HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering
Degree Programme in Computer Science and Engineering

Arto Teräs

Database for Ground Based Snowfall Observation

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Supervisor: Professor Olli Simula
Instructor: Professor Ken-ichiro Muramoto

Helsinki May 25, 2004

Author: Arto Teräs
Title of thesis: Database for Ground Based Snowfall Observation
Finnish title: Tietokanta lumisateen havainnointidatan tallennukseen
Date: May 25, 2004 **Pages:** 84

Department: Department of Computer Science and Engineering
Professorship: T-115 Computer and Information Science

Supervisor: Professor Olli Simula
Instructor: Professor Ken-ichiro Muramoto

This thesis presents a system based on a relational database for storing and analyzing snowfall observation data. Using a database facilitates data retrieval and collaboration between several research groups. Visualization tools were developed to quickly determine weather conditions and to detect possible problems with the measurement instruments.

The data used in the project was gathered as part of the Coordinated Enhanced Observing Period (CEOP), an international collaborative research campaign focusing particularly on the global water cycle. During winter 2003, a large number of instruments including radars, radiometers, an optical lidar and an image processing based snowfall observation system were installed at Fukui, Japan. They provided simultaneous measurements about cloud structure and precipitation rate, type, particle velocity and diameter distribution on the ground level. Clocks were synchronized and temporal resolution of less than one minute allowed distinguishing between different phases of a precipitation event.

Geographical location and other measurement parameters are stored with the data which makes the system suitable for large scale campaigns with multiple observation sites. In the future more data will be added which will provide a good foundation for statistical analysis and development of automatic classification algorithms.

The database tables were designed with SQL standard compliance, flexibility and extensibility in mind. Some performance issues were encountered during the project and resolved by deviating from the standard a little bit and using array related special features of the PostgreSQL database engine. An important goal was to make the table structure easy to understand so that scientists could focus on their research instead of studying the database or the original file formats produced by the instruments.

This database and visualization tools are designed for weather and particularly snowfall observation, but the same concept could be useful in many other types of research involving measurement data.

Keywords: database, snowfall, cloud, weather observation, visualization, Z-R relation, radar, lidar, image processing

Tekijä: Arto Teräs
Työn nimi: Tietokanta lumisateen havainnointidatan tallennukseen
English title: Database for Ground Based Snowfall Observation
Päivämäärä: 25.5.2004 **Sivumäärä:** 84

Osasto: Tietotekniikan osasto
Professuuri: T-115 Informaatiotekniikka

Valvoja: Professori Olli Simula
Ohjaaja: Professori Ken-ichiro Muramoto

Työssä suunniteltiin ja toteutettiin relaatiotietokantaan pohjautuva lumisateen havainnointidatan tallennukseen ja analyysiin tarkoitettu järjestelmä. Tietokanta tarjoaa yhtenäisen tavan päästä käsiksi dataan ja helpottaa sen yhteiskäyttöä useamman tutkimusryhmän kesken. Projektissa kehitettiin myös visualisointityökalut, joilla saadaan nopeasti yleiskuva säätilasta halutulla ajanhetkellä ja voidaan havaita mahdolliset ongelmat mittalaitteiden toiminnassa.

Projektissa käytetty data kerättiin kansainvälisessä Coordinated Enhanced Observation Period (CEOP) -hankkeessa, joka keskittyy erityisesti globaalin veden kierron tutkimukseen. Talvella 2003 suuri määrä erilaisia mittalaitteita, mm. erilaisia tutkia, radiometrejä ja videokameroihin ja kuvankäsittelyyn pohjautuva lumisateen havainnointilaitteisto oli asennettuna Fukuihin, Japaniin. Laitteista saatiin samanaikaista mittaustietoa pilvien rakenteesta sekä sadehiukkasten tyypistä, koko- ja nopeusjakaumasta maanpinnan tasolla. Laitteiden kellot oli tahdistettu ja ne tuottivat mittausarvoja vähintään kerran minuutissa, joka riittää yksittäisenkin sade- tai lumikuuron eri vaiheiden erottamiseen toisistaan.

Laitteiden maantieteellinen sijainti ja asetukset tallennetaan mittaustulosten yhteyteen, joten järjestelmä soveltuu myös useita havainnointipaikkoja käsittäviin laajoihin hankkeisiin. Jatkossa tietokantaan on tarkoitus syöttää lisää dataa jolloin siitä muodostuu hyvä pohja tilastolliseen analyysiin ja automaattisten luokittelualgoritmien kehittämiseen.

Tietokannan taulujen suunnittelussa huomioitiin SQL-standardi, joustavuus ja laajennettavuus. Projektin aikana kohdattiin joitakin suorituskykyongelmia, jotka ratkaistiin poikkeamalla standardista hieman ja käyttämällä taulukoiden tallennukseen liittyviä PostgreSQL-tietokannan erityisominaisuuksia. Tärkeä päämäärä oli saada taulurakenteesta helposti ymmärrettävä, ettei aika tuhlaantuisi tietokannan tai laitteiden alkuperäisten tiedostomuotojen opiskeluun vaan tutkijat voisivat keskittyä uusien algoritmien kehittämiseen.

Työssä kuvattu tietokanta ja visualisointityökalut on suunniteltu säätutkimukseen ja erityisesti lumisateen havainnointiin, mutta samankaltainen ratkaisu voisi olla hyödyllinen monessa muussakin tutkimusprojektissa, jossa käsitellään mittaustietoa.

Avainsanat: tietokanta, lumisade, pilvi, säähavainto, visualisointi, Z-R-suhde, tutka, pilvenkorkeusmittari, kuvankäsittely

Preface

The work described in this thesis was done during my exchange year in Japan, where I was a research student in the Image Information Science laboratory at Kanazawa University.

I wish to express my gratitude to professor Ken-ichiro Muramoto, who provided me excellent facilities in his laboratory and supported me during the whole project. Professor Olli Simula reviewed my thesis several times during the writing process and guided me to focus on important topics.

Thomas Pfaff introduced me to radio soundings and radiometers and shared his knowledge about radars and meteorology. Thomas also used the database heavily already during the prototype phase and his bug reports, questions and suggestions were very useful in the development. Masahiro Tamura and Hideki Aoyama spent long nights in the garage at Fukui airport monitoring the measurement instruments and writing weather notes. Marko Niinimäki helped me in database related topics.

I also want to thank all the people who made my stay in Japan such a rich and enjoyable experience. The whole exchange year wouldn't have been possible without the help of my Japanese professor Junichiro Okura, Anneli Sinkkonen and Henri Servomaa. Kanazawa University Student Exchange Division staff made great effort in arranging all necessary practical issues related to living in Japan. Mamoru Kubo, Ryotaro Komura and other laboratory members were always available when I needed help or had questions about Japanese language and culture. Everywhere I went I was surrounded with hospitality and friendliness I will remember forever.

Finally, I thank my family for continuous support and encouragement during all my study years. It didn't matter whether we were living under the same roof or thousands of kilometers apart — you were always there with me.

Helsinki May 25, 2004

Arto Teräs

Contents

1 Introduction **8**

2 Snowfall **10**

2.1 Clouds and Snow 10

2.2 Snowflake Measurements 10

2.3 Radar and Satellite Observations 11

2.4 Snowfall Characteristics on the Sea of Japan Coast 12

2.5 Coordinated Enhanced Observing Period and the Wakasa Bay Experiment . 13

2.5.1 Instruments 13

2.5.2 Snowfall events and field work 14

3 Database as a Research Aid **16**

3.1 Need for a Database 16

3.2 Goals of the Project 17

3.3 Comparison with Other Database Applications 17

3.4 Choosing the Right Database and Tools 18

3.4.1 Relational and Object Databases 18

3.4.2 Database Engines 18

3.4.3 Compatibility Issues 19

3.4.4 Chosen Software and Hardware 19

4 Design and Implementation **20**

4.1 Table Structure 20

4.1.1 Data tables and parameter tables 21

4.1.2 Measurement Data Layout 22

4.1.3 Normalization 25

4.2 Inserting and Deleting Data 27

4.3 Data Retrieval and Search Functionality 29

4.3.1 Timestamp Synchronization 29
4.3.2 Continuous and Discontinuous Blocks 30
4.4 Expandability and Flexibility 31

5 Snowfall Event Analysis 32
5.1 System Characteristics 32
5.2 Visualization Tools 32
5.3 Analysis examples 33
5.4 Suitability for Statistical Analysis 36
5.5 User Experiences 37

6 System Performance 38
6.1 Performance Requirements 38
6.2 System Setup 38
6.2.1 Test Data 38
6.2.2 Hardware and Software Environment 40
6.2.3 Main Factors Affecting Performance 41
6.3 Results 42
6.3.1 Inserting Data 42
6.3.2 Visualization Tools 43
6.3.3 Exporting Data 47
6.4 Result Analysis and Scalability Estimates 49

7 Application to Analysis of Relationship Between Radar Reflectivity Factor and Snowfall Rate 51
7.1 Background 51
7.2 Observation Data, Database and Data Extraction 51
7.3 Division Into Individual Snowfall Events 52
7.4 Shortening and Division of Selected Events Using Optical Lidar Data . . . 54
7.5 Discussion 57

8 Future Experiments 58

9 Conclusion 59

10 Acknowledgements 61

A Database Table Layout **62**

A.1 Standard version 63

A.2 Array version 65

B Detailed Results of Performance Measurements **67**

B.1 Graph Types, Command Lines and SQL Queries 67

B.2 Visualization Tools 73

B.3 Exporting data 75

C Visualization examples **77**

Chapter 1

Introduction

In Kanazawa, Japan, there's a saying "Bentou wo wasuretemo kasa wa wasurena", which translates to "Even if you forget your lunch box, don't forget your umbrella". Indeed, the weather can change rapidly from blue sky to rain, sleet, snowfall or even a graupel throwing thunderstorm. This provides ideal conditions for the snowfall team in Image Information Science laboratory at Kanazawa University. The team has been studying characteristics of different snowfall events using video cameras, radars and other instruments for several years.

To gain better understanding of weather conditions and develop better forecasts, it is important to compare measurements of several different kinds of instruments. Good ground reference data at many sites around the world is necessary to calibrate new satellite based observation instruments and to validate global weather models. Weather phenomena extend beyond nations and continents which is why researchers from all around the world need to work together and access each others' data.

This work describes a database designed for storing measurement data of snowfall research. It was developed to address a specific need to share data between several research groups which participated in an observation campaign in Fukui, Japan, January-February 2003. The number of measurement instruments was large and there were many practical problems in using the data which was scattered in text and binary files of various formats. Transferring measurement values to a relational database helps the groups to manage their data and allows them to concentrate on developing new analysis algorithms.

Although the database tables are designed for weather observation and a certain set of instruments, the same concept could be used for many other types of research involving measurement data. As the amount of data grows and statistical algorithms together with numerical simulations become more prominent way of doing research, it is likely that databases only increase their popularity.

Chapter 2 is an introduction to snowfall, instruments used to measure precipitation and the Wakasa Bay experiment which initiated this project. Chapter 3 describes how databases can be useful for researchers, covers available databases types and justifies why a relational database was suitable in this case. Chapter 4 presents the details of table design and data retrieval and search functionality.

Chapter 5 introduces visualization tools which allow to easily plot data from several measurement instruments side by side and get a quick understanding of weather conditions.

Simple snowfall analysis examples are also given. Chapter 6 presents performance measurements and scalability estimates of the database system in various usage scenarios. Chapter 7 contains an example of how the database and visualization tools were used to analyze relationship between radar reflectivity factor and snowfall rate. Chapter 8 describes future plans for the system and conclusions are presented in Chapter 9. Complete table design diagrams, details of performance measurements and additional examples of visualization tools can be found in the appendices.

Chapter 2

Snowfall

During winters, snow is abundant in northern and southern areas of the globe. Yet two snow crystals have never been found to be identical. There are large wet snowflakes, tiny ice needles, hard graupel pellets, gently falling small crystals. Size, velocity, density, temperature and shape are widely varying. The Inuits have at least 13 separate words for different types of snow [1].

