
`-d`	`--data=STRING`	Specify the data type to plot. Accepted values are:	
		`wind_dir`	Wind direction (degrees clockwise, 0 = north)
		`wind_speed`	Wind speed
		`temperature`	Air temperature
		`humidity`	Relative humidity
		`dewpoint`	Dew point
		`air_pres`	Air pressure
		Example: `-d temperature`	

In the database, radio soundings (radio sondes) are separated by launch in the `sounding_launches` table. However, this plotter only accepts the common `--starttime` and `--endtime` options for specifying the plot range. Therefore, to plot the height profile data of a specific sonde, give a start time earlier than the sonde launch and an end time later than last data point of the same sonde. The `--timeres` parameter is not supported by `plotsonde.py`.

3.4.8 Plotvideo.py

`Plotvideo.py` is a plotter for the data produced by the video camera based snowfall observation system of Image Information Science lab, Kanazawa University. It can be used to display snowflake number concentration and size and velocity distributions, either averaged during a specified period or plotted against time. The plotter specific parameters accepted by `plotvideo.py` are listed in the following table.

Table 11: `plotvideo.py` specific command line parameters

Parameter		Description
`-d`	`--data=STRING`	Specify the data type to plot. Accepted values are: `number` Snowflake number concentration `davg` Average snowflake diameter `vavg` Average snowflake velocity `ddistr` Average diameter distribution during the given time period `vdistr` Average velocity distribution during the given time period `ddistrmap` Diameter distribution against time `vdistrmap` Velocity distribution against time Example: `-d ddistr`
	`--dres=N`	Resolution in millimeters for diameter distribution plots. Both integer and floating point values are accepted. Default is the maximum resolution of the system (depends on the camera setup). Example: `--dres=1` (resolution of 1 mm)
	`--vres=N`	Resolution in m/s for velocity distribution plots. Both integer and floating point values are accepted. Default is the maximum resolution of the system (depends on the camera setup). Example: `--vres=0.1` (resolution of 0.1 m/s)
	`--absolute`	Plot distributions using "absolute" values for snow flake numbers. In diameter and velocity distribution plots, each bar represents a certain diameter or velocity range. If the `--absolute` option is specified, the height of each bar can be taken directly as the number of snowflakes of that diameter / velocity range in one cubic meter. Default is relative plotting, with values scaled to 1 mm (diameter) 1 m/s (velocity) so that bar height stays same independent of the resolution.

The observation system measures snowflakes in a relatively small volume, but in the plots the numbers are always scaled to a volume of one cubic meter. The volume of the observation space can vary between experiments and is therefore retrieved from the `videodata_parameters` table of the database.

To get accurate results for size and speed, flakes which are only partly seen in each image are discarded by the observation system. This causes a small error in the number of flakes occupying the observation volume. Therefore, when scaling the number of snowflake to one cubic meter, the actual volume of observation space is adjusted by using the `volume_bias_correction` value in the `videodata_parameters` table. For detailed information, refer to section B.1.22 in the appendix.

In the current version of the plotter, user-selectable diameter resolution and velocity resolution have not been implemented yet. Also, `--timeres` option is not supported for "absolute" mode plots.

3.4.9 Plotweather.py

`Plotweather.py` is a plotter for weather station data. It can be used to visualize various values reported by weather stations of several manufacturers. The plotter specific parameters accepted by `plotweather.py` are listed in the following table.

Table 12: `plotweather.py` specific command line parameters

Parameter		Description
`-d`	`--data=STRING`	Specify the data type to plot. Accepted values are: `wind_dir` Wind direction (degrees clockwise, 0 = north) `wind_speed` Wind speed `temperature` Air temperature `humidity` Relative humidity `air_pres` Air pressure `rain_rate` Precipitation rate Example: `-d temperature`

Note that the available values depend on the weather station manufacturer and station configuration. For example, not all stations can measure precipitation rate. In these cases, the query will either return no data (in which case no plot will be produced) or zero values not reflecting the actual weather conditions.

3.5 Plotter output examples

Example output from all the plotters is presented in this section. The users are encouraged to try out the examples and experiment with different command line parameters to see how they affect the result.

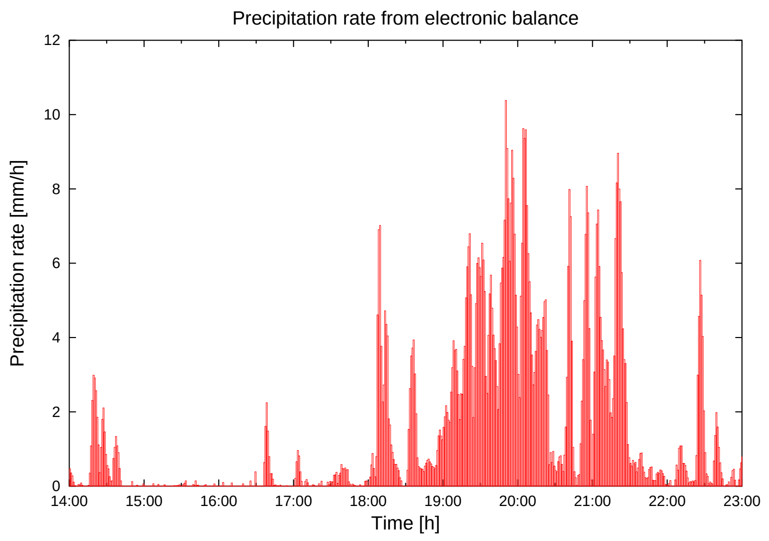


Figure 6: Example output of `plotbalance.py`. This plot was produced using the following command line: `./plotbalance.py -p wakasa2003 -s "2003-01-28 14:00:00+09" -e "2003-01-28 23:00:00+09" -l --timeres=60 --avg=3 --yrange=0:12 --font="Helvetica, 18"`

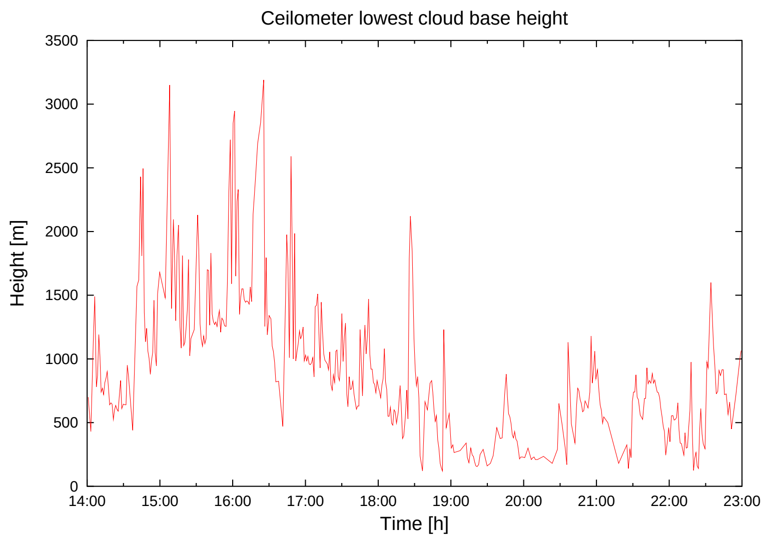


Figure 7: Example output of `plotceilo.py`. This plot was produced using the following command line: `./plotceilo.py -p wakasa2003 -d lowestcb -s "2003-01-28 14:00:00+09" -e "2003-01-28 23:00:00+09" -l --timeres=60 --font="Helvetica, 18"`

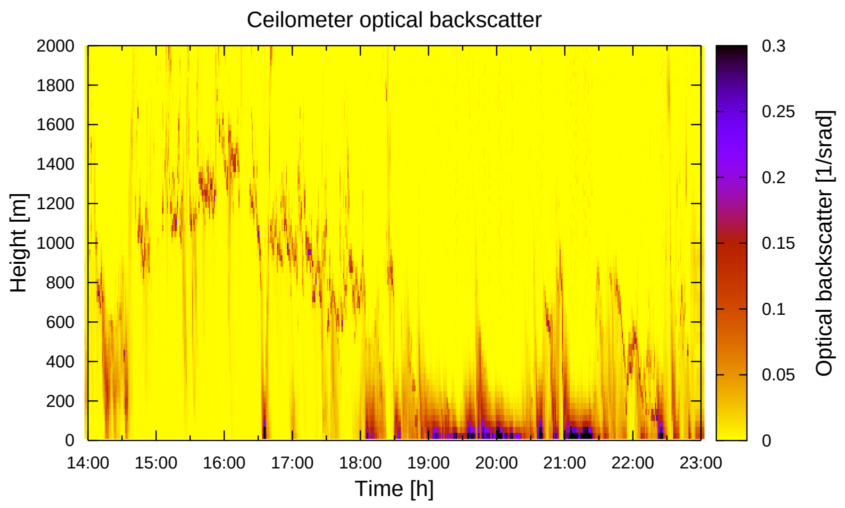


Figure 8: Example output of `plotceilo.py`. This plot was produced using the following command line: `./plotceilo.py -p wakasa2003 -d bsmap --height=0:2000 -s "2003-01-28 14:00:00+09" -e "2003-01-28 23:00:00+09" -l --cbrange=0:0.3 --palette="negative" --timeres=60 --font="Helvetica, 18"`

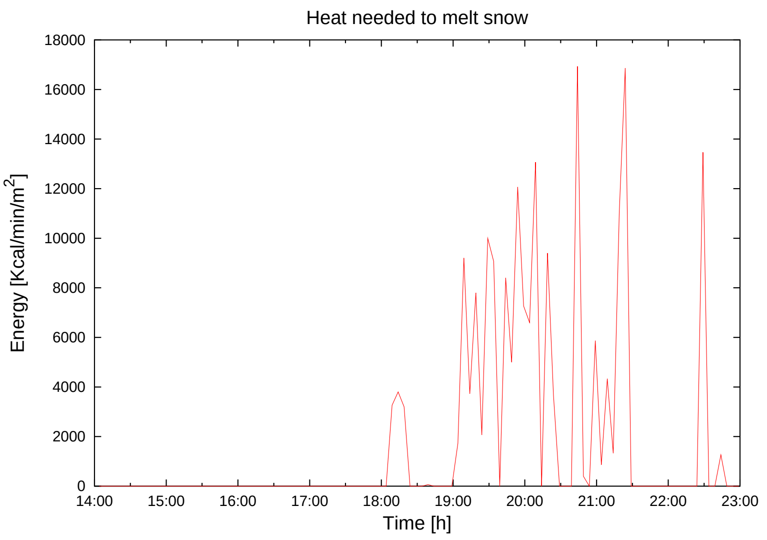


Figure 9: Example output of `plotheatsensor.py`. This plot was produced using the following command line: `./plotheatsensor.py -p wakasa2003 -d heat_melting -s "2003-01-28 14:00:00+09" -e "2003-01-28 23:00:00+09" -l --font="Helvetica, 18"`

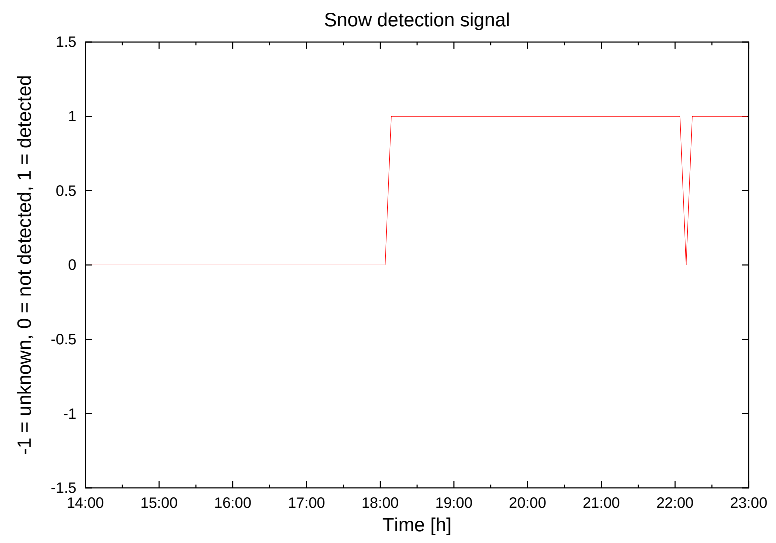


Figure 10: Example output of `plotheatsensor.py`. This plot was produced using the following command line: `./plotheatsensor.py -p wakasa2003 -d snow_flag -s "2003-01-28 14:00:00+09" -e "2003-01-28 23:00:00+09" -l --font="Helvetica, 18"`

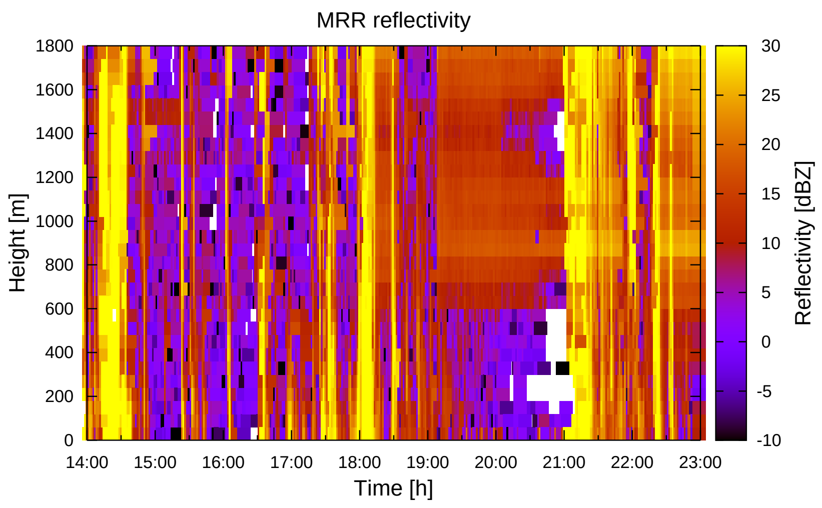


Figure 11: Example output of `plotmrr.py`. This plot was produced using the following command line: `./plotmrr.py -p wakasa2003_60m -d zmap -s "2003-01-28 14:00:00+09" -e "2003-01-28 23:00:00+09" -l --centerheights --cbrange=-10:30 --timeres=60 --font="Helvetica, 18" --pointsize=1.7`