2.1 Clouds and Snow

In the atmosphere, concentrations of water and ice are called clouds. There are basically two methods how ice crystals can be formed: the freezing of a liquid droplet and direct sublimation of vapor to the solid phase. After the initial formation, the crystal can grow by diffusion from other droplets or by colliding with other crystals, which is called aggregation [2]. These processes are happening constantly inside the cloud, and therefore normally many types of particles coexist.

Precipitation reaching the ground doesn't include the smallest particles, and it may also be in different form. For example, most of the rain that falls to ground surface is in fact melted snow. Therefore, to fully understand rain- and snowfall, we have to make observations on the ground, look inside the clouds and take larger pictures of the cloud shape using radars and satellites.

2.2 Snowflake Measurements

Individual snowflakes have been studied in detail for decades. The first scientific reference is "On the Six-Cornered Snowflake" by Johannes Kepler, written in 1611. In the early 1900s, thousands of snow crystal photographs were taken and published by Wilson A. Bentley. Ukichiro Nakaya created the first artificial snow crystals in laboratory conditions in 1936. More recently, very detailed microscope images have been published, the shapes and structures of snowflake types have been imitated in laboratories and the formation of different types of snow has been studied [3] [4]. Also, the relation of size, shape and fall velocity of precipitation particles has been studied both on the ground level and in the clouds [2] [5] [6].

On the other hand, long term statistics of the amount and type of snowfall and related data such as temperature have been gathered at all regions around the world. Anyone who is interested can participate in this activity, for example by buying a weather station and sending the data via Internet to Weather Underground [7].

However, long term observations including detailed snowflake data have only been done fairly recently. In Kanazawa University, Hokkaido University and Toyama National College of Technology in Japan, a video camera and image processing based observation system has been developed to continuously measure the size and velocity distribution of falling snowflakes (Figure 2.1) [8]. A similar type of system has also been designed and constructed by Johanneum Research in Austria [11]. An optical spectrometer with a smaller sampling area but capability of recording detailed snowflake shape features and velocity has been developed in Switzerland [12].

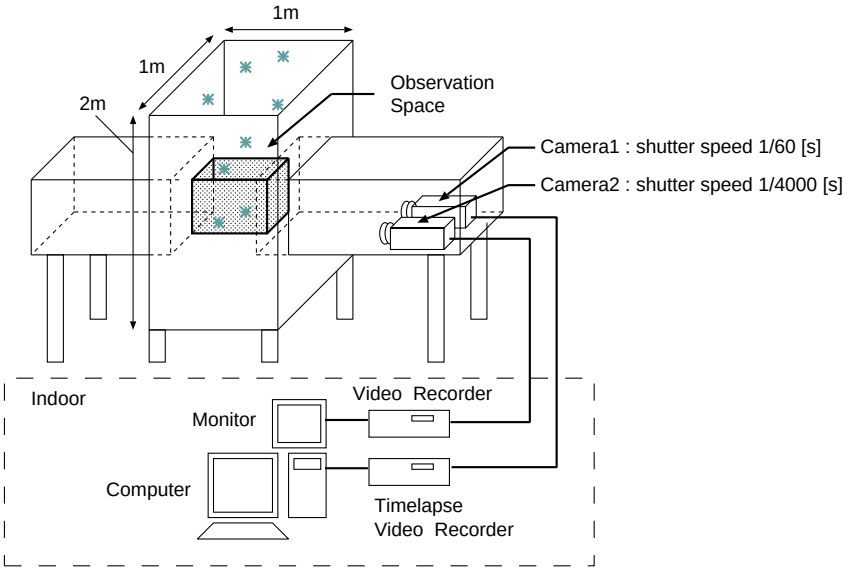

Figure 2.1: Video camera based observation system to measure snowflake size and velocity distributions.

2.3 Radar and Satellite Observations

Radars and satellites are currently the main tools to obtain meteorological data of large areas. Both show the location and shape of clouds, radars can also be used to look inside the clouds. Radars have been used in meteorology since 1940s, and doppler radars which are capable of not only measuring the distance but also the speed of the targets, became common in 1970s [13].

Radar measurements of precipitation are based on electromagnetic waves being scattered by precipitation particles. The radar sends pulses or frequency modulated continuous waves and receives the signals reflected back from the particles, called backscatter. The volume of one measurement in all meteorological radars is orders of magnitude larger than an in-

dividual precipitation particle, therefore the reflectivity values measured by the radar are a combination of small reflections from a very large number of raindrops, snowflakes and other particles.

Generic theory of the scattering of electromagnetic waves by particles was developed by Mie in early 20th century. The scattering depends on the amount, size and material of particles, and also the radar wavelength. When the size of the particles is small compared to the wavelength, the backscattering can be approximated by a simpler formula called Rayleigh approximation [14]. For raindrops it is usually quite accurate with current radar wavelengths, but in snowfall it doesn't always apply [15].

The relationship between radar reflectivity and amount of precipitation is called the $Z-R$ relation. It is usually estimated by the classic equation first presented by Marshall and Gunn [16]:

$$Z = BR^{\beta} \quad [mm^6/m^3], \tag{2.1}$$

where Z is reflectivity, R is precipitation intensity in mm/h (liquid water) and B and β are coefficients. However, the coefficients B and β are not universal constants, but vary depending on the radar and the type, size and shape of precipitation particles.

For rainfall, it is possible to find values for constants B and β so that the $Z-R$ relationship is adequately modeled for a particular type of radar. The case of snowfall is more difficult, because the snowflakes vary not only in size and speed, but also shape and density. Therefore it is difficult and unreliable to estimate the amount of snowfall using a single radar. New techniques using two or several radars of different wavelengths have been proposed [17] [18] to improve the estimates. Another approach using other devices to determine the snowfall type and calculating B and β separately for different types of events is being studied in Kanazawa and described in chapter 7 of this thesis. Still, a lot of work remains before snowfall can be measured reliably and accurately using radars.

Satellites are convenient for observing large continuous areas. From the 1960s until present, they have been used to take 2-dimensional images of cloud tops in increasingly better resolution. Radar technology to achieve 3-dimensional data is difficult to use in satellites because of the large distance and interference of the surface of the earth. However, in the late 1990s good results for rainfall have been obtained in the Tropical Rainfall Measuring Mission [19]. Another recent technology currently being integrated to meteorological satellites are radiometers: passive receivers which measure the natural radiation of microwave frequencies, reflecting the humidity and amount of different types of molecules between the satellite and ground level.

To better understand the relations between radar and satellite measurements and precipitation, obtaining reference data both on the ground level and inside clouds is essential. In-cloud measurements can be done using specially equipped aircrafts and radio soundings, ground level observations involve setting up measurement instruments in several locations around the world.

2.4 Snowfall Characteristics on the Sea of Japan Coast

By the Sea of Japan, clouds brought by the north-western wind are often stopped by mountains and release their precipitation on the coast. During winter, the temperature is usually

around or slightly above zero degrees Celsius, which leads to frequent and varying types of precipitation events.

The type of snowflakes in the area is usually graupel of snowflake aggregate, and it often changes from graupel to aggregate and sometimes from aggregate to graupel. Raindrops mixed with snowflakes are also very common. Individual snowfall events are often short, and can include many types of precipitation.

These fast changing conditions present a challenge to meteorologists. It is difficult to estimate the precipitation type and amount from satellite or radar data alone. On the other hand, the frequent and varying precipitation events offer an ideal opportunity for observing and studying the relation between different types of clouds and precipitation. In the Image Information Science laboratory at Kanazawa University [20], rain- and snowfall has been studied for several years, using higher temporal resolution than most other meteorological observations [21].

2.5 Coordinated Enhanced Observing Period and the Wakasa Bay Experiment, Winter 2003

International collaboration in the Coordinated Enhanced Observing Period (CEOP) [22] has been designed to promote cooperation in global climate research, focusing particularly on the global water cycle. The project started in July 2001 and will last for a total of 4 years. Satellite data is used extensively and compared with in-situ observation data at reference sites around the world.

As part of CEOP and the AMSR/AMSR-E satellite data validation program [23], a rainfall and snowfall observation campaign was carried out at Fukui airport and over Wakasa Bay, located north of Osaka on the Sea of Japan, from January 13 to February 6, 2003. The participants were University of Tokyo [24], Kanazawa University [20], Meteorological Research Agency of Japan [25], National Space Development Agency of Japan (NASDA, later merged to JAXA [26]), National Aeronautics and Space Administration of the United States (NASA) [27] [28], Fukui University [29], Sankosha Corporation [30], Yamada Giken Corporation [31] and the Japan Science and Technology Agency [32]. The aim was to gather data for CEOP and also for various research projects done independently at the laboratories of the participating universities and institutes.

2.5.1 Instruments

Measurements were done both at ground level and at high altitudes. An overview of the used instruments is shown in Figure 2.2. More detailed list of instruments installed at ground level is shown in Table 2.1.

In addition, radiosondes measuring temperature, humidity and wind profiles were released four times a day. NASA P-3 Orion aircraft did periodic measurements using a dual frequency precipitation radar, a cloud radar and two radiometers. Data for the same region was also obtained from a Sankosha DWSR-2501C dual polarization doppler radar installed 10 kilometers away. The total amount of data produced by the instruments was about 2 gigabytes per day.

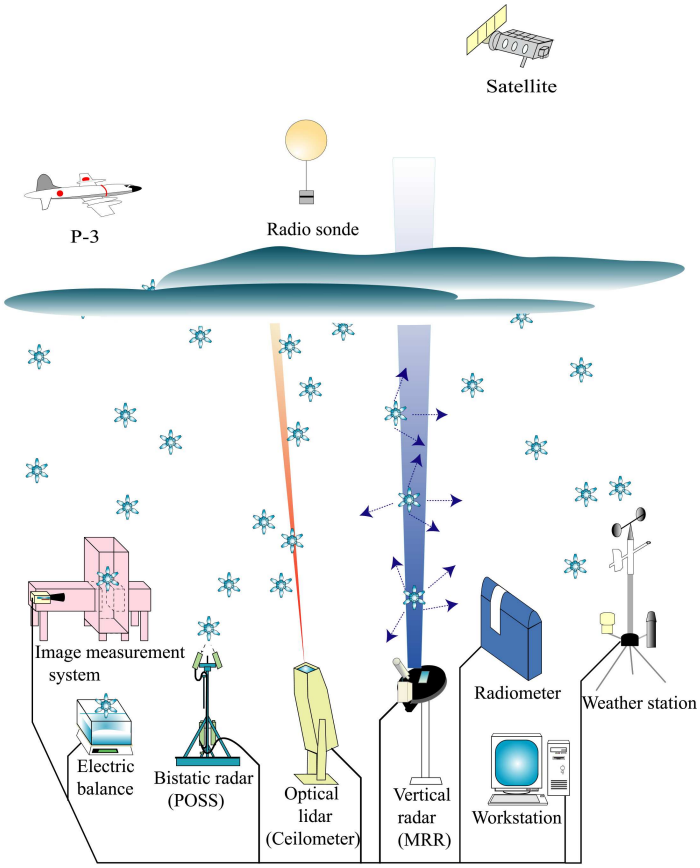

Figure 2.2: Wakasa Bay observation campaign overview.

2.5.2 Snowfall events and field work

During the Wakasa Bay experiment, many types of precipitation events occurred. During the first two weeks most precipitation was rain, complemented by a few graupel showers. At the end of the observation period the temperature dropped and snowfalls became more frequent. Big and small snowflakes, graupel and sleet were observed, and there was a three day period of snowstorms with heavy snowfall and strong wind.

As could be expected, various practical problems were encountered during the experiment. For example, wet snow accumulated on the MRR radar dish affecting the measurements, and the radar didn't work very well even in more favorable conditions. The rain gauge attached to one of the weather stations gave completely meaningless results in snowfall. Wind was often quite heavy and coming from unexpected south-western direction, which caused some bias to measurements done using devices which were installed inside a wind-breaker net (electronic balance, video camera based observation system, snow heat capacity sensor). All these experiences suggest that it is useful to have several instruments measuring the same elements — for example snowfall rate — so that results can be compared with each other.