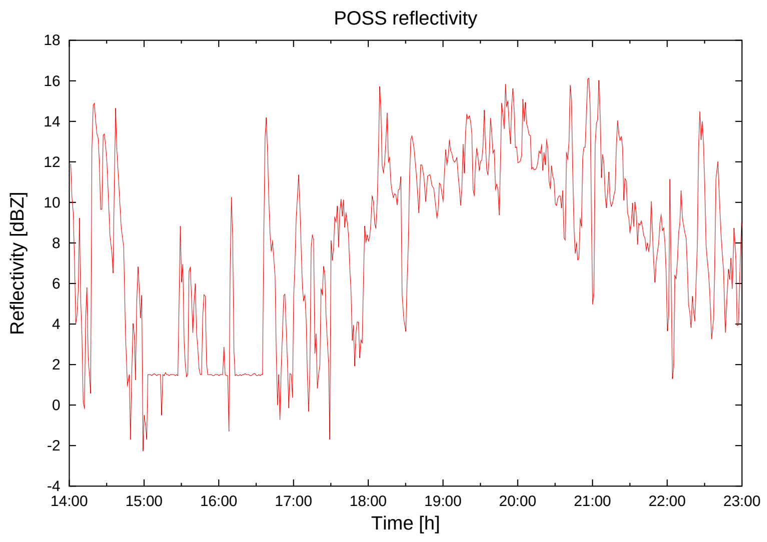


Figure 12: Example output of `plotposs.py`. This plot was produced using the following command line: `./plotposs.py -p wakasa2003 -s "2003-01-28 14:00:00+09" -e "2003-01-28 23:00:00+09" -l --timeres=60`

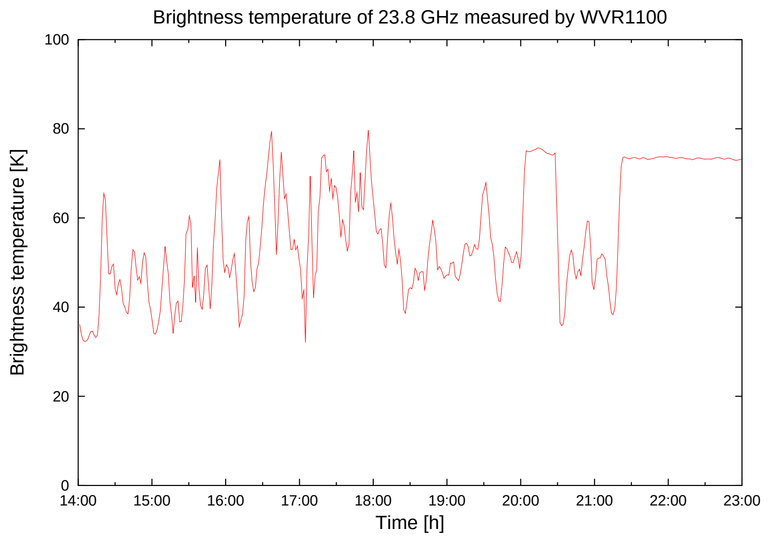


Figure 13: Example output of `plotradiometer.py`. This plot was produced using the following command line: `./plotradiometer.py -p wakasa2003-wvr -d "bt(23.8)" -s "2003-01-28 14:00:00+09" -e "2003-01-28 23:00:00+09" -l --timeres=60 --avg=3 --title="Brightness temperature of 23.8 GHz measured by WVR1100" --yrange=0:100 --font="Helvetica, 18"`

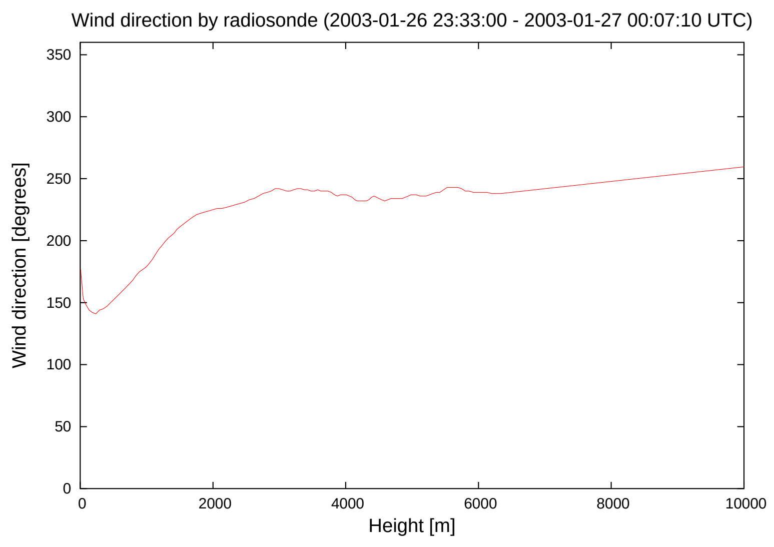


Figure 14: Example output of `plotsonde.py`. This plot was produced using the following command line: `./plotsonde.py -p wakasa2003-fukui -d wind_dir -s "2003-01-27 08:00:00+09" -e "2003-01-27 10:00:00+09" --xrange=0:10000 --yrange=0:360 --font="Helvetica, 18"`

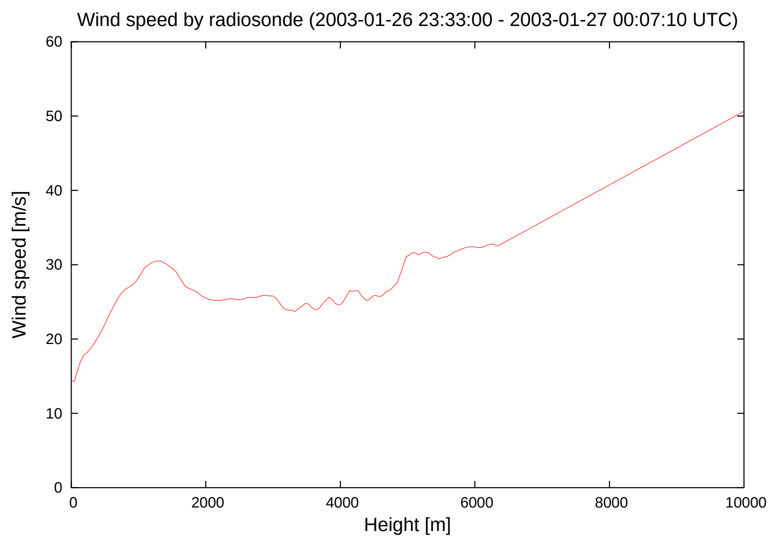


Figure 15: Example output of `plotsonde.py`. This plot was produced using the following command line: `./plotsonde.py -p wakasa2003-fukui -d wind_speed -s "2003-01-27 08:00:00+09" -e "2003-01-27 10:00:00+09" --xrange=0:10000 --yrange=0:60 --font="Helvetica, 18"`

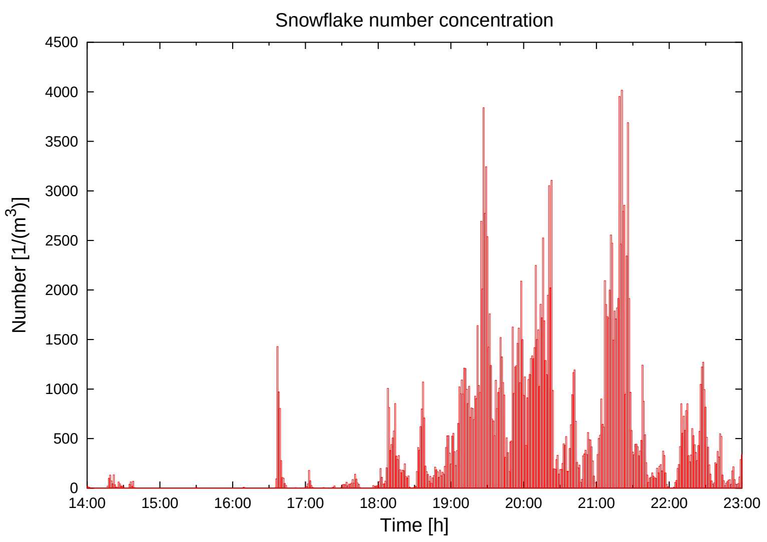


Figure 16: Example output of `plotvideo.py`. This plot was produced using the following command line: `./plotvideo.py -p wakasa2003 -d number -s "2003-01-28 14:00:00+09" -e "2003-01-28 23:00:00+09" -l --font="Helvetica, 18"`

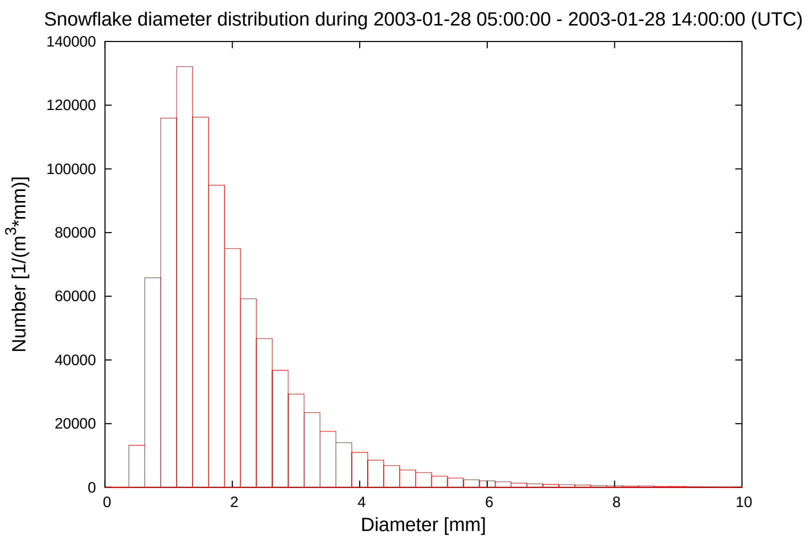


Figure 17: Example output of `plotvideo.py`. This plot was produced using the following command line: `./plotvideo.py -p wakasa2003 -d ddistr -s "2003-01-28 14:00:00+09" -e "2003-01-28 23:00:00+09" -l --xrange=0:10 --font="Helvetica, 18"`

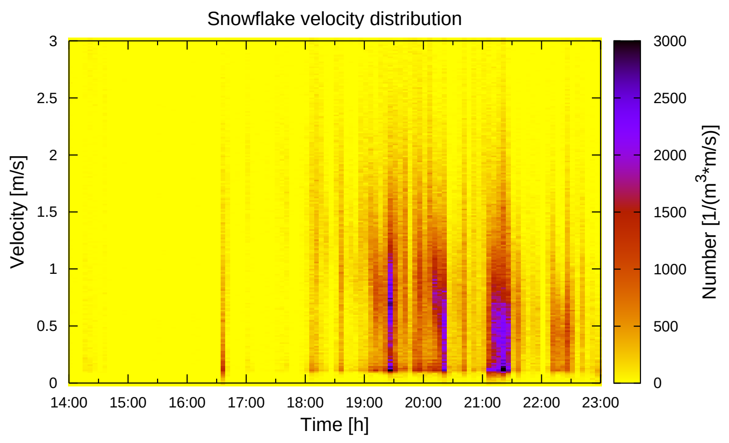


Figure 18: Example output of `plotvideo.py`. This plot was produced using the following command line: `./plotvideo.py -p wakasa2003 -d vdistrmap -s "2003-01-28 14:00:00+09" -e "2003-01-28 23:00:00+09" -l --cbrange=0:3000 --timeres=300 --palette="negative" --font="Helvetica, 18"`

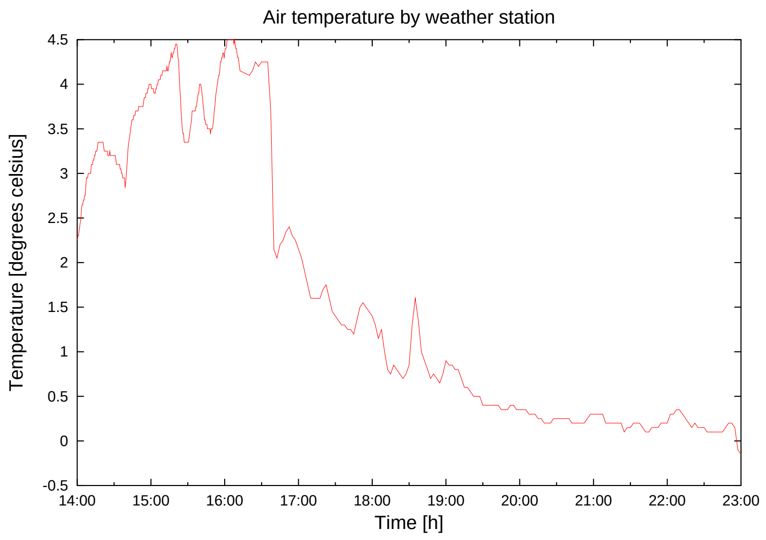


Figure 19: Example output of `plotweather.py`. This plot was produced using the following command line: `./plotweather.py -p wakasa2003 -d temperature -s "2003-01-28 14:00:00+09" -e "2003-01-28 23:00:00+09" -l --font="Helvetica, 18"`

4 Using the data in external programs

The plotting tools provide an overview of the data stored in the database. For more detailed analysis, it is usually necessary to use other tools. There are mainly two approaches: retrieving the desired part of data and converting it to a format suitable for further processing, or accessing the data in the database directly.