Weather conditions were also observed visually and written in a logbook during the whole experiment, which proved very useful when analyzing the measurements later.

Table 2.1: Instruments installed at ground level during the Wakasa bay observation campaign

Instrument	Measured elements
Video camera and image processing based snowfall measurement system [8]	Number concentration of snow particles Size distribution of snow particles Velocity distribution of snow particles
Electronic balance	Precipitation rate (calculated from weight data)
Andrew POSS ground level bi-static radar	Reflectivity, doppler velocity Estimated precipitation type and rate
Metek MRR-2 vertically pointing doppler radar	Reflectivity and doppler velocity up to 6000 m Rain rate, drop size distribution (calculated, for water only)
Vaisala Ceilometer CT-25K optical lidar	Optical backscatter profile up to 7680 m Cloud base height
Radiometrics WVR1100 radiometer	Brightness temperatures for 23.8 and 31.4 GHz Integrated water vapor and liquid water path
Radiometer Physics GmbH RPG-LWP and RPG-TEMPRO 90 radiometers	Brightness temperatures for 23.8, 36.5 and 90 GHz Oxygen line profile consisting of 10 frequencies between 50 and 60 GHz
Snow heat capacity sensor	Amount of heat to melt snow particles
GPS sensor	Time Integrated water vapor
Weather station (2 units)	Rain rate by rain gauge Wind direction and strength Temperature, humidity, air pressure

Chapter 3

Database as a Research Aid

Research projects involving observations of natural phenomena often gather significant amounts of data. This data is commonly used to test several algorithms and results analyzed statistically. In this kind of environment, a database can ease data management and allow researchers to concentrate on the actual analysis.

In meteorological research, most data is still stored in custom file formats produced by radars and other instruments. The popularity of databases has been increasing, especially for gathering climate data of long time periods and providing easy access for users [9]. In some cases database systems have been used to select right data files from a large data set [10]. The project described in this thesis pushes the database to a lower level by using it as the storage for raw measurement data. The goal is not to develop new database technology but use a generic database as a tool for other research.

3.1 Need for a Database

The Wakasa observation (Section 2.5) was a joint project of several groups of researchers. Each of them brought their own instruments and expertise to the site, and the plan was to share data between participants. However, there were many practical problems. The data from each instrument was saved in text or binary files, most of them in different formats. Usually there was one file per day, sometimes one file per hour. Some files had timestamps in UTC time, others in local time. File naming convention and directory structure varied widely. Groups were generally familiar with the data and file formats of their own equipment, but not other groups' instruments. Furthermore, there were no simple visualization tools for some data, and the existing tools were separate programs with different interfaces and operating system requirements, making it difficult to do comparisons between instruments.

Database seemed to be the best way to achieve more convenient and uniform access to the data. There was another database project in CEOP, aiming to store satellite data and certain standardized weather data from reference sites around the world, but it was not ready and not planned to include support for all instruments used in the Wakasa experiment in Fukui. There was a clear demand for a relatively simple yet flexible database setup which could be used in various measurements with different device configurations.

3.2 Goals of the Project

The first goal of the project was to gather data collected at Wakasa experiment in an easily accessible format, of which a copy could be made for each group. Satellite data, aircraft data and external Sankosha radar data were left out at this phase, the focus was on the instruments installed at the main observation site.

A longer term goal was to create a system which would be easy to use also in future experiments. Some key features were identified as follows.

- Support observation data from several instruments installed at different locations.
- Provide an easy way to insert new data gathered in future observation campaigns.
- Provide basic visualizations of the data produced by each instrument.
- Allow to easily retrieve any time range of data for further analysis with external tools such as Excel, Matlab and custom applications.
- Be easy to understand so that new analysis tools could be written to use data in the database directly.

In the future, the plan is to insert data in the base already during an observation campaign. Then the visualization tools could be used to get an overview what is happening and detect possible problems with instruments early. Another major goal was to facilitate statistical analysis and cross-analysis between instruments, by reducing the time which is currently used to extract the relevant data from the original text files.

3.3 Comparison with Other Database Applications

In most database applications, the main view to the system is a graphical interface which hides all the internals of data storage from users. However, in this project the users are scientists which will analyze the data using various external tools and often also write their own small programs to test a new algorithm. Therefore, one simple interface to the database cannot satisfy all analysis needs. Data should be simple to examine and export to external programs.

The structure of the system should also be easy to understand. It has been planned to have one administrator to take care of database specific tasks, but other users will also be accessing raw data and writing programs which read values directly from the database. They are more advanced computer users than an average person, but cannot be expected to have much experience with databases.

The database administrator will most likely be one of the scientists and user of the data as well. However, he will handle additional tasks such as installing new software, taking backups, inserting new data and verifying that it has been inserted correctly. Generating a set of graphs and statistics for the whole time period after an observation campaign will be another task of the administrator. He might also create new tables to the database for storing intermediate analysis results. When new instruments are acquired, the administrator or another developer should extend the structure and source code so that new types of measurements can be inserted to the same database.

3.4 Choosing the Right Database and Tools

3.4.1 Relational and Object Databases

Early database technologies included hierarchical and network data models [33]. Relational databases were invented in the 1970s and became popular about one decade later. The relational model was the first database technology which achieved standardization between different vendors. It is based on two-dimensional tables, or relations in database terminology, and data in the base can be manipulated using the Structured Query Language (SQL), standardized by a joint effort of American National Standards Institute (ANSI) and International Standards Organization (ISO). At least in theory, all compliant database engines from different vendors implement the same set of SQL commands and the application programmer does not need to care of which engine is used.

One major disadvantage of relational databases is the need to map the data of the application to the relational model. Especially when using object oriented programming languages, this translation is complex — it has been claimed that as much as 30% of programming effort is devoted to translations of data between the database and the application. Various object-oriented databases have been developed to address this issue and make database access more seamless for the programmer. However, there is no universal standard for object-oriented bases and they have not yet reached the same level of popularity as relational systems. In 1997 the share of object-oriented systems was about 3% of the overall database market [34].

The relational database camp has also approached object modeling. Most popular database engines today are relational but include lots of vendor-specific extensions to the traditional relational model. These extensions are helpful in many applications but incompatible with each other, making it harder to switch between different database engines. The latest version of SQL standard, SQL3, also takes a major step in the object oriented direction. Unfortunately, none of the current database engines on market is even near to implement SQL3 completely. The last widely supported version is SQL2 which appeared in 1992[1].

3.4.2 Database Engines

Database engines compete based on their features, performance, ease of use, price, type of license and user support. Traditionally the field has been dominated by large companies, the best known products being Oracle from Oracle Corporation, DB2 from IBM and SQL Server from Microsoft. These products are all relatively expensive, typical licenses starting from a few thousand euros and going up to hundreds of thousands for large installations. Various light-weight versions such as Microsoft Access have also been popular, but they have missed critical features such as proper transaction support.

During the last few years, two open source database engines have changed the field considerably. MySQL [36] and PostgreSQL [37] have made real database functionality available to smaller projects with limited budget. MySQL and PostgreSQL are both free of charge and can also be redistributed and modified within the conditions of two popular open source licenses. MySQL is available under the GNU General Public License (GPL) [38] and PostgreSQL under the BSD License [39]. Also two other formerly proprietary database engines

[1] Even SQL2 is often not implemented in entirety, but commonly used features of the standard are available in most databases.

have been made available as open source: Firebird [40] (previously known as Interbase) by Borland and SAPDB by SAP, recently bought by MySQL and re-branded as MaxDB [41].

The development of open source databases has led to numerous other open source projects which use a database engine as a backend for storing data. Especially web applications commonly rely on a database. This has in turn further spurred the development of the databases themselves and made them easier to use from a large variety of programming languages. It can be said that database technology has moved from the domain of dedicated database experts to a common building block of any software developer.

3.4.3 Compatibility Issues

As mentioned in Section 3.3, the users of the database in Wakasa project are scientists which will use a variety of tools to analyze the data. Therefore, compatibility with third-party database tools, large number of programming languages and conformance to standards was important in this project. Ideally, the data should be also easy to move to another database engine, but this was considered less critical.

One question was whether to use strictly standard SQL or exploit various database specific extensions. This is often a compromise between compatibility and ease of implementation or performance. If extensions make using the data easier or offer a major speed improvement, the benefits can overweigh the disadvantages.

3.4.4 Chosen Software and Hardware

After a short consideration of other alternatives, PostgreSQL was chosen as the database engine. It was open source, relatively well standards compliant, had decent documentation and the necessary features for the project, including extended support for arrays and matrices which seemed useful considering the type of gathered data.

To implement data insertion, retrieval and visualization, Python programming language [42] was chosen. It is a modern language which has been gaining popularity and gathered an enthusiastic group of users during the last few years. Python quickly proved to be a good choice - the language was easy to learn and implementing the necessary functionality required considerably fewer lines of code than doing the same for example in C language.

Gnuplot [43] was chosen as the plotting backend for data visualizations. Proprietary packages such as Matlab offer more elaborate graphics functionality, but the aim was to use only open source components and Gnuplot functionality was sufficient[2]. There was also a Python module which provided an easy connection interface with Gnuplot.

The hardware used for initial development was a standard desktop PC with Intel Celeron 600 MHz processor and 512 MB of memory. When taking the system to production use, a 800 MHz Pentium III computer with 1024 MB of memory was dedicated as the database server. It can be easily upgraded later if the performance turns out to be insufficient. In addition, the system was installed for demonstration purposes on a Pentium MMX 266 MHz laptop with 64 MB of memory.

[2]The most recent "stable" version of Gnuplot available at the time of development (version 3.7.3) was missing certain features and therefore the experimental "development" version was chosen.

Chapter 4

Design and Implementation

Designing a good table layout is an important part of any project involving a relational database. The database described in this thesis was developed for storing data of the Wakasa experiment, and therefore will be referred to as the "Wakasa database". However, from the very beginning one important goal was to design a system which could be used for other observation campaigns and measurements without modifications, only adding tables for new instruments.

4.1 Table Structure

Usually the first step before any thought on database table structure is to collect information about the real world case being modeled. It is important to detect which information is necessary to store in the database and find relationships and dependencies between data. The relationships can be modeled for example using the Entity-Relationship (ER) model [33] or less formal methods [35] and further transferred to relational database tables.

In the case of Wakasa database, it was already clear that data which needed to be stored were the values produced by the measurement instruments. There was no need to carry out extensive research to find the relationships and dependencies. The task in designing the table structure was to find an efficient and intuitive way to map the gathered data into the relational model.

The Wakasa database contains data from 9 different types of instruments. For each instrument type, there is one parameter table and one or more data tables. An overview of the structure is shown in Figure 4.1. A more detailed figure of the structure, including labels of individual columns, can be found in Appendix A.

Actually, two versions of the Wakasa database were developed: **standard version** and **array version**. The structure shown in Figure 4.1 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 specifi c extension to standard SQL.

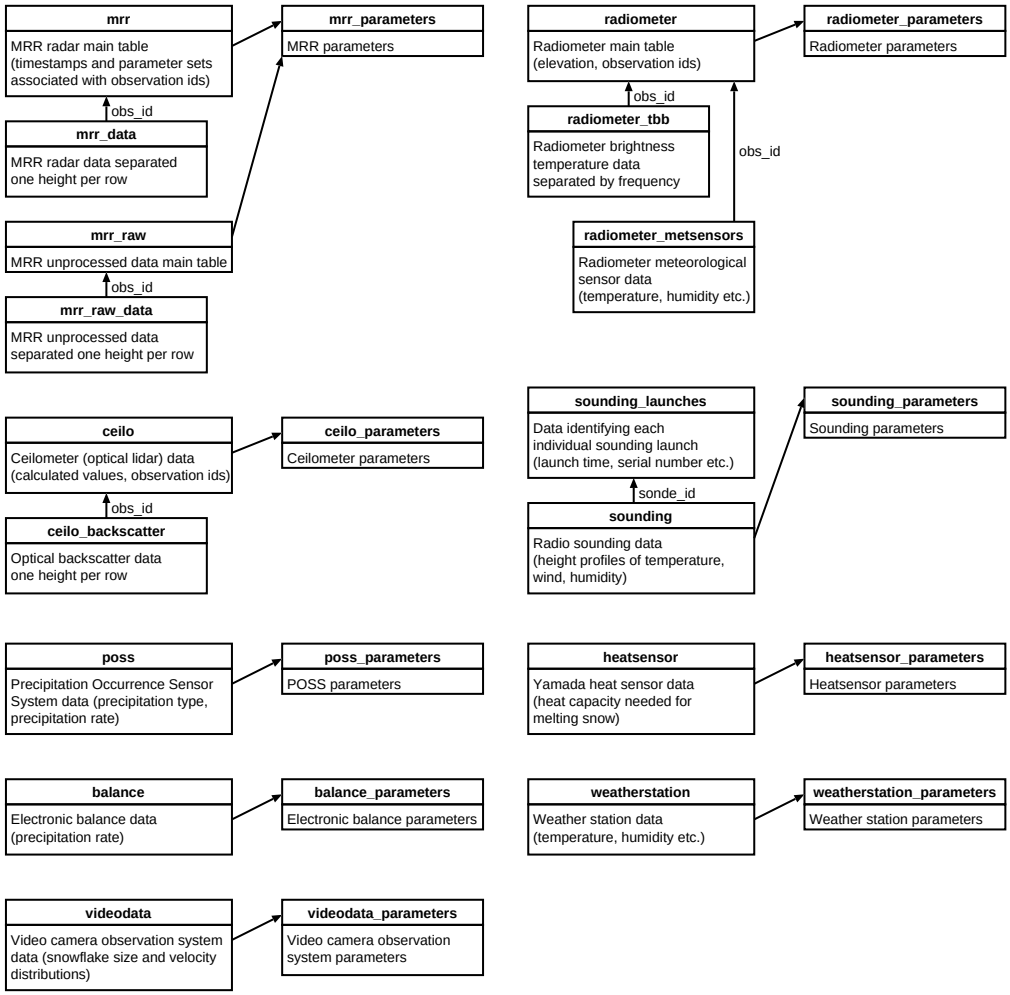

Figure 4.1: Wakasa database structure, standard version.

4.1.1 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** 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 4.2 illustrates how the data is stored.

One data table can contain data from several similar instruments. In the example of Figure 4.2 there are three weather stations, one located in Kanazawa and two in Fukui, represented by parameter set id's 1, 2 and 3, respectively. The measurement data in data tables is linked to the parameter sets using these id numbers, which are generated automatically by the database engine. Each parameter set also has a more descriptive name chosen by the administrator when feeding data in: `Kanazawa-jwa`, `Fukui2003-aws` and `Fukui2003-air` in this example.

All data tables have certain common columns: timestamp (in universal coordinated time), parameter set id and reliability. One observation is identified by the combination of times-

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

Figure 4.2: Weather station data in the database

tamp and parameter set id, in other words the pair (`time_utc`, `paramset_id`) is the **key** of the table. The reliability field is reserved for the future: the idea is to define a coding for the reliability of the measurement. Other columns in the data tables contain measurement results and are thus specific to the type of instrument.

Respectively, the parameter tables have certain columns which are uniform between instruments, and other instrument specific columns. The most important common columns are location information (`location_latitude`, `location_longitude` and `location_elevation`) which can be used to select correct data when results are available from several observation sites concurrently.

Parameter sets can also be used to separate between several similar instruments installed on the same location, even when their configuration is identical. Running several instruments in parallel and comparing the results improves the reliability of measurement data. Creating a unique parameter set id for each instrument makes it possible to store the data from all instruments in the same data table but still separate between individual instruments later during analysis.

4.1.2 Measurement Data Layout

As a basic rule, each measurement by one instrument (a set of values produced by the instrument at a given time) is represented on one row in the data table. Timestamps are stored in universal coordinated time (UTC) to ensure comparability between instruments. If some values of the measurement are missing or invalid, they are represented by the special value `NULL`.

Individual values for each timestamp can be easily mapped to the two-dimensional relational model. For example, temperature and humidity are two columns in the weatherstation table, and precipitation rate estimate is one column in the POSS radar table. However, many devices produce array or matrix values. For example, the optical lidar gives the reflectivity profile from 0 to 7680 meters in 30 meter height steps and the POSS radar reports the fast fourier transform (FFT) spectra of received signal near the ground level. The MRR radar measures the FFT spectra for 30 separate height steps, leading to a two-dimensional matrix

for each individual timestamp. Arrays and matrices can be inserted in the database using a few different methods:

1. Create as many columns as the maximum array length.
2. Store the array or matrix in a separate file and insert only the file name and position indexes in the database.
3. Encode the array or matrix as one long string or binary value which is inserted in the table cell.
4. Create additional data tables where each array is distributed to several rows.
5. Use database specific features like array datatype or user specific datatypes, not part of SQL-92.

The first method can be used only for fixed size arrays and selecting the desired range using SQL language is inconvenient. The second method leaves the actual data outside the database and can usually be replaced by the third method — modern databases internally store large data blocks in separate files to maintain decent performance. In a purely relational model, method 4 is the only right way to proceed. However, methods 3 and 5 are also often used in practice.

Storing the data as one string or binary block (method 3) is often a good choice if the whole block is needed during processing. The disadvantage is that individual values from the array cannot be retrieved and SQL search operations cannot look inside the block. Method 4 preserves the possibility to access and search for individual values, but performance can suffer heavily if large blocks of the data are needed. Method 5 can ideally combine the advantages of methods 3 and 4 but deviates from the SQL standard[2], implementation differs between database engines and search functionality may be limited.

In the Wakasa project, it seemed important to be able to select values of a certain height or of an arbitrary height range when retrieving data. Therefore arrays consisting of data from different heights were split to additional data tables (method 4, standard version). To compare performance, an alternative version using PostgreSQL array extensions (method 5, array version) was also developed. Most other arrays and matrices such as FFT spectra and snowflake diameter and velocity distribution data were stored as comma separated value strings (method 3).

The standard and array versions of the system differ only in the table layouts of the optical lidar (ceilometer) and mrr radar. Figures 4.3 and 4.4 illustrate how the optical lidar data is stored in the two versions. In the standard version, the lidar backscatter height profile is stored in a separate table `ceilo_backscatter`, 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 timestamps, parameter set id numbers and other data (cloud base height value, beam power, error codes etc.) are stored in the main data table.

In the array version, backscatter values have been moved to one column called `bs` of the main data table. Each value or a slice of the array can still be individually retrieved using

[2] SQL3 includes an optional array datatype, but few databases implement it yet.

ceilo

time_utc	paramset	obs_id	cb_height_1	other columns
2003-01-28 23:05:00	1	1247	850	. . .
2003-01-28 23:05:30	1	1248	820	. . .
2003-01-28 23:06:00	1	1249	780	. . .

ceilo_backscatter

obs_id	height	bs
1247	30	0.1660
1247	60	0.1182
1247	90	0.0964
...	...	...
1248	30	0.1047
1248	60	0.0853

Figure 4.3: Data tables for the optical lidar, standard version.

ceilo

time_utc	paramset	cb_height_1	. . .	bs			
2003-01-28 23:05:00	1	850	. . .	0.1660	0.1182	. . .	0.0000
2003-01-28 23:05:30	1	820	. . .	0.1047	0.0853	. . .	0.0000
2003-01-28 23:06:00	1	780	. . .	0.0803	0.0725	. . .	-9999

Figure 4.4: Data table for the optical lidar, array version.

SQL[3]. The height information is not directly visible, but because the height resolution is constant the indexes to retrieve the relevant part of the array can be easily calculated. Another subtle difference to the standard version is that NULL values cannot be used inside arrays to mark for missing values. Therefore a special number -9999 which never occurs in real data was chosen instead. This is simply a limitation of the PostgreSQL array implementation and may be changed in the future.

The advantage of the array version is increased performance. The difference is particularly significant when inserting or retrieving long time ranges of data using scripting languages. The main problem in the standard version 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 by an order of magnitude. The same data could of course also be encoded in the cells using text strings, but then it would no longer be possible to retrieve individual values or height ranges.

Some of the data tables, for example weather station and radiometer tables, are generic and suitable for many manufacturers and models. If one type of data is not available in a specific model, it can be marked as a NULL value in the database. Most other tables are specific to one particular manufacturer and model of the instrument.

[3]Actually, selecting a slice of an array is also a PostgreSQL extension, but integrates nicely to SQL language.

4.1.3 Normalization

Normalization is a process for organizing the table structure in the database so that data redundancy and hidden dependencies are removed or minimized. It defines a number of normal forms and a series of tests can be performed on a table whether it satisfies or violates the requirements of a given normal form. Normalization is explained in detail in any good book about relational databases, for example [34]. In the rest of this section the reader is assumed to be familiar with the concepts of normalization.

The first normal form (1NF) is defined as *A relation* (a table) *in which the intersection of each row and column contains one and only one value*. In the Wakasa database, there are sometimes several measurement values encoded as a string in one table cell. However, the string can be accessed as only one block and is only one value from the database point of view. The PostgreSQL array extension could be considered to violate this requirement by allowing to store several individually accessible values in one table cell, but the array as a whole can also be seen as one special value, preserving the normal form. Therefore it can be said that all tables in the Wakasa database are in the first normal form.

The second normal form (2NF) is defined as *A relation that is in first normal form and every non-primary-key attribute is fully functionally dependent on the primary key*. This requires removal of relations where some attribute (column) would be dependent only on some subset of the primary key. All tables with a single column key are automatically in at least 2NF, therefore all parameter tables, `radiometer_metsensors` table and `sounding_launches` table in the Wakasa database satisfy this requirement. In other tables, the primary key is composed of two columns, either `time_utc` and `paramset`, `obs_id` and `height` or `obs_id` and `frequency`. It can be easily seen that none of the other attributes in these tables is dependent on only a subset of the key and therefore all tables in the Wakasa database are in the second normal form.

The third normal form (3NF) is defined as *A relation that is in first and second normal form, and in which no non-primary-key attribute is transitively dependent on the primary key*. Transitive dependency is a condition where A, B and C are attributes of a relation such that if A $\rightarrow$ B and B $\rightarrow$ C, then C is transitively dependent on A via B. In the Wakasa database, there are a few transitive dependencies. Deviations from the third normal form are listed in Table 4.1. The relations could be normalized to 3NF by simply removing the offending columns.

The main reason for normalizing to 3NF is to avoid update anomalies: situations where updating one column and not updating another can put the database in an invalid state. However, the data in the Wakasa database is measurement results which are read from the measurement instruments, written once and not updated later. Only column in the data table which is likely to be modified at a later time is the `reliability` column which is not part of any transitive dependency. Therefore the deviations from the third normal form are not a big problem in the Wakasa database. In all cases except `rain_rate` in `balance` table, the same dependencies exist also in the original text files produced by the instruments.

Another update anomaly not addressed by the normalization process are dependencies between tables. In the Wakasa database, `rain_rate` column in `balance` table is fully functionally dependent on the primary key (`time_utc`, `paramset`) in terms of data insertion and retrieval, but the correctness of `rain_rate` values depends on the `box_area` value stored in `balance_parameters` table. Similarly, values in the `videodata` table

depend on the columns representing observation area dimensions and resolution in the videodata_parameters table. The only way to avoid this issue would be to not store these calculated values in the database and do the calculations after data retrieval. The current solution requires some extra attention from the database administrator when inserting data in the base, but is more convenient for the users.