4.1 Exporting to other formats

Most database browsing and management tools provide an option to export data from the database to files. In many cases, a simple format such as comma separated values (csv) is sufficient and can be easily used in many external programs, such as spreadsheet applications and popular mathematics packages.

Here is a simple example of connecting to a database called `wakasa` using the `psql` program, retrieving two days of temperature and wind data from the weatherstation table and saving it in csv format to file `/tmp/weatherdata.txt`:

```
[teras@pyxis23] psql -d wakasa
Welcome to psql, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit

wakasa=> \pset format unaligned
Output format is unaligned.
wakasa=> \pset fieldsep ','
Field separator is ','.
wakasa=> \pset tuples_only
Showing only tuples.
wakasa=> \o '/tmp/weatherdata.txt'
wakasa=> SELECT time_utc, temperature, wind_dir, wind_speed
wakasa-> FROM weatherstation
wakasa-> WHERE paramset = '1'
wakasa-> AND time_utc >= '2003-01-27 00:00:00'
wakasa-> AND time_utc <= '2003-01-28 23:59:59'
wakasa-> ORDER BY time_utc;
wakasa=> \q
[teras@pyxis23]
```

As you can see, SQL queries can be written directly into the `psql` terminal window. Of course, it is also possible to write them in a file which is executed using `psql`. The beginning of the resulting file `/tmp/weatherdata.txt` looks like this:

```
2003-01-27 00:00:00,9.9,188,11.9
2003-01-27 00:02:30,9.9,54,7.1
2003-01-27 00:05:00,9.85,173,11.6
2003-01-27 00:07:30,9.8,141,7.5
2003-01-27 00:10:00,9.75,165,10
```

For learning the necessary SQL commands, one handy trick is to use verbosity level 2 (`-v -v`) of the plotting tools. In that case all the SQL commands used to retrieve the data used by the plotter are printed to standard output. For example, plotting weatherstation data using the command `plotweather.py` produces the following output (long lines wrapped):

```
[teras@pyxis23] ./plotweather.py -p wakasa2003 -d temperature -s
  "2003-01-28 00:00:00+09" -e "2003-01-28 23:59:59+09" -v -v
Connecting to database wakasa_arrays on host localhost:5432 as user teras
Connected ... Now creating a cursor
Executing query SELECT paramset_id FROM weatherstation_parameters
  WHERE name = 'wakasa2003';
Executing query SELECT time_utc, temperature FROM weatherstation
  WHERE paramset = '1' AND time_utc >= '2003-01-27 15:00:00' AND
  time_utc <= '2003-01-28 14:59:59' AND temperature IS NOT NULL
  ORDER BY time_utc;
Wrote to /tmp/@22123.0.gnuplot.data.temp
[teras@pyxis23]
```

Here you can see that two queries were executed, first one to retrieve the parameter set id number based on the given name (`wakasa2003`) and then another to retrieve the actual data. Note also that the start and end times are converted to UTC time for the queries.

4.2 Accessing data from programming languages

Practically all popular programming languages can be used to interact with databases using SQL queries. Therefore, when writing new analysis tools it is often easiest to use data in the database directly. This guide does cover this topic in more detail, the user is adviced to see PostgreSQL and the programming language documentation for more information.

The plotting tools can be used as an example of how to connect to the database and manipulate data using the Python programming language [9]. The separate Wakasa Database Developer's Guide gives more information of the source code of the plotting scripts.

5 Known bugs

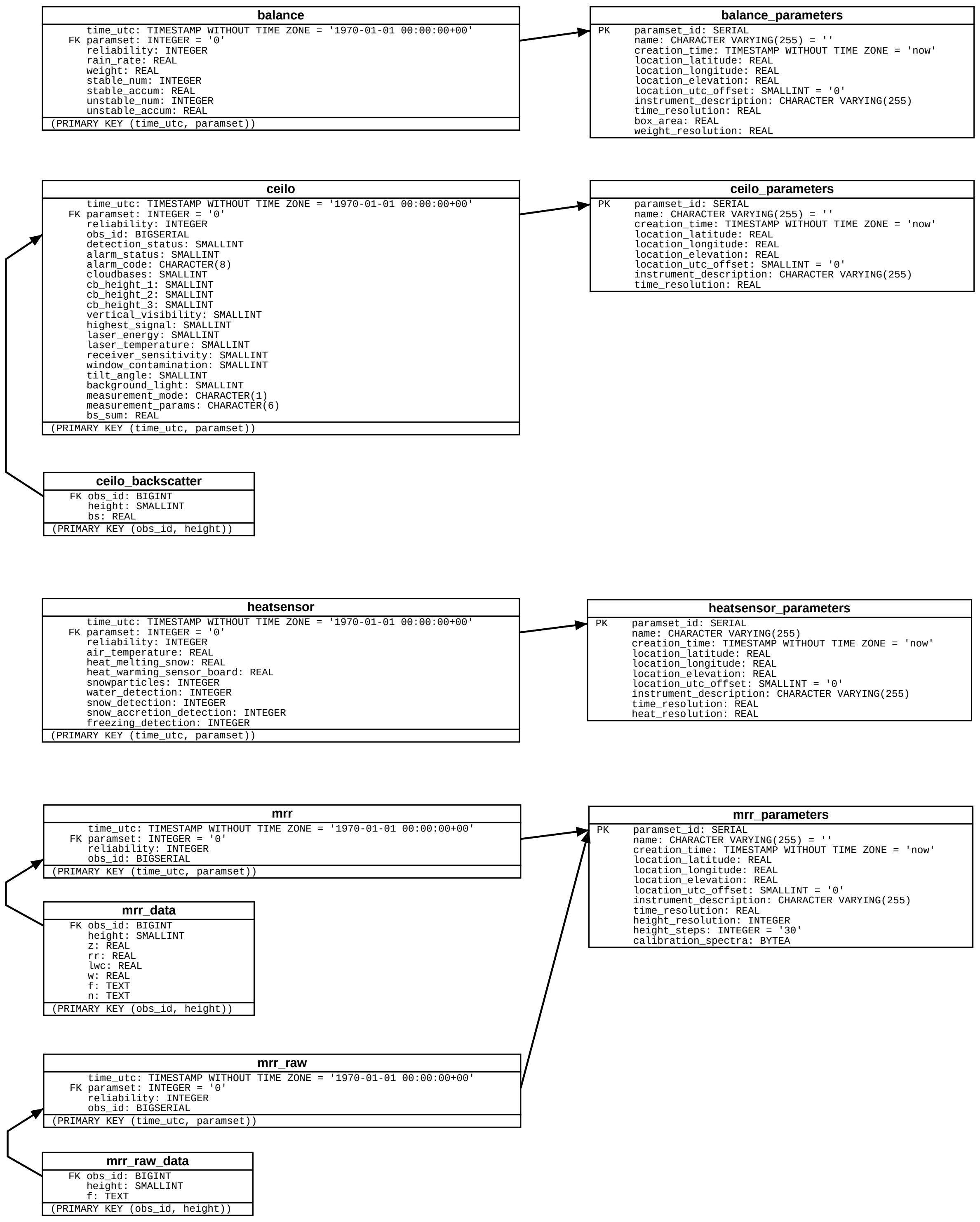
The following bugs are known in the present version of the database and plotting tools.

- All the plotters create temporary data files in Gnuplot format. However, it seems to be difficult to know when the separate Gnuplot process is finished using the data files. Therefore, the data files are not deleted but left in the /tmp directory of the system. They all end in `.gnuplot.data.temp` and can therefore be detected and manually deleted by the user.

- Adjustable diameter and velocity resolution is not yet implemented in `plotvideo.py`. Also, `--timeres` option is not implemented for some plots.

A Database table layout

A.1 Standard version



```
balance
    time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
 FK paramset: INTEGER = '0'
    reliability: INTEGER
    rain_rate: REAL
    weight: REAL
    stable_num: INTEGER
    stable_accum: REAL
    unstable_num: INTEGER
    unstable_accum: REAL
(PRIMARY KEY (time_utc, paramset))
```

```
balance_parameters
PK  paramset_id: SERIAL
    name: CHARACTER VARYING(255) = ''
    creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
    location_latitude: REAL
    location_longitude: REAL
    location_elevation: REAL
    location_utc_offset: SMALLINT = '0'
    instrument_description: CHARACTER VARYING(255)
    time_resolution: REAL
    box_area: REAL
    weight_resolution: REAL
```

```
ceilo
    time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
 FK paramset: INTEGER = '0'
    reliability: INTEGER
    obs_id: BIGSERIAL
    detection_status: SMALLINT
    alarm_status: SMALLINT
    alarm_code: CHARACTER(8)
    cloudbases: SMALLINT
    cb_height_1: SMALLINT
    cb_height_2: SMALLINT
    cb_height_3: SMALLINT
    vertical_visibility: SMALLINT
    highest_signal: SMALLINT
    laser_energy: SMALLINT
    laser_temperature: SMALLINT
    receiver_sensitivity: SMALLINT
    window_contamination: SMALLINT
    tilt_angle: SMALLINT
    background_light: SMALLINT
    measurement_mode: CHARACTER(1)
    measurement_params: CHARACTER(6)
    bs_sum: REAL
(PRIMARY KEY (time_utc, paramset))
```

```
ceilo_parameters
PK  paramset_id: SERIAL
    name: CHARACTER VARYING(255) = ''
    creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
    location_latitude: REAL
    location_longitude: REAL
    location_elevation: REAL
    location_utc_offset: SMALLINT = '0'
    instrument_description: CHARACTER VARYING(255)
    time_resolution: REAL
```

```
ceilo_backscatter
 FK obs_id: BIGINT
    height: SMALLINT
    bs: REAL
(PRIMARY KEY (obs_id, height))
```

```
heatsensor
    time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
 FK paramset: INTEGER = '0'
    reliability: INTEGER
    air_temperature: REAL
    heat_melting_snow: REAL
    heat_warming_sensor_board: REAL
    snowparticles: INTEGER
    water_detection: INTEGER
    snow_detection: INTEGER
    snow_accretion_detection: INTEGER
    freezing_detection: INTEGER
(PRIMARY KEY (time_utc, paramset))
```

```
heatsensor_parameters
PK  paramset_id: SERIAL
    name: CHARACTER VARYING(255)
    creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
    location_latitude: REAL
    location_longitude: REAL
    location_elevation: REAL
    location_utc_offset: SMALLINT = '0'
    instrument_description: CHARACTER VARYING(255)
    time_resolution: REAL
    heat_resolution: REAL
```

```
mrr
    time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
 FK paramset: INTEGER = '0'
    reliability: INTEGER
    obs_id: BIGSERIAL
(PRIMARY KEY (time_utc, paramset))
```

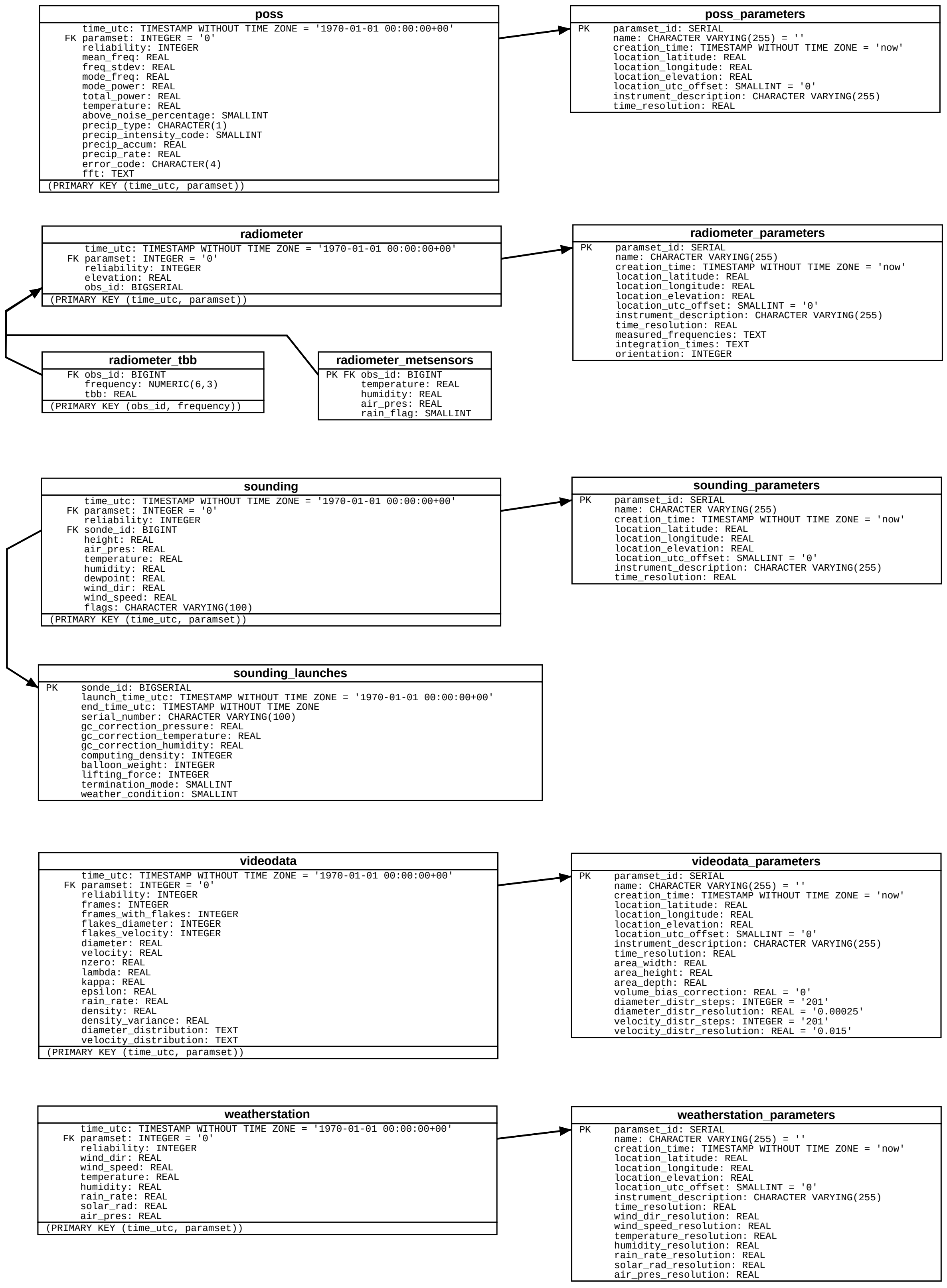
```
mrr_parameters
PK  paramset_id: SERIAL
    name: CHARACTER VARYING(255) = ''
    creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
    location_latitude: REAL
    location_longitude: REAL
    location_elevation: REAL
    location_utc_offset: SMALLINT = '0'
    instrument_description: CHARACTER VARYING(255)
    time_resolution: REAL
    height_resolution: INTEGER
    height_steps: INTEGER = '30'
    calibration_spectra: BYTEA
```

```
mrr_data
 FK obs_id: BIGINT
    height: SMALLINT
    z: REAL
    rr: REAL
    lwc: REAL
    w: REAL
    f: TEXT
    n: TEXT
(PRIMARY KEY (obs_id, height))
```

```
mrr_raw
    time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
 FK paramset: INTEGER = '0'
    reliability: INTEGER
    obs_id: BIGSERIAL
(PRIMARY KEY (time_utc, paramset))
```

```
mrr_raw_data
 FK obs_id: BIGINT
    height: SMALLINT
    f: TEXT
(PRIMARY KEY (obs_id, height))
```