Table 4.1: Deviations from the third normal form (3NF) in the Wakasa database

Table	Column	Description
balance	rain_rate	Transitively dependent on the set of columns stable_num, stable_accum, unstable_num and unstable_accum.
ceilo	alarm_status	Transitively dependent on column alarm_code.
ceilo	cloudbases	Transitively dependent on column detection_status.
ceilo	bs_sum	Transitively dependent on column bs (in the array version).
poss	precip_intensity_code	Transitively dependent on the set of columns precip_type and precip_rate.
videodata	several	The values actually measured by the instrument are time_utc, frames, frames_with_flakes, flakes_diameter, flakes_velocity, diameter_distribution and velocity_distribution. All other columns except paramset and reliability are transitively dependent on one or more of these columns.

General definitions of second and third normal forms extend the previously stated requirements from primary key to any candidate key in the tables. In Wakasa database, only tables which more than one candidate key are those which have (time_utc, paramset) as the primary key and obs_id column. In those tables, obs_id is also a candidate key. This does not change the situation, choosing obs_id as the primary key would give exactly the same result.

Boyce-Codd normal form (BCNF), fourth normal form (4NF) and fifth normal form (5NF) define some additional requirements for a relation. They are usually relevant only for situations with multiple column keys and complex dependencies between the data. In the Wakasa database, there are no relations which would be in third normal form but violate some of these stricter normal forms. The previously identified deviations from third normal form of course apply also to BCNF, 4NF and 5NF.

4.2 Inserting and Deleting Data

On the low level, all data manipulations in the database are performed using SQL language commands. The measurement instruments don't connect to the database directly but store their data in text files. A script called `insertdata.py` was written to read these files and feed the data in the base. The script also supports the concept of parameter sets described in Subsection 4.1.1. The administrator of the database writes an instrument specific parameter file which is given as a parameter for the script. An example parameter file is shown below:

```
# Parameter file for POSS data from Fukui Jan-Feb 2003
#
[wakasa2003]
location_latitude: 36.14
location_longitude: 136.22
location_utc_offset: +9
time_resolution: 60
```

It is not obligatory to specify values for all the parameters of the instruments. Most of the parameter fields are optional: if the value is missing, the corresponding column in the instrument parameters table in the database is marked as `NULL`. Missing values can be filled in later by adding them in the parameter file and calling `insertdata.py` again. However, to prevent common mistakes the script does not allow feeding in modified values with the same parameter set name. If the administrator notices an error in values already stored in the base, he must either update the values manually using SQL commands or delete and reinsert the data belonging to the erroneous parameter set.

Instruments supported by the `insertdata.py` script are listed in Table 4.2. The user specifies the type of instrument using a command line parameter when calling the script. The script also supports various other command line options, allowing for example inserting data of a limited time range. There is also another script `deletedata.py` which supports deleting data of the same instruments.

The core part of the data insertion script is a collection of parsers for the instrument specific data files. The parsers are the only place where it is necessary to know the detailed data file format of each instrument. After the data is in the base it can be retrieved using SQL queries or a variety of other generic tools.

Considerable effort was made to minimize device specific code and make it simple to add a new parser when a new instrument is acquired. For example the parameter set code is generic — the only change required for a new instrument is to add the new parameter table name, the list of columns in that table and which parameters are obligatory.

Table 4.2: Instruments supported by `insertdata.py`

Instrument	Device name	Description
Electronic balance	balance	Precipitation rate measured using an electronic balance, data in the format produced by the logger of WIS lab [20].
Optical lidar	ceilo	Vaisala Ceilometer CT25K optical lidar, data in the format produced by the logger of WIS lab.
Heat sensor	heatsensor_yamada	Yamada heat capacity sensor, data in csv format as produced by the device.
MRR-2 radar (processed data)	mrr	Metek MRR-2 radar, processed data in the format produced by the device.
MRR-2 radar (raw data)	mrrraw	Metek MRR-2 radar, raw data in the format produced by the device
POSS radar	poss	Andrew POSS bistatic radar, data in the format produced by the logger of WIS lab.
Radiometer WVR	radiometrics_wvr	Radiometrics WVR1100 radiometer, data in the `.los` files produced by the device.
Radiometer RPG	rpg_brt	Radiometer Physics RPG-LWP and RPG-TEMP90 radiometers, data in the binary format produced by the devices[4].
Vaisala soundings	radiosonde_vaisala	Vaisala radio sounding data, in the `.AED` and `.APA` files produced by the device.
Video based observation system	video	Video camera based observation system data, in the text format produced by the program used in the WIS lab.
Weatherstation AWS	weatherstation_aws	Data produced by the “AWS” automatic weather station, data in comma separated format (`.csv` files).
Weatherstation JWA	weatherstation_jwa	Data produced by the “JWA” automatic weather station, in the format produced by the logger of WIS lab.

[4]The parser for the RPG radiometer files is very slow in the current version. It will probably need to be rewritten if large amounts of RPG data will be handled.

4.3 Data Retrieval and Search Functionality

The SQL language provides efficient functions for retrieving and searching for data in the database. It is simple to select the desired columns of a table and add logical and comparison operators to limit the selection, for example to choose a time range. The syntax looks like English language and basic use is therefore easy to learn. Doing complicated queries efficiently can be tricky, but in most cases simple ones are sufficient. More complex arithmetic processing is then done outside the database using other tools.

A typical query retrieving video camera observation system data is shown below. The retrieval is limited to use only data from parameter set "`wakasa2003`".

```
  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;
```

More detailed information on SQL search capabilities can be found in any database related book (for example [34]) and does not belong to the scope of this thesis. However, two important limitations are discussed in the following sections.

4.3.1 Timestamp Synchronization

In the relational model, two tables can be joined based on a column which is common to both. For example, it is trivial to join timestamps from table `mrr` with height and reflectivity values from table `mrr_data`, because they have the same observation id. Joining is performed on demand when executing the query and the result is a combined virtual table.

In principle, timestamps can also be used to join tables between different devices. For example, the user might want to join the `poss` table with the `balance` table to compare POSS reflectivity with rain rate calculated using the electric balance. It is easy to plot a specific time range from both, but synchronizing individual observations is more difficult. If the timestamps are different even by only one second, the SQL join operation will not place the POSS values and balance values on the same row. Different height resolutions between devices pose similar problems.

One possibility to achieve synchronization would be to round timestamps when inserting data, for example to the nearest full minute. Values from instruments providing several values each minute could be averaged to form the result. The disadvantage is that time resolution of fast devices would be reduced, while slower instruments (not providing a result every minute) would still have either gaps or duplicated data. Also, it would be difficult to choose the best resolution because needs vary between different types of analysis.

Another possibility would be to not do any averaging and store all values of all devices, but still have "checkpoints" where the time stamps would be synchronized. This would allow joining tables at these checkpoint positions. The disadvantage of this is that adding new data becomes more complicated and errors in individual data points become more significant.

For example, let's consider a situation where one device records data every 10 seconds and the "checkpoints" are every 5 minutes. If one of the 10 second values is invalid, it may still end up representing the whole 5 minutes in the combined table.

After considering the alternatives it was decided to store timestamps as exact, leaving the synchronization problem to be resolved later. It is probably best to write scripts which go through all the values once doing the averaging at desired time resolution and write the results in new, temporary tables. These tables can then be joined easily. PostgreSQL also provides some SQL extensions to do arithmetic operations on columns, allowing timestamp comparisons such as "within 10 seconds". These at least partly solve the synchronization problem but are rather slow to be used regularly, at least for large amounts of data.

4.3.2 Continuous and Discontinuous Blocks

As mentioned previously, SQL allows various criteria to be placed on any column when selecting data. In the Wakasa database, users would often like to do queries such as "select all snowfall events with temperature being between -2 and 0 degrees". The temperature limit is easy to express in is SQL, but the result of the query will be a large number of rows, and it is not obvious which of them form continuous periods of time satisfying the criteria. The problem is illustrated by the example below, displaying an excerpt of weather station data (unnecessary columns omitted):

Time	Temperature (degrees Celsius)
2003-01-28 12:00:00	-2.1
2003-01-28 12:01:00	-2.0
2003-01-28 12:02:00	-2.1
2003-01-28 12:03:00	-2.1
2003-01-28 12:04:00	-2.0
2003-01-28 12:05:00	-1.9
2003-01-28 12:06:00	-1.9
2003-01-28 12:07:00	-2.0
2003-01-28 12:08:00	-2.1
2003-01-28 12:09:00	-2.1

Filtering the above data with the condition (`temperature >= 2.0`) will produce a set of five rows, where the first one is time-wise disconnected from the others:

Time	Temperature (degrees Celsius)
2003-01-28 12:01:00	-2.0
2003-01-28 12:04:00	-2.0
2003-01-28 12:05:00	-1.9
2003-01-28 12:06:00	-1.9
2003-01-28 12:07:00	-2.0

In many cases it would be nicer to receive just lists of starting and ending times of continuous periods where the given criteria is satisfied. This kind of behavior cannot be achieved

by using only SQL commands, but a script could compare the time difference of two successive rows. All instrument parameter tables contain a `time_resolution` field to make such analysis easier.

4.4 Expandability and Flexibility

The first goal of the database system was to provide storage for the data collected at the Wakasa experiment in Fukui, winter 2003. It will be used in the future for other measurement campaigns so flexibility and expandability was kept in mind during development. Inclusion of location information makes the system suitable for multi-site observations. There are no fundamental limits for scalability concerning the number of sites or the length of observation periods. However, the system has been designed for data from individual point locations, there is no direct support for map-like information such as radar or satellite images over a region.

Adding new measurement instruments is relatively easy. The procedure consists of creating a new parameters table and one or more data tables, and writing a parser for the file format of the new instrument. Alternatively data could be sent directly to the database without the need for an intermediate results file. However, many instruments are supplied with software which stores data in files. Unless real-time access is required, it is usually easier to write a converter for the files than to read values from the device directly.

Another likely future need is to store intermediate analysis results consistently. As the raw data is already in the database, it is the logical place for intermediate results as well. New either permanent or temporary tables can be created for these results. However, what is the best table structure in each case and which data should be stored in the first place is not an easy question. Creating new tables without careful planning and documentation quickly leads to outdated and unused datasets lying around and making the whole system more complicated to use.

When the database is used as a central data repository for several research groups, data mirroring and synchronization become important. Typical data access may involve retrieving dozens of megabytes of raw measurements, which essentially requires a local database server for each group. Solutions to automatically synchronize data between servers are available but they were not examined in detail within this project. At least in the development phase, the system was used only at two laboratories and data was transferred by manually copying the full database dump onto a portable hard disk.

Chapter 5

Snowfall Event Analysis

The database system is intended to facilitate algorithm development for snowfall analysis. The database project did not include research of any new algorithms, but a set of basic visualization tools was developed. These tools give a quick overview of the measurements. Plotting data from several instruments side by side and inspecting the graphs visually reveals already many characteristics of snowfall events.

5.1 System Characteristics

As mentioned earlier, the system contains data from point locations, in the current phase mainly from one single location: Fukui, Japan. Measurement instruments were installed at the ground and collecting data either from ground level or from a number of height steps up to a few thousand meters of altitude. Compared to most other observation systems the instruments operated at a high time resolution. When weather forecasts usually have only one or a few data samples per hour for a specific location, most instruments in the Wakasa experiment were operating at a time resolution of one minute or better. This makes it possible to distinguish between different phases of an individual snowfall event.

Clocks were synchronized to allow comparisons between different instruments. Data from multiple sources is essential for development and especially validation of new algorithms. The installation also produced several types of reference data which are usually not available in weather forecasting, such as detailed snowflake size and velocity distributions.

5.2 Visualization Tools

Several instruments used in the Wakasa observation campaign were shipped with some kind of visualization programs developed by the vendor. Unfortunately most of them had limited functionality, widely different user interfaces and were all device specific, making it difficult to do comparisons between instruments. Few were suitable for batch processing of long time ranges of data, some even crashed frequently. In any case, inserting the data in the database rendered the programs unusable because they were all delivered in binary format only and therefore couldn't be modified to use the database as their data source.

A new set of visualization tools were developed to provide basic graphs of the data of each instrument. The graphs can be used to quickly see what is going on and detect possible errors in the data already during an observation campaign. The same time period from many types of data and several instruments can be plotted using the same scale and compared visually.

The visualization tools were implemented as Python language [42] scripts using Gnuplot software [43] as the plotting backend. The scripts take a number of parameters from the user, compose the SQL query to select necessary data from the database, write the result to a temporary file and call Gnuplot to actually produce the graphical output. An example output is shown in Figure 5.1. More examples can be found in Appendix C.

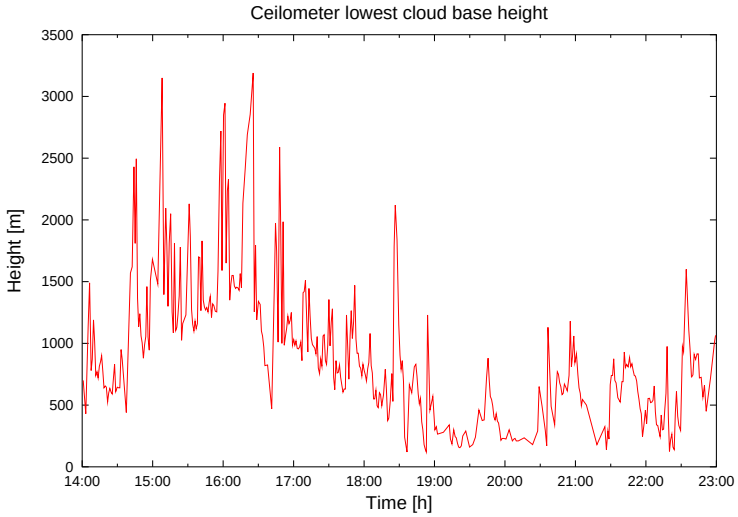

Figure 5.1: Example output of visualization script `plotceilo.py` showing lowest cloud base height on January 28, 2003. This plot was produced using the following command line: `./plotceilo.py -p wakasa2003 -d lowestcb -s "2003-01-28 14:00:00+09" -e "2003-01-28 23:00:00+09" -l --timeres=60 --font="Helvetica, 18"`

There is one script per instrument but most of the code is shared, minimizing programming effort and providing a consistent interface for the user. Time resolution and smoothing using moving average are selectable in all plots. It is also possible to redirect the Gnuplot data file and Gnuplot commands to an external file for further processing.

Current user interface is command line based. Besides interactive use, it is suitable for batch processing, for example automatically generating a set of plots for every day during a long observation period.

5.3 Analysis examples

As mentioned in Section 2.5, a rainfall and snowfall observation campaign was carried out at Fukui airport, near Wakasa Bay, Japan, from January 13 to February 6, 2003. The time

period included many types of precipitation. Snowfall and mixed precipitation events in late January were the most interesting for the research at the Image Information Science laboratory. The analysis of selected precipitation events is shown to present typical use of the database visualization tools. A similar analysis with different choice of examples has already been published in an earlier paper [44].

Figure 5.2 shows air temperature, snowfall rate measured by the electronic balance, POSS radar reflectivity and lidar optical backscatter on January 30, 12:00-20:00 Japanese time. Looking at the balance data, several precipitation events can be seen in the early afternoon between 12:30 and 15:00, and a single one in the evening after 19:00. Most of the events are small but there's a peak of heavy precipitation at 14:10. The precipitation is also reflected in the radar and lidar data.

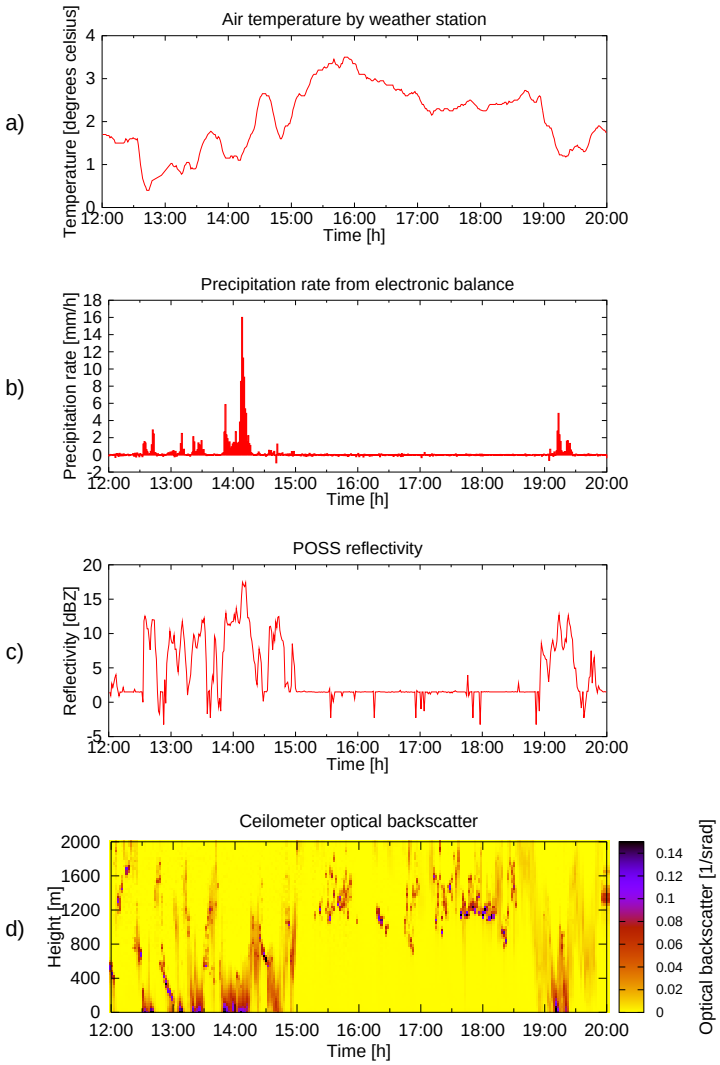

Figure 5.2: Measurements on January 30, 2003, 12:00-20:00 Japanese time: a) air temperature, b) precipitation rate, c) POSS radar reflectivity, d) optical lidar backscatter.

Temperature for the whole period is above zero, reaching a maximum of about 3.5 degrees Celsius in the afternoon. At the same time lidar data shows backscatter only high above 1000 meters and has gaps. It seems that during that period there was only a light cloud base and sun was warming the air. The most interesting part for analysis is the precipitation event with a sharp peak around 14:10. The temperature was only slightly above zero and raising, increasing the likelihood of change in the type of precipitation and composition of precipitation particles.

The video camera observation system data can give additional information about the type of precipitation. Figure 5.3 shows the velocity distribution of precipitation particles during three different 10 minute periods.

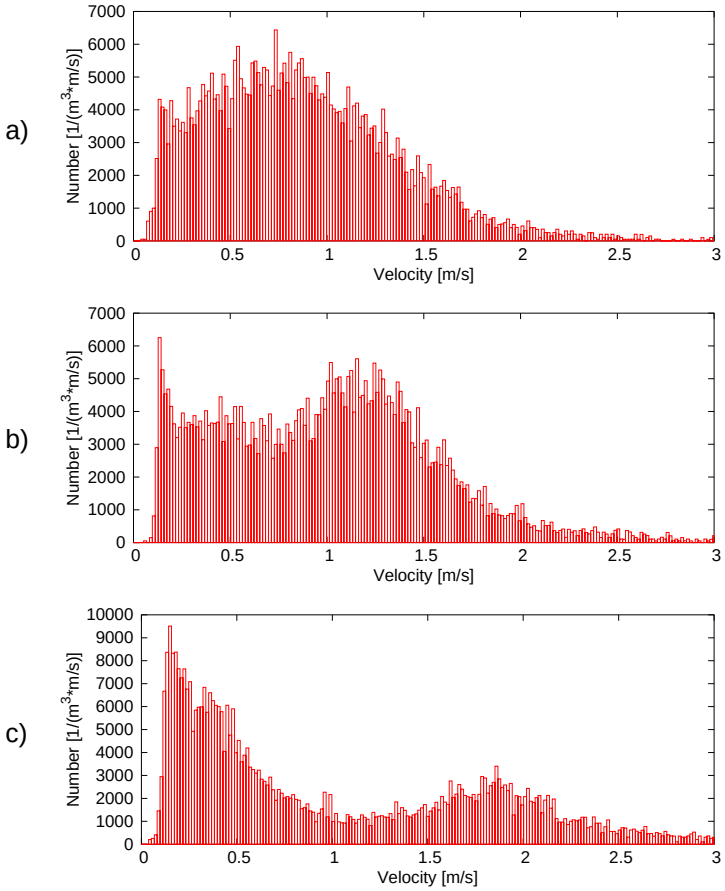

Figure 5.3: Precipitation particle velocity distribution on January 30, 2003 during three short time periods: a) 13:20-13:30, b) 13:50-14:00, c) 14:05:14:15.

The precipitation during 13:20-13:30 consists of particles with relatively low velocities, which signifies snowfall. During 13:50-14:00 there is a larger range of velocities; taking account the rising temperature it seems likely that snowflakes are slowly turning into sleet. The short period with intense precipitation during 14:05-14:15 shows both low velocities and a second peak around 2 m/s. This signifies mixed precipitation with two different types of particles, probably graupel and snowflakes.

Diameter and velocity distributions during a longer period are interesting statistical values. Figure 5.4 shows the distributions calculated from the whole Wakasa observation period. The data includes only sleet, snowfall and graupel events; the system was not operated during rainfall.

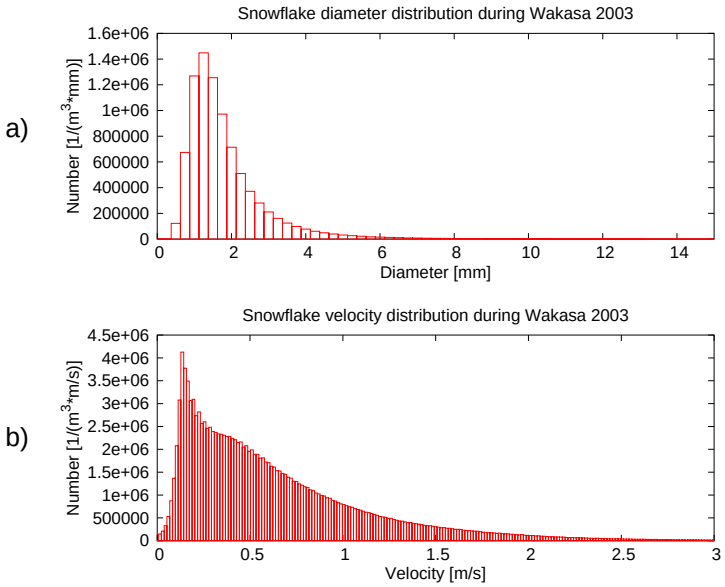

Figure 5.4: Precipitation particle distributions during the Wakasa observation, January-February 2003: a) diameter distribution, b) velocity distribution.

The diameter distribution looks normal, but the velocity distribution has a strange peak at approximately 0.15 m/s. The same peak is also partly visible in the graphs of Figure 5.3. It doesn't correspond to any long term snowfall statistics and also deviates from earlier results obtained using the same system [8]. Therefore it seems that the setup suffered from some kind of noise problem during the Wakasa observation. The data is not completely useless, but special care must be taken when using it. Lowest velocities will probably have to be discarded or bias-corrected.

5.4 Suitability for Statistical Analysis

Many modern signal processing, pattern recognition and classification methods are based on statistical analysis of the input data. For such analysis it is a major advantage when the data is in a consistent format in the database instead of scattered text files. There is no need to write a parser to extract the necessary data from the instrument file format. Multiple years of data can easily be processed in one run.