Figure 20: Wakasa table layout, standard version, page 1(2)



```
poss
    time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
 FK paramset: INTEGER = '0'
    reliability: INTEGER
    mean_freq: REAL
    freq_stdev: REAL
    mode_freq: REAL
    mode_power: REAL
    total_power: REAL
    temperature: REAL
    above_noise_percentage: SMALLINT
    precip_type: CHARACTER(1)
    precip_intensity_code: SMALLINT
    precip_accum: REAL
    precip_rate: REAL
    error_code: CHARACTER(4)
    fft: TEXT
(PRIMARY KEY (time_utc, paramset))
```

```
poss_parameters
PK    paramset_id: SERIAL
      name: CHARACTER VARYING(255) = ''
      creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
      location_latitude: REAL
      location_longitude: REAL
      location_elevation: REAL
      location_utc_offset: SMALLINT = '0'
      instrument_description: CHARACTER VARYING(255)
      time_resolution: REAL
```

```
radiometer
    time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
 FK paramset: INTEGER = '0'
    reliability: INTEGER
    elevation: REAL
    obs_id: BIGSERIAL
(PRIMARY KEY (time_utc, paramset))
```

```
radiometer_parameters
PK    paramset_id: SERIAL
      name: CHARACTER VARYING(255)
      creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
      location_latitude: REAL
      location_longitude: REAL
      location_elevation: REAL
      location_utc_offset: SMALLINT = '0'
      instrument_description: CHARACTER VARYING(255)
      time_resolution: REAL
      measured_frequencies: TEXT
      integration_times: TEXT
      orientation: INTEGER
```

```
radiometer_tbb
 FK obs_id: BIGINT
    frequency: NUMERIC(6,3)
    tbb: REAL
(PRIMARY KEY (obs_id, frequency))
```

```
radiometer_metsensors
PK FK obs_id: BIGINT
      temperature: REAL
      humidity: REAL
      air_pres: REAL
      rain_flag: SMALLINT
```

```
sounding
    time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
 FK paramset: INTEGER = '0'
    reliability: INTEGER
 FK sonde_id: BIGINT
    height: REAL
    air_pres: REAL
    temperature: REAL
    humidity: REAL
    dewpoint: REAL
    wind_dir: REAL
    wind_speed: REAL
    flags: CHARACTER VARYING(100)
(PRIMARY KEY (time_utc, paramset))
```

```
sounding_parameters
PK    paramset_id: SERIAL
      name: CHARACTER VARYING(255)
      creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
      location_latitude: REAL
      location_longitude: REAL
      location_elevation: REAL
      location_utc_offset: SMALLINT = '0'
      instrument_description: CHARACTER VARYING(255)
      time_resolution: REAL
```

```
sounding_launches
PK    sonde_id: BIGSERIAL
      launch_time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
      end_time_utc: TIMESTAMP WITHOUT TIME ZONE
      serial_number: CHARACTER VARYING(100)
      gc_correction_pressure: REAL
      gc_correction_temperature: REAL
      gc_correction_humidity: REAL
      computing_density: INTEGER
      balloon_weight: INTEGER
      lifting_force: INTEGER
      termination_mode: SMALLINT
      weather_condition: SMALLINT
```

```
videodata
    time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
 FK paramset: INTEGER = '0'
    reliability: INTEGER
    frames: INTEGER
    frames_with_flakes: INTEGER
    flakes_diameter: INTEGER
    flakes_velocity: INTEGER
    diameter: REAL
    velocity: REAL
    nzero: REAL
    lambda: REAL
    kappa: REAL
    epsilon: REAL
    rain_rate: REAL
    density: REAL
    density_variance: REAL
    diameter_distribution: TEXT
    velocity_distribution: TEXT
(PRIMARY KEY (time_utc, paramset))
```

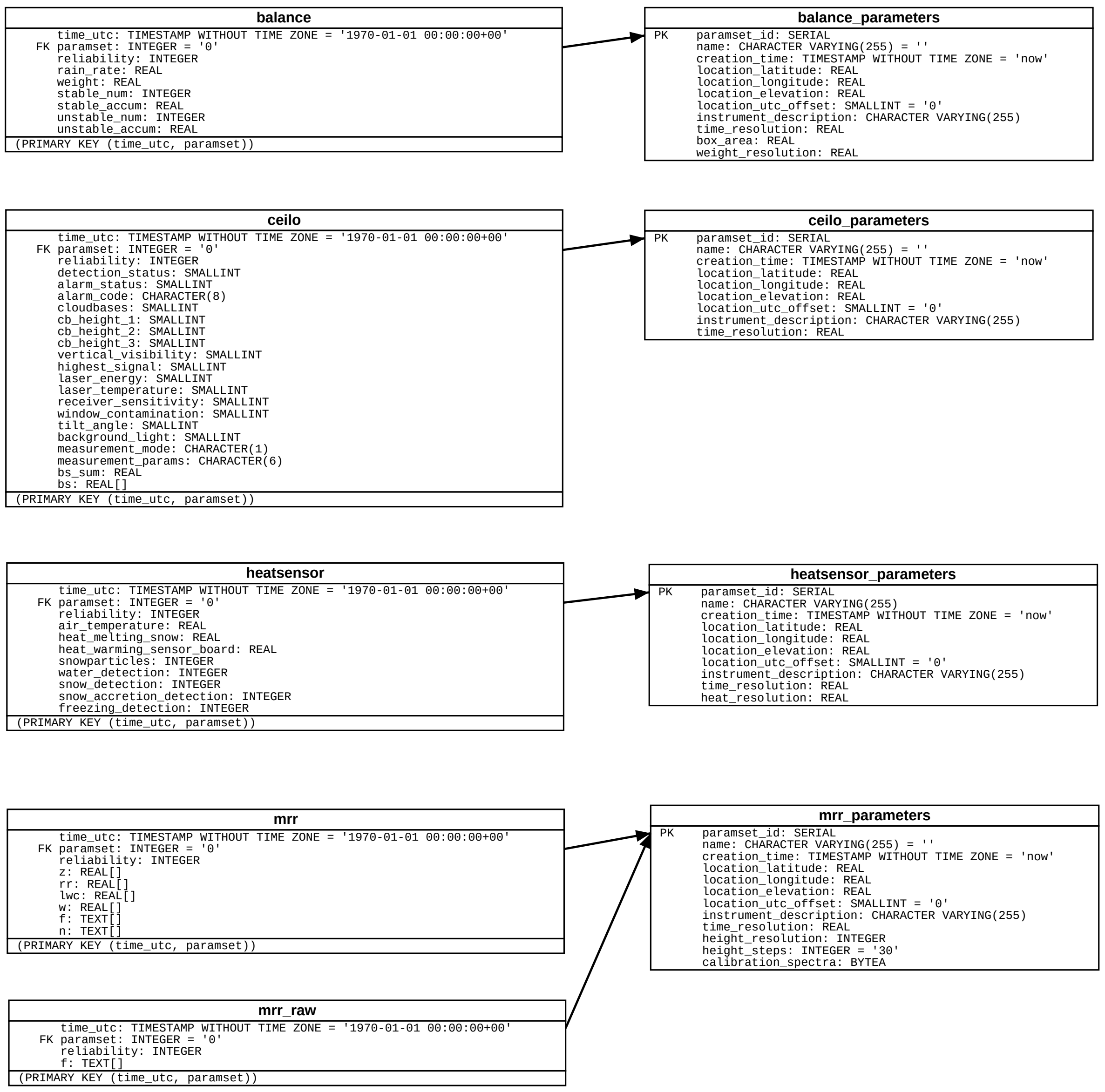
```
videodata_parameters
PK    paramset_id: SERIAL
      name: CHARACTER VARYING(255) = ''
      creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
      location_latitude: REAL
      location_longitude: REAL
      location_elevation: REAL
      location_utc_offset: SMALLINT = '0'
      instrument_description: CHARACTER VARYING(255)
      time_resolution: REAL
      area_width: REAL
      area_height: REAL
      area_depth: REAL
      volume_bias_correction: REAL = '0'
      diameter_distr_steps: INTEGER = '201'
      diameter_distr_resolution: REAL = '0.00025'
      velocity_distr_steps: INTEGER = '201'
      velocity_distr_resolution: REAL = '0.015'
```

```
weatherstation
    time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
 FK paramset: INTEGER = '0'
    reliability: INTEGER
    wind_dir: REAL
    wind_speed: REAL
    temperature: REAL
    humidity: REAL
    rain_rate: REAL
    solar_rad: REAL
    air_pres: REAL
(PRIMARY KEY (time_utc, paramset))
```

```
weatherstation_parameters
PK    paramset_id: SERIAL
      name: CHARACTER VARYING(255) = ''
      creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
      location_latitude: REAL
      location_longitude: REAL
      location_elevation: REAL
      location_utc_offset: SMALLINT = '0'
      instrument_description: CHARACTER VARYING(255)
      time_resolution: REAL
      wind_dir_resolution: REAL
      wind_speed_resolution: REAL
      temperature_resolution: REAL
      humidity_resolution: REAL
      rain_rate_resolution: REAL
      solar_rad_resolution: REAL
      air_pres_resolution: REAL
```

Figure 21: Wakasa table layout, standard version, page 2(2)

A.2 Array version



```
balance
      time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
   FK paramset: INTEGER = '0'
      reliability: INTEGER
      rain_rate: REAL
      weight: REAL
      stable_num: INTEGER
      stable_accum: REAL
      unstable_num: INTEGER
      unstable_accum: REAL
(PRIMARY KEY (time_utc, paramset))
```

```
balance_parameters
PK    paramset_id: SERIAL
      name: CHARACTER VARYING(255) = ''
      creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
      location_latitude: REAL
      location_longitude: REAL
      location_elevation: REAL
      location_utc_offset: SMALLINT = '0'
      instrument_description: CHARACTER VARYING(255)
      time_resolution: REAL
      box_area: REAL
      weight_resolution: REAL
```

```
ceilo
      time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
   FK paramset: INTEGER = '0'
      reliability: INTEGER
      detection_status: SMALLINT
      alarm_status: SMALLINT
      alarm_code: CHARACTER(8)
      cloudbases: SMALLINT
      cb_height_1: SMALLINT
      cb_height_2: SMALLINT
      cb_height_3: SMALLINT
      vertical_visibility: SMALLINT
      highest_signal: SMALLINT
      laser_energy: SMALLINT
      laser_temperature: SMALLINT
      receiver_sensitivity: SMALLINT
      window_contamination: SMALLINT
      tilt_angle: SMALLINT
      background_light: SMALLINT
      measurement_mode: CHARACTER(1)
      measurement_params: CHARACTER(6)
      bs_sum: REAL
      bs: REAL[]
(PRIMARY KEY (time_utc, paramset))
```

```
ceilo_parameters
PK    paramset_id: SERIAL
      name: CHARACTER VARYING(255) = ''
      creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
      location_latitude: REAL
      location_longitude: REAL
      location_elevation: REAL
      location_utc_offset: SMALLINT = '0'
      instrument_description: CHARACTER VARYING(255)
      time_resolution: REAL
```

```
heatsensor
      time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
   FK paramset: INTEGER = '0'
      reliability: INTEGER
      air_temperature: REAL
      heat_melting_snow: REAL
      heat_warming_sensor_board: REAL
      snowparticles: INTEGER
      water_detection: INTEGER
      snow_detection: INTEGER
      snow_accretion_detection: INTEGER
      freezing_detection: INTEGER
(PRIMARY KEY (time_utc, paramset))
```

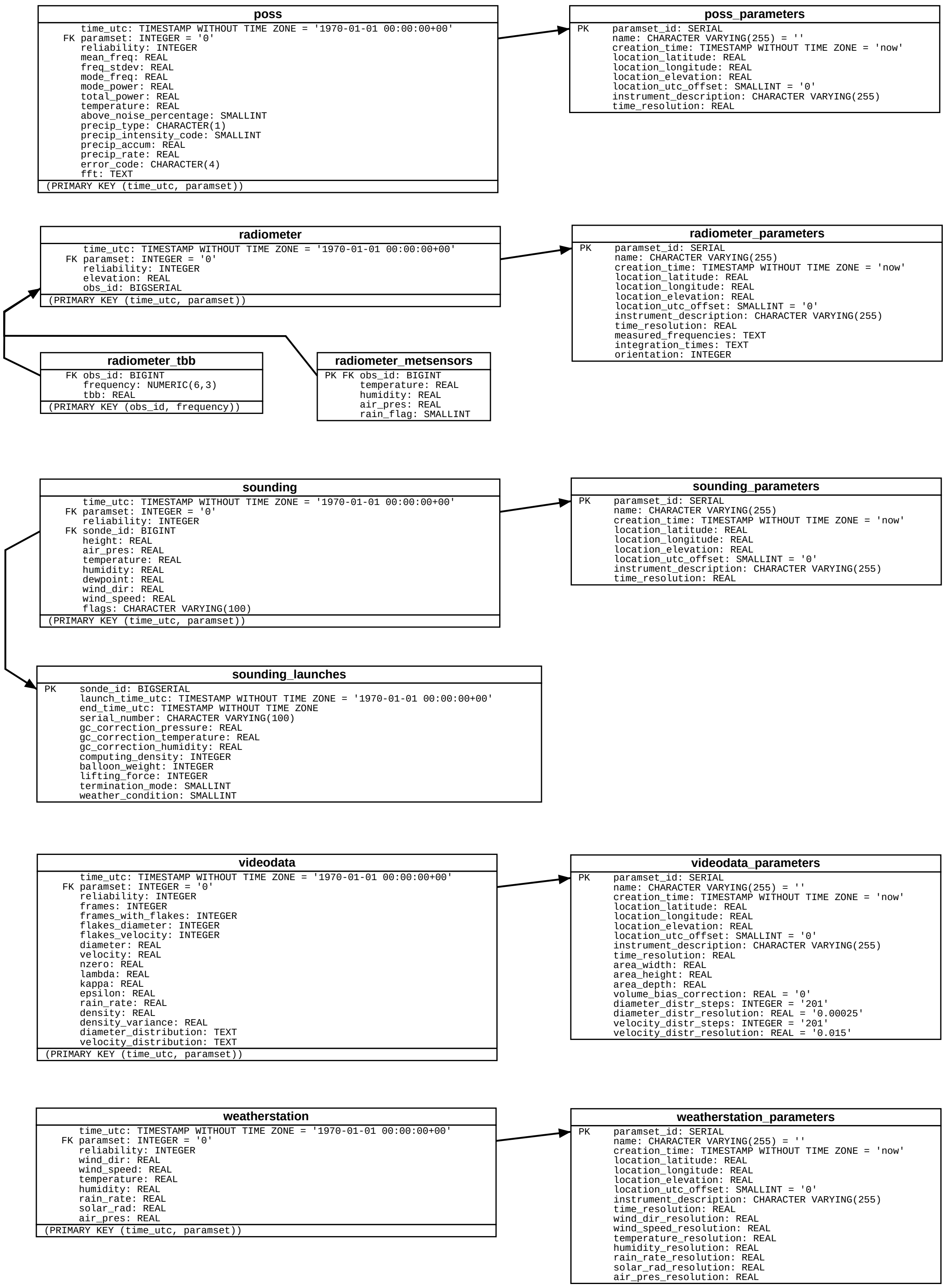
```
heatsensor_parameters
PK    paramset_id: SERIAL
      name: CHARACTER VARYING(255)
      creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
      location_latitude: REAL
      location_longitude: REAL
      location_elevation: REAL
      location_utc_offset: SMALLINT = '0'
      instrument_description: CHARACTER VARYING(255)
      time_resolution: REAL
      heat_resolution: REAL
```

```
mrr
      time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
   FK paramset: INTEGER = '0'
      reliability: INTEGER
      z: REAL[]
      rr: REAL[]
      lwc: REAL[]
      w: REAL[]
      f: TEXT[]
      n: TEXT[]
(PRIMARY KEY (time_utc, paramset))
```

```
mrr_parameters
PK    paramset_id: SERIAL
      name: CHARACTER VARYING(255) = ''
      creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
      location_latitude: REAL
      location_longitude: REAL
      location_elevation: REAL
      location_utc_offset: SMALLINT = '0'
      instrument_description: CHARACTER VARYING(255)
      time_resolution: REAL
      height_resolution: INTEGER
      height_steps: INTEGER = '30'
      calibration_spectra: BYTEA
```

```
mrr_raw
      time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
   FK paramset: INTEGER = '0'
      reliability: INTEGER
      f: TEXT[]
(PRIMARY KEY (time_utc, paramset))
```

Figure 22: Wakasa table layout, array version, page 1(2)



```
poss
      time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
   FK paramset: INTEGER = '0'
      reliability: INTEGER
      mean_freq: REAL
      freq_stdev: REAL
      mode_freq: REAL
      mode_power: REAL
      total_power: REAL
      temperature: REAL
      above_noise_percentage: SMALLINT
      precip_type: CHARACTER(1)
      precip_intensity_code: SMALLINT
      precip_accum: REAL
      precip_rate: REAL
      error_code: CHARACTER(4)
      fft: TEXT
(PRIMARY KEY (time_utc, paramset))
```

```
poss_parameters
PK    paramset_id: SERIAL
      name: CHARACTER VARYING(255) = ''
      creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
      location_latitude: REAL
      location_longitude: REAL
      location_elevation: REAL
      location_utc_offset: SMALLINT = '0'
      instrument_description: CHARACTER VARYING(255)
      time_resolution: REAL
```

```
radiometer
      time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
   FK paramset: INTEGER = '0'
      reliability: INTEGER
      elevation: REAL
      obs_id: BIGSERIAL
(PRIMARY KEY (time_utc, paramset))
```

```
radiometer_parameters
PK    paramset_id: SERIAL
      name: CHARACTER VARYING(255)
      creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
      location_latitude: REAL
      location_longitude: REAL
      location_elevation: REAL
      location_utc_offset: SMALLINT = '0'
      instrument_description: CHARACTER VARYING(255)
      time_resolution: REAL
      measured_frequencies: TEXT
      integration_times: TEXT
      orientation: INTEGER
```

```
radiometer_tbb
   FK obs_id: BIGINT
      frequency: NUMERIC(6,3)
      tbb: REAL
(PRIMARY KEY (obs_id, frequency))
```

```
radiometer_metsensors
PK FK obs_id: BIGINT
      temperature: REAL
      humidity: REAL
      air_pres: REAL
      rain_flag: SMALLINT
```

```
sounding
      time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
   FK paramset: INTEGER = '0'
      reliability: INTEGER
   FK sonde_id: BIGINT
      height: REAL
      air_pres: REAL
      temperature: REAL
      humidity: REAL
      dewpoint: REAL
      wind_dir: REAL
      wind_speed: REAL
      flags: CHARACTER VARYING(100)
(PRIMARY KEY (time_utc, paramset))
```

```
sounding_parameters
PK    paramset_id: SERIAL
      name: CHARACTER VARYING(255)
      creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
      location_latitude: REAL
      location_longitude: REAL
      location_elevation: REAL
      location_utc_offset: SMALLINT = '0'
      instrument_description: CHARACTER VARYING(255)
      time_resolution: REAL
```

```
sounding_launches
PK    sonde_id: BIGSERIAL
      launch_time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
      end_time_utc: TIMESTAMP WITHOUT TIME ZONE
      serial_number: CHARACTER VARYING(100)
      gc_correction_pressure: REAL
      gc_correction_temperature: REAL
      gc_correction_humidity: REAL
      computing_density: INTEGER
      balloon_weight: INTEGER
      lifting_force: INTEGER
      termination_mode: SMALLINT
      weather_condition: SMALLINT
```

```
videodata
      time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
   FK paramset: INTEGER = '0'
      reliability: INTEGER
      frames: INTEGER
      frames_with_flakes: INTEGER
      flakes_diameter: INTEGER
      flakes_velocity: INTEGER
      diameter: REAL
      velocity: REAL
      nzero: REAL
      lambda: REAL
      kappa: REAL
      epsilon: REAL
      rain_rate: REAL
      density: REAL
      density_variance: REAL
      diameter_distribution: TEXT
      velocity_distribution: TEXT
(PRIMARY KEY (time_utc, paramset))
```

```
videodata_parameters
PK    paramset_id: SERIAL
      name: CHARACTER VARYING(255) = ''
      creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
      location_latitude: REAL
      location_longitude: REAL
      location_elevation: REAL
      location_utc_offset: SMALLINT = '0'
      instrument_description: CHARACTER VARYING(255)
      time_resolution: REAL
      area_width: REAL
      area_height: REAL
      area_depth: REAL
      volume_bias_correction: REAL = '0'
      diameter_distr_steps: INTEGER = '201'
      diameter_distr_resolution: REAL = '0.00025'
      velocity_distr_steps: INTEGER = '201'
      velocity_distr_resolution: REAL = '0.015'
```

```
weatherstation
      time_utc: TIMESTAMP WITHOUT TIME ZONE = '1970-01-01 00:00:00+00'
   FK paramset: INTEGER = '0'
      reliability: INTEGER
      wind_dir: REAL
      wind_speed: REAL
      temperature: REAL
      humidity: REAL
      rain_rate: REAL
      solar_rad: REAL
      air_pres: REAL
(PRIMARY KEY (time_utc, paramset))
```

```
weatherstation_parameters
PK    paramset_id: SERIAL
      name: CHARACTER VARYING(255) = ''
      creation_time: TIMESTAMP WITHOUT TIME ZONE = 'now'
      location_latitude: REAL
      location_longitude: REAL
      location_elevation: REAL
      location_utc_offset: SMALLINT = '0'
      instrument_description: CHARACTER VARYING(255)
      time_resolution: REAL
      wind_dir_resolution: REAL
      wind_speed_resolution: REAL
      temperature_resolution: REAL
      humidity_resolution: REAL
      rain_rate_resolution: REAL
      solar_rad_resolution: REAL
      air_pres_resolution: REAL
```

Figure 23: Wakasa table layout, array version, page 2(2)

B Table column reference

Data types and descriptions of all table columns are listed in this section. It has been automatically generated from the table comments in the database.

B.1 Standard version

B.1.1 balance

Table to record measurements done using electric balances, should be suitable for different manufacturers and models. NULL in each field means that the value was not available, due to the feature not present in the model or an error or in the measurement.

Table 13: Table `balance` column descriptions

Column name	**Type**	**Description**
time_utc	timestamp without time zone	*NOT NULL default '1970-01-01 00:00:00+00'* Timestamp (UTC) of the recorded observation. See parameters table for timezone if you need local time.
paramset	integer	*NOT NULL default '0'* Parameters describing the balance model and settings used when recording the stored observation. Refers to table balance_parameters.
reliability	integer	Estimated reliability of the measurement (NULL = unknown).
rain_rate	real	Rainfall rate in mm/h (for snowfall, value converted to be equivalent with rain rate).
weight	real	The current weight in grams.
stable_num	integer	Number of stable messages received from the balance during the integration period
stable_accum	real	Accumulated weight in grams, based on stable messages during the integration period
unstable_num	integer	Number of unstable messages received from the balance during the integration period
unstable_accum	real	Accumulated weight in grams, based on unstable messages during the integration period

Table 14: Table `balance` constraints

Name	Constraint
balance_pkey	PRIMARY KEY (time_utc, paramset)

B.1.2 balance_parameters

Parameters for electric balance devices in context of snowfall rate measurement, should be suitable for different manufacturers and models. In any column, NULL means that the value is unknown.

Table 15: Table `balance_parameters` column descriptions

Column name	Type	Description
paramset_id	serial	*NOT NULL PRIMARY KEY* Unique identifier for the set (used in table balance).
name	character varying(255)	*NOT NULL default ''* Name of the set (chosen by the user).
creation_time	timestamp without time zone	*default ('now'::text)::timestamp(6) with time zone* Time when the parameter set was created.
location_latitude	real	Latitude coordinate (degrees) of the device during measurements.
location_longitude	real	Longitude coordinate (degrees) of the device during measurements.
location_elevation	real	Ground elevation of the device in meters above sea level
location_utc_offset	smallint	*NOT NULL default '0'* UTC time offset of the location of the device during measurements (e.g. +09 in Japan).
instrument_description	character varying(255)	Manufacturer, model or anything that can be used to identify the instrument used.
time_resolution	real	Time resolution in seconds.
box_area	real	*NOT NULL* Area on which the weight is measured in mˆ2 (the area of a box placed on the balance for rainfall/snowfall measurement)
weight_resolution	real	Weight resolution in grams

Tables referencing this one via foreign key constraints:

- balance

B.1.3 ceilo

Table to record measurements done using Vaisala CT-25K Ceilometers. Probably not suitable for other models but measurements from several separate CT-25K devices can be fed in the same table (and distinguished by using separate parameter sets). NULL in each field means that the value was not available, due to state (e.g. three cloud base values are provided only if three cloud bases are detected) or due to an error or in the measurement.