More importantly, the need for error handling is reduced. Often the longest time when writing a parser is spent in small details, situations where the instrument produces an incomplete or otherwise unexpected data line, for example containing some kind of error code due to temporary malfunction. With the database, these problems have been encountered when

writing the initial parser and missing or invalid data is consistently marked with the special value NULL.

Accessing the database directly is a good idea when the researcher is writing his or her own analysis programs. General purpose spreadsheets and scientific computing packages cannot usually access the database directly, they require the data in a regular file. This is not a problem because data can be trivially exported to comma separated format (csv) files which are supported by most programs. Also in this case, the ability to easily select the relevant data and reduced need for error handling are clear advantages of the database.

The database gives interesting possibilities for producing generic statistics without even knowing the exact type of data. For example, mean value and standard deviation could be calculated daily for each measurement data column. Visual weather observations which are currently recorded by hand into a logbook could also be better taken advantage of. Using standard Synop "present weather" codes [45] and feeding them into the database would give a good foundation for developing classification algorithms. There was not enough time to implement these improvements in the scope of this project, but they may be added in the future.

5.5 User Experiences

During the prototype phase, there were three users of the database system. Professor Muramoto and B. Sc. student Hideki Aoyama from the WIS lab [20] used the visualization tools to prepare an analysis of snowfall events for the Wakasa workshop in May 2003, including a set of graphs to be distributed for other participants. M. Sc. student Thomas Pfaff from the River and Environmental Engineering Laboratory, University of Tokyo [24] accessed the database directly with SQL queries to retrieve measurement values for his weather model. He also participated in the design of the tables and contributed some parts of parser code. The work of Mr. Pfaff is described in more detail in his thesis [46].

Due to the small number of users, a comprehensive summary of user experiences cannot yet be done. Several features have received only very limited testing, in particular the support for multiple observation sites. Still, bug reports, questions, and comments from users were encouraging and important during the development of the system. The database is currently being used in Kanazawa University by a few students. Hideki Aoyama's work is described in more detail in chapter 7.

Chapter 6

System Performance

In the Wakasa database project, high raw system performance was not one of the key objectives. The design goals were ease of use, standards compliance and simple structures without sacrificing extendibility. During the project some performance issues surfaced and were addressed by making a small change in the table structure as described in Section 4.1. The performance of these two alternative implementations, standard version and array version, is compared in this chapter.

6.1 Performance Requirements

A single Wakasa database installation is not expected to have a large number of simultaneous users. However, it might have a considerable amount of data, in the order of tens of gigabytes. The data is usually inserted only once and not modified during usage, so writing performance is not very important. When reading, both finding individual data elements at random times and exporting large continuous datasets should be acceptably fast. All usage patterns in future research cannot be foreseen, but already for basic visualizations there are two distinct types of use which have different performance requirements.

If plots are generated for a long period of time in a batch processing manner, a response time of several minutes for certain plots is accessible. A good basic criteria is that the system should be able to process the data of one winter during one night. On the other hand, if the system is used for interactive plotting, for example through a web interface, it should be able to process any query in a few seconds or at least under one minute.

In any type of use, the system should be scalable to include data from several years and several observation locations without significant loss of performance in plots and and queries of fixed time periods. Also, exporting continuous datasets of several days out of the database to external files should be reasonably fast, taking at the most a few minutes.

6.2 System Setup

6.2.1 Test Data

The data available at the time of development was the data collected during the Wakasa observation campaign in Fukui, January-February 2003. For some instruments there was

data from several previous years, but for performance testing it was better to have an equal time range of data from all instruments. The measurements taken between January 21, 00:00:00 JST 2003 and January 30, 23:59:59 JST 2003 were chosen as the basic test set. Table 6.1 lists the included instruments and data set sizes for the chosen 10-day period.

Table 6.1: Data set selected for performance testing: Measurements recorded during January 21 – January 30, 2003.

		Standard version		**Array version**	
Instrument[1]	**File size MB**	**Database size MB**	**Number of rows**	**Database size MB**	**Number of rows**
Electronic balance	3.7	9.5	86397	9.5	86397
Heat sensor	0.2	0.2	1680	0.2	1680
Optical lidar	20.1	636.2	7393376	33.2	28768
MRR-2 radar (processed)	504.2	444.1	892676	179.0	28796
MRR-2 radar (raw)[2]	1007.8	463.9	1807080	130.0	54760
POSS radar	3.9	3.8	14400	3.8	14400
Radiometer WVR1100	2.1	8.6	81484	8.6	81484
Radio sounding[3]	0.6	1.0	6460	1.0	6460
Video observation system[4]	21.2	6.6	6442	6.6	6442
Weatherstation AWS	0.3	0.8	7126	0.8	7126
Total	1564.1	1574.7	10300121	372.7	319313

As can be seen from the table, the data set disk space requirements for a particular instrument vary largely depending whether the data is stored in the original text or binary files or in the database. In most cases, using a database increases the required disk space, which can be explained by the need of search indexes and data types which take a larger amount of space to store. However, in some cases, MRR-2 radar in particular, the original data file format is space inefficient and the database format is more compact. In the case of video based observation system data, only a subset of the recorded values for each timestamp were inserted into the database.

As explained in Section 4.1, the standard and array versions of the database differ only in the case of optical lidar and MRR-2 radar data. The space savings in the array version are significant. In particular, it can be noted that storing height data on separate rows as in standard version is very inefficient in the case of optical lidar. However, considering current hard disk prices the disk space requirements are not a problem in either version, because all the measurement data for one whole winter can be stored in less than 10 gigabytes.

[1] More detailed description of the instruments and the data types is in Table 4.2.

[2] Due to a time zone error, the MRR-2 raw data in the test data set was from a time window starting and ending 9 hours earlier than other instruments. However, the data set size is fully comparable.

[3] Radio sounding data was not continuous, the data included in the test set consisted of 34 sounding launches.

[4] Video observation system was operated only during precipitation and therefore continuous data was not available for the whole 10 day period. The data set included consists of approximately 116 hours (4.8 days) of data.

6.2.2 Hardware and Software Environment

All performance tests were run on a computer which was dedicated as the database server. The hardware is summarized in Table 6.2.

Table 6.2: Database server hardware

Component	Model / Amount
Processor	Intel Pentium III 800 MHz
Memory	1024 MB
IDE controller	VIA vt82c596b on-board UDMA66 controller
SCSI controller	Adaptec 29160 Ultra160 SCSI adapter
Operating system disk	20 GB 5400 rpm IDE disk (Quantum Fireball lct10)
Database disk	40 GB 5400 rpm IDE disk (Seagate ST340810A) or 72 GB 10000 rpm SCSI disk (HitachiDK32DJ-72MW)[5]

DMA transfers, multiple sector I/O (16 sectors) and read-lookahead were enabled for the IDE disks. The operating system was GNU/Linux and the software versions relevant to this project are listed in Table 6.3.

Table 6.3: Database server software

Program name	Version
Vine Linux distribution	2.6
Linux kernel	2.4.19
GNU libc	2.2.4
GNU C compiler	2.95.3
PostgreSQL database server	7.2.3
Python programming language	2.2.2
Numeric Python extension	23.0
Python PostgreSQL interface pyPgSQL	2.3
Gnuplot plotting tool	3.8j beta[6]

All software was either installed from rpm packages or compiled using default settings and not specially tuned. During the performance tests, the computer was running normal system daemons in addition to the PostgreSQL database but it was not used for other purposes.

[5] Initially, the machine was equipped with a 40 GB IDE disk for the database. However, later this disk was broken and replaced by the 72 GB SCSI disk. Most performance measurements were done using the SCSI disk, but some data insertion benchmarks were performed with both configurations.

[6] The Gnuplot plotting tool was compiled from source code which was downloaded directly from the project cvs repository on July 11, 2003.

Operating system, home directories and source data files were placed on the operating system disk, database data files on the database disk. All tests were carried out locally without using network connections for anything else than remotely logging in and starting the test.

Most of the measurements presented in this chapter were obtained by running the database engine with default settings. Before the final set of visualization tool and data export performance tests a few settings were adjusted as advised in a PostgreSQL performance tuning guide on the web [48]. `Shared_buffers` value was increased from 64 to 2048 (16 MB), `effective_cache_size` value from 1000 to 89600 (500 MB) and `sort_mem` value from 512 to 4196 (4MB / query).

6.2.3 Main Factors Affecting Performance

In database based applications, design choices and sometimes even small implementation details may have a significant impact on performance. In the Wakasa database project, the following key factors were considered.

1. Modeling array and matrix data. As described in Subsection 4.1.2, there are several possibilities of storing array and matrix data in the database. Difference in performance can be more than one order of magnitude, depending on the number of rows in the database and the type of query.

2. Database engine. Different database engines can have significantly different performance figures for a given application.

3. Programming language and database interface. Performance bottlenecks may not be in the database engine itself, but rather in the database interface of a chosen programming language, or some completely database unrelated part of the application.

4. SQL queries and database indexes. Certain types of SQL queries can lead to complex and inefficient database operations. Sometimes these can be speeded up considerably by manually giving a command to create indexes inside the database for the affected columns.

5. Server hardware and operating system. Obviously, database server hardware and operating system also affects performance. In most cases, memory and disk performance are more important in database applications than processor speed.

In Wakasa project, several methods to model array and matrix data were used, and analyzed in the results. PostgreSQL was selected as the database engine because of it was open source and had a reputation of being a mature system. It would have been interesting to try different engines, but there was no time for that within the scope of the project. For programming, Python language was chosen because of convenience and speed of implementing the desired features, although lower level languages as C were expected to give shorter execution times. Query optimization and indexes were not studied in detail because the visualization tools needed only relatively simple queries. However, this topic may become more important in future research.

Comparing the performance of different hardware was not the purpose of the project, so a standard computer which was already available at the laboratory was chosen as the database

server. When doing performance measurements, both the raw data, database and visualization tools were located on the server. Therefore, all invocations of the Python interpreter and the Gnuplot plotting tool also happened locally. It would be possible to balance the load by running Python and Gnuplot on separate computer(s) and use the network to retrieve values from the database. This was verified to work but the performance of such setup was not measured.

During the tests one surprising factor came up which had a very significant impact on the performance. Directly after insertion of data PostgreSQL was using slow sequential searches in data retrieval even when table keys were used to select the desired columns. Queries produced correct results but the performance was poor. The behavior was corrected by running the command `vacuumdb --analyze` which initialized PostgreSQL into using indexed searches. Results both before and after vacuuming are presented for comparison in Subsections 6.3.2 and 6.3.3. They show clearly how small details can play a groundbreaking role in performance.

6.3 Results

6.3.1 Inserting Data

The performance of inserting data in the database was tested using the `insertdata.py` script, described in Section 4.2. The database table structure was created from scratch, data of all instruments inserted, after which the whole database was erased and the process started all over again. Always after the insertion of data from a single instrument was complete, the write cache was flushed to the disk and the timestamp recorded.

For the array version of the database, the insertion and deletion process was repeated 10 times. For the standard version (and also array version in the case of MRR-2 radar) it was performed only 3 times because of the long duration of the test run. The order of instruments in the insertion script was fixed, but the insertion process was also tested in reverse order. No significant differences in performance were found due to the change in insertion order.

The results of data insertion performance are presented in Table 6.4. The confidence ranges were calculated using standard deviation and assuming normal distribution; in the case of tests which were performed only 3 times the highest deviation from average value is used. All measured times are wall-clock time.

Insertion time becomes very long in the standard version, taking over five days to complete. The time is highly dominated by optical lidar and MRR-2 radar data. This problem was addressed in the array version of the database, where the whole insertion task takes less than two hours. The results remained consistent during tests, deviations were small. The small differences between standard and array versions for other instruments than optical lidar or MRR-2 radar can be explained by the fact that the data ends up being stored on different areas on the hard disk.

During the data insertion tests, the database was stored on a 5400 rpm IDE hard disk. Later it was replaced with a 10000 rpm SCSI unit because of a disk failure. A surprising observation was that data insertion was significantly slower when using the SCSI disk. Due to time constraints a full test run was not done with the SCSI disk, but the insertion times approximately doubled. The result is probably due to poor SCSI I/O performance in the Linux

Table 6.4: Performance of inserting data. All execution times in seconds.