Table 16: Table `ceilo` column descriptions

Column name	Type	Description
time_utc	timestamp without time zone	*NOT NULL default '1970-01-01 00:00:00+00'* Timestamp (UTC) of the recorded observation. See parameters table for timezone if you need local time.
paramset	integer	*NOT NULL default '0'* Parameters describing the settings used when recording the stored observation. Refers to table ceilo_parameters.
reliability	integer	Estimated reliability of the measurement (NULL = unknown).
obs_id	bigserial	*NOT NULL UNIQUE* Unique ID number for the observation.
detection_status	smallint	Status of detection, range 0-5. 0 = no significant backscatter, 1 = one cloud base, 2 = two cloud bases, 3 = three cloud bases, 4 = full obscuration but no cloud base, 5 = some obscuration but determined to be transparent
alarm_status	smallint	Status of alarms, range 0-2. 0 = Self-check OK (no warnings or alarms), 1 = at least one warning active, 2 = at least one alarm active
alarm_code	character(8)	Alarm, warning and internal status information. Refer to device manual page 32.
cloudbases	smallint	Number of cloud bases detected, range 0-3
cb_height_1	smallint	Height of first cloud base in meters.
cb_height_2	smallint	Height of second cloud base in meters.
cb_height_3	smallint	Height of third cloud base in meters.
vertical_visibility	smallint	Vertical visibility in meters, calculated (only available with detection status 4).
highest_signal	smallint	Height of highest signal detected in meters (only available with detection status 4).
laser_energy	smallint	Laser pulse energy, percents of nominal factory setting (range 0-999).
laser_temperature	smallint	Laser temperature in degrees Celsius (range -50...+99).

(Continued on next page)

(Continued from previous page)

Column name	Type	Description
receiver_sensitivity	smallint	Receiver sensitivity, percents of nominal factory setting (range 0-999).
window_contamination	smallint	Window contamination, millivolts at internal ADC input (range 0-2500).
tilt_angle	smallint	Tilt angle, degrees from vertical (range -15...90).
background_light	smallint	Background light, millivolts at internal ADC input (range 0-2500).
measurement_mode	character(1)	Measurement mode, N = normal, C = close range (page 35 in the device manual).
measurement_params	character(6)	Measurement parameters (pulse length, gain, bandwidth etc.) encoded in 6 letters. See page 35 in the device manual.
bs_sum	real	Sum of detected and normalized backscatter, in 1/srad. The value is already scaled appropriately using the SCALE parameter (reported by the device)

Table 17: Table `ceilo` constraints

Name	Constraint
ceilo_pkey	PRIMARY KEY (time_utc, paramset)

Tables referencing this one via foreign key constraints:

- ceilo_backscatter

B.1.4 ceilo_backscatter

Table to store the vertical backscatter profile in Vaisala CT-25K Ceilometer measurements.

Table 18: Table `ceilo_backscatter` column descriptions

Column name	Type	Description
obs_id	bigint	*NOT NULL* Unique ID number for the observation, refers to table ceilo.

(Continued on next page)

(Continued from previous page)

Column name	Type	Description
height	smallint	*NOT NULL* Height (upper limit) of the measurement, in meters.
bs	real	Backscatter value, sensitivity and range normalized, in units of 1/(srad*km). Already scaled appropriately using the SCALE parameter. 30 meter height resolution. NULL means that the value was not available, probably due to an error in the measurement.

Table 19: Table `ceilo_backscatter` constraints

Name	Constraint
ceilo_backscatter_pkey	PRIMARY KEY (obs_id, height)

B.1.5 ceilo_parameters

Parameters for Vaisala Ceilometer CT-25K devices. In any column, NULL means that the value is unknown.

Table 20: Table `ceilo_parameters` column descriptions

Column name	Type	Description
paramset_id	serial	*NOT NULL PRIMARY KEY* Unique identifier for the set (used in table ceilo).
name	character varying(255)	*NOT NULL default ”* Name of the set (chosen by the user).
creation_time	timestamp without time zone	*default (’now’::text)::timestamp(6) with time zone* Time when the parameter set was created.
location_latitude	real	Latitude coordinate (degrees) of the device during measurements.
location_longitude	real	Longitude coordinate (degrees) of the device during measurements.
location_elevation	real	Ground elevation of the device in meters above sea level

(Continued on next page)

(Continued from previous page)

Column name	Type	Description
location_utc_offset	smallint	*NOT NULL default '0'* UTC time offset of the location of the device during measurements (e.g. +09 in Japan).
instrument_description	character varying(255)	Manufacturer, model or anything that can be used to identify the instrument used.
time_resolution	real	Time resolution in seconds.

Tables referencing this one via foreign key constraints:

- ceilo

B.1.6 heatsensor

Table to record measurements done using snow heat capacity measuring devices. NULL in each field means that the value was not available.

Table 21: Table `heatsensor` column descriptions

Column name	Type	Description
time_utc	timestamp without time zone	*NOT NULL default '1970-01-01 00:00:00+00'* Timestamp (UTC) of the recorded observation. See parameters table for timezone if you need local time.
paramset	integer	*NOT NULL default '0'* Parameters describing the heat capacity sensor model (probably only one possibility), location and settings used when recording the stored observation. Refers to table heatsensor_parameters.
reliability	integer	Estimated reliability of the measurement (NULL = unknown).
air_temperature	real	Temperature of air in degrees Celsius.
heat_melting_snow	real	Amount of heat needed to melt snow on the sensor board [Kcal/min/mˆ2]
heat_warming_sensor_board	real	Amount of heat needed to keep the sensor board from freezing [Kcal/min/mˆ2]
snowparticles	integer	Number of snow particles detected during this measurement
water_detection	integer	Detection signal for water on the sensor, can be also caused by snowfall (?). NULL = unknown, 0 = No water detected, 1 = Water detected.

(Continued on next page)

(Continued from previous page)

Column name	Type	Description
snow_detection	integer	Snowfall detection signal. NULL = unknown, 0 = No snowfall detected, 1 = Snowfall detected.
snow_accretion_detection	integer	Snow accretion detection signal. NULL = unknown, 0 = No snow accretion detected, 1 = Snow accretion detected.
freezing_detection	integer	Freezing detection signal. NULL = unknown, 0 = No freezing detected, 1 = Freezing detected.

Table 22: Table `heatsensor` constraints

Name	Constraint
heatsensor_pkey	PRIMARY KEY (time_utc, paramset)

B.1.7 heatsensor_parameters

Parameters for snow heat capacity sensors manufactured by Yamada Giken. In any column, NULL means that the value is unknown.

Table 23: Table `heatsensor_parameters` column descriptions

Column name	Type	Description
paramset_id	serial	*NOT NULL PRIMARY KEY* Unique identifier for the set (used in table heatsensor).
name	character varying(255)	*NOT NULL* Name of the set (chosen by the user).
creation_time	timestamp without time zone	*default ('now'::text)::timestamp(6) with time zone* Time when the parameter set was created.
location_latitude	real	Latitude coordinate (degrees) of the device during measurements
location_longitude	real	Longitude coordinate (degrees) of the device during measurements
location_elevation	real	Ground elevation of the device in meters above sea level
location_utc_offset	smallint	*NOT NULL default '0'* UTC time offset of the location of the device during measurements (e.g. +09 in Japan).

(Continued on next page)

(Continued from previous page)

Column name	Type	Description
instrument_description	character varying(255)	*default 'Yamada Giken Snow heat capacity measuring device'* Manufacturer, model or anything that can be used to identify the instrument used.
time_resolution	real	Time resolution in seconds.
heat_resolution	real	Heat resolution in Kcal/min/mˆ2

Tables referencing this one via foreign key constraints:

- heatsensor

B.1.8 mrr

Main table to index measurements done using Metek MRR devices. Actually contains no measurement results, but acts instead as an index for the data table mrr_data, storing the time stamp and information about used parameter set for each observation id.

Table 24: Table `mrr` column descriptions

Column name	Type	Description
time_utc	timestamp without time zone	*NOT NULL default '1970-01-01 00:00:00+00'* Timestamp (UTC) of the recorded observation. See parameters table for timezone if you need local time.
paramset	integer	*NOT NULL default '0'* Parameters describing the settings used when recording the stored observation. Refers to table mrr_parameters.
reliability	integer	Estimated reliability of the measurement (NULL = unknown).
obs_id	bigserial	*NOT NULL UNIQUE* Unique ID number for the observation.

Table 25: Table `mrr` constraints

Name	Constraint
mrr_pkey	PRIMARY KEY (time_utc, paramset)

Tables referencing this one via foreign key constraints:

- mrr_data

B.1.9 mrr_data

Table to store data produced by MRR, indexed by height and observation id. For each row, refer to table mrr_parameters (via table mrr and obs_id) for the height resolution currently in use.

Table 26: Table `mrr_data` column descriptions

Column name	Type	Description
obs_id	bigint	*NOT NULL* Unique ID number for the observation, refers to table mrr.
height	smallint	*NOT NULL* Height (upper limit) of the measurement, in meters.
z	real	The reflectivity value at the given height, in dBZ. If NULL, means that the device did not provide a value for that height.
rr	real	The calculated rain rate value at the given height, in mm/h. If NULL, means that the device did not provide a value for that height.
lwc	real	The calculated liquid water contents value at the given height, in g/mˆ3. If NULL, means that the device did not provide a value for that height.
w	real	The calculated characteristic fall velocity of precipitation particles (assuming rain) at the given height, in m/s. If NULL, means that the device did not provide a value for that height.
f	text	The FFT spectra (backscattered power values) at the given height, in dBn. 64 floating point values (F00-F63), stored as text in csv format, values separated by commas. Two successive commas without a value in between means that the device did not provide a value for that row. Similarly, if the row ends in a comma the last value (F63) was missing.

(Continued on next page)

(Continued from previous page)

Column name	Type	Description
n	text	The calculated drop size distribution (assuming rain) for the given height, in 1/(m^3*mm^1). 43 floating point values (N04-46), stored as text in csv format, values separated by commas. Two successive commas without a value in between means that the device did not provide a value for that row. Similarly, if the row ends in a comma the last value (N46) was missing.

Table 27: Table `mrr_data` constraints

Name	Constraint
mrr_data_pkey	PRIMARY KEY (obs_id, height)

B.1.10 mrr_parameters

Parameters for Metek Micro Rain Radar devices. This should be suitable for both Metek MRR-1 and Metek MRR-2, but probably not for other makers and models.

Table 28: Table `mrr_parameters` column descriptions

Column name	Type	Description
paramset_id	serial	*NOT NULL PRIMARY KEY* Unique identifier for the set (used in table mrr).
name	character varying(255)	*NOT NULL default ”* Name of the set (chosen by the user).
creation_time	timestamp without time zone	*default ('now'::text)::timestamp(6) with time zone* Time when the parameter set was created.
location_latitude	real	Latitude coordinate (degrees) of the device during measurements.
location_longitude	real	Longitude coordinate (degrees) of the device during measurements.
location_elevation	real	Ground elevation of the device in meters above sea level.
location_utc_offset	smallint	*NOT NULL default '0'* UTC time offset of the location of the device during measurements (e.g. +09 in Japan).

(Continued on next page)

(Continued from previous page)

Column name	Type	Description
instrument_description	character varying(255)	*default ’Metek MRR-2 Micro Rain Radar’* Manufacturer, model or anything that can be used to identify the instrument used.
time_resolution	real	Time resolution in seconds.
height_resolution	integer	*NOT NULL* Height resolution in meters.
height_steps	integer	*NOT NULL default ’30’* The number of height steps. 6 for MRR-1, 30 for MRR-2.
calibration_spectra	bytea	Calibration spectra as stored by the MRR software in the Windows registry.

Tables referencing this one via foreign key constraints:

- mrr
- mrr_raw

B.1.11 mrr_raw

Index to observations of the raw unprocessed FFT data produced by MRR.

Table 29: Table `mrr_raw` column descriptions

Column name	Type	Description
time_utc	timestamp without time zone	*NOT NULL default ’1970-01-01 00:00:00+00’* Timestamp (UTC) of the recorded observation. See parameters table for timezone if you need local time.
paramset	integer	*NOT NULL default ’0’* Parameters describing the settings used when recording the stored observation. Refers to table mrr_parameters.
reliability	integer	Estimated reliability of the measurement (NULL = unknown).
obs_id	bigserial	*NOT NULL UNIQUE* Unique ID number for the observation.

Table 30: Table `mrr_raw` constraints

Name	Constraint
mrr_raw_pkey	PRIMARY KEY (time_utc, paramset)

Tables referencing this one via foreign key constraints:

- mrr_raw_data

B.1.12 mrr_raw_data

Table to store unprocessed FFT data produced by MRR, indexed by height and observation id. For each row, refer to table mrr_parameters (via table mrr and obs_id) for the height resolution currently in use.

Table 31: Table `mrr_raw_data` column descriptions

Column name	Type	Description
obs_id	bigint	*NOT NULL* Unique ID number for the observation, refers to table mrr_raw.
height	smallint	*NOT NULL* Height of the measurement, in meters. Note that there are two extra height steps in raw data compared to processed data (at least in MRR-2): one at zero and one above the maximum.
f	text	The received signal at the given height, unit unknown. 64 floating point values (F00-F63), stored as text in csv format, values separated by commas. Two successive commas without a value in between means that the device did not provide a value for that row. Similarly, if the row ends in a comma the last value (F63) was missing.

Table 32: Table `mrr_raw_data` constraints

Name	Constraint
mrr_raw_data_pkey	PRIMARY KEY (obs_id, height)

B.1.13 poss

Table to record measurements done using Andrew POSS devices. Probably not suitable for other models but measurements from several separate POSS devices can be fed in the same table (and distinguished by using separate parameter sets). NULL in each field means that the value was not available, due to state or an error or in the measurement.

Table 33: Table `poss` column descriptions

Column name	Type	Description
time_utc	timestamp without time zone	*NOT NULL default '1970-01-01 00:00:00+00'* Timestamp (UTC) of the recorded observation. See parameters table for timezone if you need local time.
paramset	integer	*NOT NULL default '0'* Parameters describing the settings used when recording the stored observation. Refers to table poss_parameters.
reliability	integer	Estimated reliability of the measurement (NULL = unknown).
mean_freq	real	Mean frequency first moment in Hz
freq_stdev	real	Standard deviation of frequency 2nd moment in Hz.
mode_freq	real	Mode frequency in Hz.
mode_power	real	Mode power in linear units.
total_power	real	Total power (zeroth moment) in linear units.
temperature	real	Temperature in degrees celsius
above_noise_percentage	smallint	Percentage of spectra in the one minute average with total power exceeding the "adaptive noise threshold"
precip_type	character(1)	Precipitation type: L = Drizzle, R = Rain, S = Snow, A = Hail, P = Precipitation (undefined), N = No precipitation.
precip_intensity_code	smallint	Precipitation intensity code: 0 = No precipitation detected, 1 = Very light, 2 = Light, 3 = Moderate, 4 = Heavy. For more details see POSS manual, page 20.
precip_accum	real	Accumulated precipitation amount in mm.
precip_rate	real	Precipitation rate in mm/h.
error_code	character(4)	Error code as described in POSS manual page 21. Value 0000 means no error.

(Continued on next page)

(Continued from previous page)

Column name	Type	Description
fft	text	The values of FFT power spectra, stored as comma separated text string. 64 entries from 0 to 1024 Hz, in 16 Hz increments. (Note: This is copied from POSS manual, but actually if the first value is at 0 Hz and last at 1024 Hz that would yield 65 values in 16 Hz steps!). If the value is NULL, the whole FFT spectra was missing from the output (not produced by the device if there is no precipitation).

Table 34: Table `poss` constraints

Name	Constraint
poss_pkey	PRIMARY KEY (time_utc, paramset)

B.1.14 poss_parameters

Parameters for Andrew POSS (Precipitation Occurrence Sensor System) devices. In any column, NULL means that the value is unknown.

Table 35: Table `poss_parameters` column descriptions

Column name	Type	Description
paramset_id	serial	*NOT NULL PRIMARY KEY* Unique identifier for the set (used in table poss).
name	character varying(255)	*NOT NULL default ”* Name of the set (chosen by the user).
creation_time	timestamp without time zone	*default ('now'::text)::timestamp(6) with time zone* Time when the parameter set was created.
location_latitude	real	Latitude coordinate (degrees) of the device during measurements.
location_longitude	real	Longitude coordinate (degrees) of the device during measurements.
location_elevation	real	Ground elevation of the device in meters above sea level
location_utc_offset	smallint	*NOT NULL default '0'* UTC time offset of the location of the device during measurements (e.g. +09 in Japan).

(Continued on next page)

(Continued from previous page)

Column name	Type	Description
instrument_description	character varying(255)	*default 'Andrew POSS'* Manufacturer, model or anything that can be used to identify the instrument used.
time_resolution	real	Time resolution in seconds.

Tables referencing this one via foreign key constraints:

- poss

B.1.15 radiometer

Table to record measurements done by radiometers, should be suitable for different manufacturers and models. NULL in each field means that the value was not available.

Table 36: Table `radiometer` column descriptions

Column name	Type	Description
time_utc	timestamp without time zone	*NOT NULL default '1970-01-01 00:00:00+00'* Timestamp (UTC) of the recorded observation. See parameters table for timezone if you need local time.
paramset	integer	*NOT NULL default '0'* Parameters describing the radiometer model and settings used when recording the stored observation. Refers to table radiometer_parameters.
reliability	integer	Estimated reliability of the measurement (NULL = unknown).
elevation	real	Elevation angle in degrees at which the measurement was taken (0=looking horizontal, 90=looking at zenith/vertical)
obs_id	bigserial	*NOT NULL UNIQUE* Unique ID number for the observation.

Table 37: Table `radiometer` constraints

Name	Constraint
radiometer_pkey	PRIMARY KEY (time_utc, paramset)

Tables referencing this one via foreign key constraints:

- radiometer_metsensors
- radiometer_tbb

B.1.16 radiometer_metsensors

Table to store data from meteorological sensors attached to (some) radiometers.

Table 38: Table `radiometer_metsensors` column descriptions

Column name	Type	Description
obs_id	bigint	*NOT NULL PRIMARY KEY* Unique ID number for the observation, refers to table radiometer.
temperature	real	Temperature in degrees Celsius
humidity	real	Humidity in percents of relative humidity (0-100)
air_pres	real	Air pressure in hPA
rain_flag	smallint	Rain Flag. Indicates that the measurement was taken under raining conditions. 0 = no rain, 1 = rain

B.1.17 radiometer_parameters

Parameters for radiometers, should be suitable for different manufacturers and models. In any column, NULL means that the value is unknown.

Table 39: Table `radiometer_parameters` column descriptions

Column name	Type	Description
paramset_id	serial	*NOT NULL PRIMARY KEY* Unique identifier for the set (used in table radiometer).
name	character varying(255)	*NOT NULL* Name of the set (chosen by the user).
creation_time	timestamp without time zone	*default ('now'::text)::timestamp(6) with time zone* Time when the parameter set was created.

(Continued on next page)

(Continued from previous page)

Column name	Type	Description
location_latitude	real	Latitude coordinate (degrees) of the device during measurements
location_longitude	real	Longitude coordinate (degrees) of the device during measurements.
location_elevation	real	Ground elevation of the device in meters above sea level
location_utc_offset	smallint	*NOT NULL default '0'* UTC time offset of the location of the device during measurements (e.g. +09 in Japan).
instrument_description	character varying(255)	Manufacturer, model or anything that can be used to identify the instrument used.
time_resolution	real	Time resolution in seconds.
measured_frequencies	text	List of frequencies measured, stored as a string containing comma separated values.
integration_times	text	Integration time for each measured frequency, in the same order as the frequencies in the measured_frequencies field. Stored as a string containing comma separated values.
orientation	integer	Azimuth angle of the viewing direction of the radiometer, given as integer value from 0 to 360 degrees clockwise (0=north, 90=east) at a mirror elevation of 0 degrees.

Tables referencing this one via foreign key constraints:

- radiometer

B.1.18 radiometer_tbb

Table to store the brightness temperatures measured by radiometers.

Table 40: Table `radiometer_tbb` column descriptions

Column name	Type	Description
obs_id	bigint	*NOT NULL* Unique ID number for the observation, refers to table radiometer.
frequency	numeric(6,3)	*NOT NULL* Measurement frequency in GHz, 3 decimal points. Maximum value 999.999 GHz.
tbb	real	The measured brightness temperature value.

Table 41: Table `radiometer_tbb` constraints

Name	Constraint
radiometer_tbb_pkey	PRIMARY KEY (obs_id, frequency)

B.1.19 sounding

Table to record values obtained during atmospheric soundings, should be suitable for different locations and sounding types. NULL in each field means that the value was not available.

Table 42: Table `sounding` column descriptions

Column name	Type	Description
time_utc	timestamp without time zone	*NOT NULL default '1970-01-01 00:00:00+00'* Timestamp (UTC) of the recorded observation. See parameters table for timezone if you need local time.
paramset	integer	*NOT NULL default '0'* Parameters describing the type of sounding and settings used when recording the stored observation. Refers to table sounding_parameters.
reliability	integer	Estimated reliability of the measurement (NULL = unknown).
sonde_id	bigint	*NOT NULL* Id number of the sonde which was used for this measurement, refers to sonde_id in table sounding_launches
height	real	Height of the respective measurement [m].
air_pres	real	Air pressure at the corresponding height level [hPa].
temperature	real	Temperature at the corresponding height level [deg C].
humidity	real	Relative humidity at the corresponding height level [%].
dewpoint	real	Dewpoint temperature at the corresponding height level [deg C].
wind_dir	real	Wind direction in degrees counted clockwise (0=from north, 90=from east) at the corresponding height level.

(Continued on next page)

(Continued from previous page)

Column name	Type	Description
wind_speed	real	Wind speed at the corresponding height level [m/s].
flags	character varying(100)	Measurement flags.

Table 43: Table `sounding` constraints

Name	Constraint
sounding_pkey	PRIMARY KEY (time_utc, paramset)

B.1.20 sounding_launches

Table 44: Table `sounding_launches` column descriptions

Column name	Type	Description
sonde_id	bigserial	*NOT NULL PRIMARY KEY* Unique ID number for the sonde.
launch_time_utc	timestamp without time zone	*NOT NULL default '1970-01-01 00:00:00+00'* Launch time for the sounding.
end_time_utc	timestamp without time zone	End time for the sounding.
serial_number	character varying(100)	Serial number of the sonde.
gc_correction_pressure	real	Ground calibration pressure correction [hPa]
gc_correction_temperature	real	Ground calibration temperature correction [deg C]
gc_correction_humidity	real	Ground calibration humidity correction [% rel humidity]
computing_density	integer	Computing density [s]
balloon_weight	integer	Weight of the balloon [g]
lifting_force	integer	Buoyancy of the filled balloon at ground level [g]
termination_mode	smallint	Termination mode of the sounding. Integer encoded. 0=auto(burst), 1=manual(vanishing signal).

(Continued on next page)

(Continued from previous page)

Column name	Type	Description
weather_condition	smallint	Weather condition at launch time given as SYNOP-ww-code [see http://www.brixworth.demon.co.uk/weather19.htm (as of 2003-05-22) for detailed description]

Tables referencing this one via foreign key constraints:

- sounding

B.1.21 sounding_parameters

Parameters for soundings, should be suitable for different locations and sounding types. In any column, NULL means that the value is unknown.

Table 45: Table `sounding_parameters` column descriptions

Column name	Type	Description
paramset_id	serial	*NOT NULL PRIMARY KEY* Unique identifier for the set (used in table sounding).
name	character varying(255)	*NOT NULL* Name of the set (chosen by the user).
creation_time	timestamp without time zone	*default ('now'::text)::timestamp(6) with time zone* Time when the parameter set was created.
location_latitude	real	Latitude coordinate (degrees) of the launch site of the soundings.
location_longitude	real	Longitude coordinate (degrees) of the launch site of the soundings.
location_elevation	real	Ground elevation of the launch site of the soundings, in meters above sea level
location_utc_offset	smallint	*NOT NULL default '0'* UTC time offset of the launch location (e.g. +09 in Japan).
instrument_description	character varying(255)	Manufacturer, model or anything that can be used to identify the instrument used.
time_resolution	real	Time resolution in seconds.

Tables referencing this one via foreign key constraints:

- sounding

B.1.22 videodata

Main table to store measurements done using the video camera based snowflake observation system developed in Image Information Science lab, Kanazawa university.

Table 46: Table `videodata` column descriptions

Column name	Type	Description
time_utc	timestamp without time zone	*NOT NULL default '1970-01-01 00:00:00+00'* Timestamp (UTC) of the recorded observation. See parameters table for timezone if you need local time.
paramset	integer	*NOT NULL default '0'* Parameters describing the settings used when recording the stored observation. Refers to table videodata_parameters.
reliability	integer	Estimated reliability of the measurement (NULL = unknown).
frames	integer	Number of frames during the averaging period.
frames_with_flakes	integer	Number of frames containing at least one snowflake during the averaging period.
flakes_diameter	integer	Number of snowflakes used for calculating the diameter distribution values. This can be also used as the total number of flakes during the averaging period.
flakes_velocity	integer	Number of snowflakes used for calculating the velocity distribution values. Normally same than flakes_diameter.
diameter	real	Average diameter of snowflakes in meters.
velocity	real	Average velocity of snowflakes in m/s.
nzero	real	Parameter N0 in the size distribution equation N(D) = N0*eˆ(-lambda*D).
lambda	real	Parameter lambda in the size distribution equation N(D) = N0*eˆ(-lambda*D).
kappa	real	Parameter kappa in the velocity distribution equation v(D) = kappa*Dˆ(epsilon).
epsilon	real	Parameter epsilon in the velocity distribution equation v(D) = kappa*Dˆ(epsilon).
rain_rate	real	Snowfall rate in mm/h (water equivalent).
density	real	Average density of snowflakes in g/cmˆ3.

(Continued on next page)

(Continued from previous page)

Column name	**Type**	**Description**
density_variance	real	Variance of the snowflake density during the averaging period.
diameter_distribution	text	Number of snowflakes belonging to each diameter category. Total 201 integer values, stored as csv string, values separated by commas. The first value is number of snowflakes of diameter around zero, second value the number of flakes around the first resolution step (by default 0.00025m = 0.25mm, which means that flakes of diameter 0.20 mm and 0.30 mm should be included at this position). Third value is the number of flakes of diameter around the second step (0.50 mm) etc. The last value is everything around and above the last resolution step (default 50 mm). Two successive commas without a value in between means that the device did not provide a value for that field (different from value zero, which is marked as 0).
velocity_distribution	text	Number of snowflakes belonging to each velocity category. Total 201 integer values, stored as csv string, values separated by commas. The first value is number of snowflakes of velocity around zero, second value the number of flakes around the first resolution step (by default 0.015 m/s = 15 mm/s, which means that flakes of velocity 10 mm/s and 20 mm/s should be included at this position). Third value is the number of flakes of diameter around the second step (0.030 m/s) etc. The last value is everything around and above the last resolution step (default 3 m/s). Two successive commas without a value in between means that the device did not provide a value for that field (different from value zero, which is marked as 0).

Table 47: Table `videodata` constraints

Name	**Constraint**
videodata_pkey	PRIMARY KEY (time_utc, paramset)

B.1.23 videodata_parameters

Parameters for the video camera based snowflake observation system developed in Image Information Science lab, Kanazawa university.

Table 48: Table `videodata_parameters` column descriptions

Column name	Type	Description
paramset_id	serial	*NOT NULL PRIMARY KEY* Unique identifier for the set (used in table videodata).
name	character varying(255)	*NOT NULL default ”* Name of the set (chosen by the user).
creation_time	timestamp without time zone	*default ('now'::text)::timestamp(6) with time zone* Time when the parameter set was created.
location_latitude	real	Latitude coordinate (degrees) of the device during measurements.
location_longitude	real	Longitude coordinate (degrees) of the device during measurements.
location_elevation	real	Ground elevation of the device in meters above sea level.
location_utc_offset	smallint	*NOT NULL default '0'* UTC time offset of the location of the device during measurements (e.g. +09 in Japan).
instrument_description	character varying(255)	Manufacturer, model or anything that can be used to identify the instruments used.
time_resolution	real	Time resolution in seconds.
area_width	real	*NOT NULL* Width of the measurement space in meters.
area_height	real	*NOT NULL* Height of the measurement space in meters.
area_depth	real	*NOT NULL* Depth of the measurement space in meters.

(Continued on next page)

(Continued from previous page)

Column name	Type	Description
volume_bias_correction	real	*NOT NULL default ’0’* Correction factor for calculated volume of the observation area, in percents. When counting the total number of flakes, the system discards traces which are only partially shown in the image. To get a more accurate estimate of the total number of flakes in the observation volume, it is possible to give a bias correction. For example, if in average 10% of the flakes are discarded, the volume_bias_correction should be -10. On the other hand, if some flakes are counted several times (for example the same flake in subsequent frames), it can be fixed by giving a positive volume_bias_correction value.
diameter_distr_steps	integer	*NOT NULL default ’201’* Number of steps in the diameter distribution spectra. The first value is for flakes of diameter zero.
diameter_distr_resolution	real	*NOT NULL default ’0.00025’* Resolution (length of a step) of the diameter distribution spectra in meters.
velocity_distr_steps	integer	*NOT NULL default ’201’* Number of steps in the velocity distribution spectra. The first value is for flakes of velocity zero.
velocity_distr_resolution	real	*NOT NULL default ’0.015’* Resolution (length of a step) of the velocity distribution spectra in m/s.

Tables referencing this one via foreign key constraints:

- videodata

B.1.24 weatherstation

Table to record measurements done by weather stations, should be suitable for different manufacturers and models. NULL in each field means that the value was not available, due to the specific sensor not installed or an error or in the measurement.

Table 49: Table `weatherstation` column descriptions

Column name	Type	Description
time_utc	timestamp without time zone	*NOT NULL default '1970-01-01 00:00:00+00'* Timestamp (UTC) of the recorded observation. See parameters table for timezone if you need local time.
paramset	integer	*NOT NULL default '0'* Parameters describing the weatherstation model and settings used when recording the stored observation. Refers to table weatherstation_parameters.
reliability	integer	Estimated reliability of the measurement (NULL = unknown).
wind_dir	real	Wind direction in degrees (0-360)
wind_speed	real	Wind speed in m/s
temperature	real	Temperature in degrees Celsius
humidity	real	Humidity in percents of relative humidity (0-100)
rain_rate	real	Rain rate in mm/h
solar_rad	real	Solar radiation in MJ/mˆ2
air_pres	real	Air pressure in hPA

Table 50: Table `weatherstation` constraints

Name	Constraint
weatherstation_pkey	PRIMARY KEY (time_utc, paramset)

B.1.25 weatherstation_parameters

Parameters for weather station devices, should be suitable for different manufacturers and models. In any column, NULL means that the value is unknown.

Table 51: Table `weatherstation_parameters` column descriptions

Column name	Type	Description
paramset_id	serial	*NOT NULL PRIMARY KEY* Unique identifier for the set (used in table weatherstation).
name	character varying(255)	*NOT NULL default ''* Name of the set (chosen by the user).

(Continued on next page)

(Continued from previous page)

Column name	Type	Description
creation_time	timestamp without time zone	*default ('now'::text)::timestamp(6) with time zone* Time when the parameter set was created.
location_latitude	real	Latitude coordinate (degrees) of the device during measurements
location_longitude	real	Longitude coordinate (degrees) of the device during measurements.
location_elevation	real	Ground elevation of the device in meters above sea level
location_utc_offset	smallint	*NOT NULL default '0'* UTC time offset of the location of the device during measurements (e.g. +09 in Japan).
instrument_description	character varying(255)	Manufacturer, model or anything that can be used to identify the instrument used.
time_resolution	real	Time resolution in seconds.
wind_dir_resolution	real	Wind direction resolution in degrees
wind_speed_resolution	real	Wind speed resolution in m/s
temperature_resolution	real	Temperature resolution in degrees Celsius
humidity_resolution	real	Humidity resolution in percents of relative humidity
rain_rate_resolution	real	Rain rate resolution in mm/h
solar_rad_resolution	real	Solar radiation resolution in MJ/mˆ2
air_pres_resolution	real	Air pressure resolution in hPA

Tables referencing this one via foreign key constraints:

- weatherstation

B.2 Array version

This version uses PostgreSQL arrays in some tables. Most of the tables are identical with the standard version. The only difference is that tables `ceilo_backscatter`, `mrr_data` and `mrr_raw_data` don't exist: their contents has been moved to tables `ceilo`, `mrr` and `mrr_raw`. Therefore only descriptions of these tables are listed.

B.2.1 ceilo

Table to record measurements done using Vaisala CT-25K Ceilometers. Probably not suitable for other models but measurements from several separate CT-25K devices can be fed in the same table (and distinguished by using separate parameter sets). NULL in each field means that the value was not available, due to state (e.g. three cloud base values are provided only if three cloud bases are detected) or due to an error or in the measurement.

Table 52: Table `ceilo` column descriptions

Column name	Type	Description
time_utc	timestamp without time zone	*NOT NULL default '1970-01-01 00:00:00+00'* Timestamp (UTC) of the recorded observation. See parameters table for timezone if you need local time.
paramset	integer	*NOT NULL default '0'* Parameters describing the settings used when recording the stored observation. Refers to table ceilo_parameters.
reliability	integer	Estimated reliability of the measurement (NULL = unknown).
detection_status	smallint	Status of detection, range 0-5. 0 = no significant backscatter, 1 = one cloud base, 2 = two cloud bases, 3 = three cloud bases, 4 = full obscuration but no cloud base, 5 = some obscuration but determined to be transparent
alarm_status	smallint	Status of alarms, range 0-2. 0 = Self-check OK (no warnings or alarms), 1 = at least one warning active, 2 = at least one alarm active
alarm_code	character(8)	Alarm, warning and internal status information. Refer to device manual page 32.
cloudbases	smallint	Number of cloud bases detected, range 0-3
cb_height_1	smallint	Height of first cloud base in meters.
cb_height_2	smallint	Height of second cloud base in meters.
cb_height_3	smallint	Height of third cloud base in meters.
vertical_visibility	smallint	Vertical visibility in meters, calculated (only available with detection status 4).
highest_signal	smallint	Height of highest signal detected in meters (only available with detection status 4).
laser_energy	smallint	Laser pulse energy, percents of nominal factory setting (range 0-999).
laser_temperature	smallint	Laser temperature in degrees Celsius (range -50...+99).
receiver_sensitivity	smallint	Receiver sensitivity, percents of nominal factory setting (range 0-999).
window_contamination	smallint	Window contamination, millivolts at internal ADC input (range 0-2500).
tilt_angle	smallint	Tilt angle, degrees from vertical (range -15...90).
background_light	smallint	Background light, millivolts at internal ADC input (range 0-2500).
measurement_mode	character(1)	Measurement mode, N = normal, C = close range (page 35 in the device manual).

(Continued on next page)

(Continued from previous page)

Column name	Type	Description
measurement_params	character(6)	Measurement parameters (pulse length, gain, bandwidth etc.) encoded in 6 letters. See page 35 in the device manual.
bs_sum	real	Sum of detected and normalized backscatter, in 1/srad. The value is already scaled appropriately using the SCALE parameter (reported by the device)
bs	real[]	Backscatter values, sensitivity and range normalized, in units of 1/(srad*km). Already scaled appropriately using the SCALE parameter. 30 meter height resolution. NULL means that the value was not available, probably due to an error in the measurement.

Table 53: Table `ceilo` constraints

Name	Constraint
ceilo_pkey	PRIMARY KEY (time_utc, paramset)

B.2.2 mrr

Main table to store measurements done using Metek MRR devices. The measurements are stored as arrays, where each element represents one height. Refer to the table mrr_parameters (via paramset id) for the height resolution used in each measurement.

Table 54: Table `mrr` column descriptions

Column name	Type	Description
time_utc	timestamp without time zone	*NOT NULL default '1970-01-01 00:00:00+00'* Timestamp (UTC) of the recorded observation. See parameters table for timezone if you need local time.
paramset	integer	*NOT NULL default '0'* Parameters describing the settings used when recording the stored observation. Refers to table mrr_parameters.

(Continued on next page)

(Continued from previous page)

Column name	**Type**	**Description**
reliability	integer	Estimated reliability of the measurement (NULL = unknown).
z	real[]	The reflectivity height spectra, in dBZ. If any of the array elements is NULL, means that the device did not provide a value for that height.
rr	real[]	The calculated rain rate height spectra, in mm/h. If any of the array elements is, NULL, means that the device did not provide a value for that height.
lwc	real[]	The calculated liquid water contents value height spectra, in g/mˆ3. If any of the array elements is, NULL, means that the device did not provide a value for that height.
w	real[]	The calculated characteristic fall velocity of precipitation particles (assuming rain) height spectra, in m/s. If any of the array elements is, NULL, means that the device did not provide a value for that height.
f	text[]	The FFT spectra (backscattered power values) for each height step, in dBn. If any of the array elements is NULL, means that the spectra was not obtained for that height. Each element contains 64 floating point values (F00-F63), stored as text in csv format, values separated by commas. Two successive commas without a value in between means that the device did not provide a value for that row. Similarly, if the row ends in a comma the last value (F63) was missing.
n	text[]	The calculated drop size distribution (assuming rain) for each height step, in 1/(mˆ3*mmˆ1). If any of the array elements is NULL, means that the spectra was not obtained for that height. Each element contains 43 floating point values (N04-46), stored as text in csv format, values separated by commas. Two successive commas without a value in between means that the device did not provide a value for that row. Similarly, if the row ends in a comma the last value (N46) was missing.

Table 55: Table mrr constraints

Name	Constraint
mrr_pkey	PRIMARY KEY (time_utc, paramset)

B.2.3 mrr_raw

Values of the raw unprocessed FFT data produced by MRR. The measurements are stored as arrays, where each element represents one height. Refer to the table mrr_parameters (via paramset id) for the height resolution used in each measurement.

Table 56: Table mrr_raw column descriptions

Column name	Type	Description
time_utc	timestamp without time zone	*NOT NULL default '1970-01-01 00:00:00+00'* Timestamp (UTC) of the recorded observation. See parameters table for timezone if you need local time.
paramset	integer	*NOT NULL default '0'* Parameters describing the settings used when recording the stored observation. Refers to table mrr_parameters.
reliability	integer	Estimated reliability of the measurement (NULL = unknown).
f	text[]	The received signal at each height step, unit unknown. Note that there are two extra height steps in raw data compared to processed data (at least in MRR-2): one at zero (first element of the array) and one above the maximum (last element of the array). Each element contains 64 floating point values (F00-F63), stored as text in csv format, values separated by commas. Two successive commas without a value in between means that the device did not provide a value for that row. Similarly, if the row ends in a comma the last value (F63) was missing.

Table 57: Table `mrr_raw` constraints

Name	Constraint
mrr_raw_pkey	PRIMARY KEY (time_utc, paramset)

References

[1] PostgreSQL database engine home page. `http://www.postgresql.org`

[2] T. Connolly, C. Begg, "Database Systems: A Practical Approach to Design, Implementation and Management", Third Edition, Addison-Wesley, 2002.

[3] Bruce Momjian, "PostgreSQL: Introduction and Concepts", Addison Wesley, 2000.

[4] JCC Consulting, Inc. SQL standards page. `http://www.jcc.com/SQLPages/jccs_sql.htm`

[5] SQLCourse, Interactive Online SQL Training for Beginners. `http://www.sqlcourse.com/`

[6] PostgreSQL online documentation. `http://www.postgresql.org/docs/`

[7] PgAccess, PostgreSQL database administration tool, home page. `http://www.pgaccess.org/`

[8] PhpPgAdmin, web-based administration tool for PostgreSQL, home page. `http://phppgadmin.sourceforge.net/`

[9] Python programming language home page. `http://www.python.org/`

[10] Gnuplot plotting program home page. `http://www.gnuplot.info/`

[11] Gnuplot plotting program documentation for version 3.8j (development version). `http://prdownloads.sourceforge.net/gnuplot/gnuplot-3.8j.0-docs.tar.gz?download`