Instrument	Time minutes (standard version)	Time minutes (array version)
Electronic balance	5.62 ± 0.02	5.58 ± 0.03
Heat sensor	0.12 ± 0.00	0.11 ± 0.00
Optical lidar	5221.41 ± 40.81	7.00 ± 0.01
MRR-2 radar (processed)	523.54 ± 1.22	29.26 ± 0.07
MRR-2 radar (raw)	1872.69 ± 4.52	38.22 ± 0.03
POSS radar	1.11 ± 0.00	1.09 ± 0.01
Radiometer WVR1100	30.17 ± 0.14	30.40 ± 0.08
Radio sounding	0.48 ± 0.00	0.47 ± 0.00
Video observation system	0.68 ± 0.01	0.64 ± 0.00
Weatherstation AWS	0.48 ± 0.00	0.48 ± 0.01
Total	7656.30 ± 46.72	113.25 ± 0.24

kernel. Such a big variation suggests that large speed gains may be possible by adjusting SCSI driver parameters or upgrading the kernel to 2.6 series which includes a rewritten SCSI I/O layer.

The results presented here measure the initial insertion of data using the `insertdata.py` script. Later, if the database suffers a disk failure or the data has to be transferred to another machine, the database can be reconstructed from backups. Restoring the data from a backup is a lot faster than the initial insertion. For the test data set, restoring data from a backup takes about 7 minutes in the standard version and 13 minutes in the array version.

Deleting data is a very uncommon operation in the Wakasa database. Usually, the only reason to delete data is a mistake in insertion parameters. The time range and type of necessary deletion therefore varies largely depending on the situation. The performance of data deletions was not measured but in any case it is faster than insertion and sufficient for all normal use cases.

6.3.2 Visualization Tools

The performance of visualization tools was measured by selecting a set of common graphs and producing them for different time ranges starting at random timestamps. The selected graphs are introduced in Table 6.5. A representative subset of the results is presented in this section. Full details including more graph types and time ranges, command lines and equivalent SQL queries can be found in Appendix B.

Graphs `balance`, `heat`, `lidar1` and `poss` are simple two-dimensional graphs, where the values can be obtained from two columns (timestamp and the measured value) of a single table in the database. Graphs `lidar2`, `lidar3`, `mrr`, `video1` and `video2` include retrieval of array values. In graphs `video1` and `video2` the values are stored as text blocks, in graphs

Table 6.5: Graph types used for visualization tool performance measurements.

Graph name	Instrument	Type of visualization
balance	Electronic balance	Snowfall rate
heat	Heat sensor	Snow detection flag
lidar1	Optical lidar	Lowest cloud base height
lidar2	Optical lidar	Integrated backscatter 0–3000 m
lidar3	Optical lidar	Backscatter map 0–3000 m, 30 m height resolution
mrr	MRR-2 radar	Reflectivity map 0–3000 m, 120 m height resolution
poss	POSS radar	Integrated reflectivity
video1	Video observation system	Snowflake diameter distribution averaged over time
video2	Video observation system	Snowflake diameter distribution against time (map)

`lidar2`, `lidar3`, `mrr` they are either splitted on several rows (standard version) or stored as PostgreSQL arrays (array version).

Lidar data arrays consist of 256 elements of which 100 are retrieved for height range 0–3000m, mrr arrays consist of 30 elements of which 25 are retrieved, video arrays consist of 200 elements and are retrieved in entirety. In `lidar2` and `video1` the values are added together or averaged to produce a simple line graph, whereas the output of `lidar3`, `mrr` and `video2` is a two-dimensional color map. Within pairs (`lidar2`, `lidar3`) and (`video1`, `video2`) the queries for retrieving data from the database are actually identical, only the type of output is different.

All graphs were plotted as encapsulated post script (eps) files using time resolution of one minute. If the time resolution of raw data was different, the necessary transformation to one minute was done by the visualization script. The time ranges used were 4 minutes, 15 minutes, 1 hour, 4 hours, 16 hours and 24 hours. Each of the graphs for each time range was plotted 30 times with a starting time selected at random from the test data set. The measured execution times include parsing the input parameters, fetching the necessary data from the database and producing the output file. The time needed for randomizing the starting time and constructing the command line was excluded from the measurement.

First results obtained by running the visualization scripts right after inserting data are presented in Tables 6.6 and 6.7. The confidence ranges were calculated using standard deviation and assuming normal distribution. All measured times are wall-clock time.

Very long execution times for plotting even short time ranges of lidar and mrr data in the standard version hinted that the system was not functioning properly. Tuning configuration parameters as mentioned in Subsection 6.2.2 improved the performance about 10 percent but not significantly. Finally the problem was identified by using the PostgreSQL command `EXPLAIN` which shows how queries are being executed. It turned out that the system was going through all table rows sequentially to retrieve the necessary data. Vacuuming the

Table 6.6: Performance of visualization tools, standard version of the database, system not functioning properly. All execution times in seconds.

Graph	15 min	1 h	4 h	16 h
balance	0.58 ± 0.02	0.67 ± 0.02	1.08 ± 0.02	2.63 ± 0.02
heat	0.25 ± 0.04	0.26 ± 0.04	0.28 ± 0.04	0.33 ± 0.01
lidar1	0.41 ± 0.02	0.42 ± 0.01	0.49 ± 0.04	0.81 ± 0.12
lidar2	39.59 ± 0.20	43.51 ± 0.21	58.64 ± 0.25	119.63 ± 0.32
lidar3	39.86 ± 0.22	44.17 ± 0.24	61.53 ± 0.26	131.25 ± 0.31
mrr	8.76 ± 0.01	9.21 ± 0.02	10.93 ± 0.07	17.98 ± 0.14
poss	0.35 ± 0.00	0.37 ± 0.00	0.46 ± 0.00	0.81 ± 0.00
video1	0.41 ± 0.04	0.46 ± 0.04	0.67 ± 0.08	1.48 ± 0.24
video2	0.61 ± 0.08	1.25 ± 0.26	3.55 ± 1.29	14.70 ± 3.05

Table 6.7: Performance of visualization tools, array version of the database, system not functioning properly. All execution times in seconds.

Graph	15 min	1 h	4 h	16 h
balance	0.58 ± 0.02	0.67 ± 0.02	1.07 ± 0.02	2.62 ± 0.02
heat	0.25 ± 0.04	0.27 ± 0.04	0.26 ± 0.04	0.33 ± 0.01
lidar1	0.69 ± 0.02	0.71 ± 0.03	0.78 ± 0.04	1.07 ± 0.14
lidar2	1.19 ± 0.02	2.71 ± 0.05	8.82 ± 0.17	33.19 ± 0.41
lidar3	1.37 ± 0.02	3.41 ± 0.05	11.55 ± 0.20	44.29 ± 0.39
mrr	0.70 ± 0.00	0.97 ± 0.00	2.04 ± 0.02	6.32 ± 0.04
poss	0.35 ± 0.00	0.37 ± 0.00	0.46 ± 0.00	0.81 ± 0.00
video1	0.41 ± 0.03	0.47 ± 0.02	0.68 ± 0.07	1.47 ± 0.24
video2	0.56 ± 0.12	1.22 ± 0.30	3.37 ± 1.15	13.06 ± 2.47

database as described in Subsection 6.2.3 turned the system into using indexed searches as expected. Results after vacuuming are presented in Tables 6.8 and 6.9. The configuration parameter tunings were also in place when running these tests.

Simple graphs (`balance`, `heat`, `lidar1`, `poss`) are all processed in reasonably short time. The heat sensor table is very small due to coarse time resolution of the instrument (5 minutes), so we can see that the minimum amount of time to produce a graph is around 0.24 seconds. This includes launching the visualization script, establishing a database connection, retrieving some data and producing a plot to an encapsulated postscript file. For devices with higher time resolution (e.g. `balance` 10 seconds) this time is slightly longer, but not significantly.

When longer time ranges are retrieved, the execution time increases approximately linearly and the amount of data becomes the dominating factor. For example, the time to produce a graph for `balance` is approximately 0.28 seconds + 0.13 seconds per hour, whereas the time to produce a poss graph is roughly 0.28 seconds + 0.03 seconds per hour.

Table 6.8: Performance of visualization tools, standard version of the database, after vacuuming. All execution times in seconds.

Graph	15 min	1 h	4 h	16 h
balance	0.31 ± 0.00	0.41 ± 0.02	0.82 ± 0.01	2.38 ± 0.01
heat	0.24 ± 0.04	0.26 ± 0.05	0.26 ± 0.05	0.34 ± 0.02
lidar1	0.31 ± 0.02	0.34 ± 0.01	0.40 ± 0.05	0.71 ± 0.09
lidar2	1.44 ± 0.02	4.75 ± 0.05	18.07 ± 0.14	104.55 ± 1.57
lidar3	1.62 ± 0.02	5.44 ± 0.04	20.87 ± 0.13	116.10 ± 0.91
mrr	0.48 ± 0.02	0.93 ± 0.04	2.62 ± 0.07	9.49 ± 0.16
poss	0.29 ± 0.00	0.31 ± 0.00	0.40 ± 0.00	0.76 ± 0.00
video1	0.33 ± 0.04	0.39 ± 0.05	0.62 ± 0.06	1.35 ± 0.24
video2	0.48 ± 0.14	1.12 ± 0.30	3.19 ± 1.31	12.53 ± 2.83

Table 6.9: Performance of visualization tools, array version of the database, after vacuuming. All execution times in seconds.

Graph	15 min	1 h	4 h	16 h
balance	0.31 ± 0.00	0.41 ± 0.00	0.82 ± 0.00	2.38 ± 0.01
heat	0.24 ± 0.04	0.25 ± 0.05	0.27 ± 0.05	0.34 ± 0.02
lidar1	0.31 ± 0.02	0.33 ± 0.02	0.40 ± 0.04	0.74 ± 0.11
lidar2	0.82 ± 0.01	2.31 ± 0.05	8.44 ± 0.21	32.79 ± 0.43
lidar3	0.99 ± 0.01	3.02 ± 0.05	11.20 ± 0.19	44.03 ± 0.45
mrr	0.41 ± 0.00	0.67 ± 0.00	1.75 ± 0.02	6.09 ± 0.05
poss	0.29 ± 0.00	0.31 ± 0.00	0.40 ± 0.00	0.75 ± 0.00
video1	0.33 ± 0.04	0.39 ± 0.05	0.59 ± 0.10	1.43 ± 0.21
video2	0.52 ± 0.10	1.15 ± 0.30	3.73 ± 0.68	13.00 ± 2.65

When graphs contain array values, data organization and type of output have both a significant impact on the performance. When the system was using sequential search, visualizations of even short time ranges of `mrr` and `lidar` data in the standard version took up to 40 seconds on the test machine (Table 6.6). When operating normally using indexed search the time drops to about one second. The standard version is still slightly slower than the array version, but the difference is small for short time ranges. For long time ranges the array version is faster by a large margin due to the smaller number of rows to process.

Pairs (`video1`, `video2`) and (`lidar2`, `lidar3`) show the performance difference between producing a line graph and a color map graph. In both cases, the color map takes about 11-13 seconds more time than a line graph when visualizing a time range of 16 hours. This accounts for most of the execution time in the case of video data, whereas in the case of lidar intermediate data processing remains the most time consuming part. Organizing data as text blocks (`video`) or using PostgreSQL arrays (`lidar, mrr`) seem to give roughly equivalent performance taking account the differences in time resolution and array size.

One value which sticks out of the results is lidar visualization performance for the 16 hour time range in the standard version of the base. Closer examination revealed that the database reverted to sequential search for some reason when retrieving a large total number of rows from the table, leading to longer time than four 4 hour retrievals. It might be that if such queries are frequent and `vacuumdb --analyze` command is run again the system could optimize them better, but that was not tested.

6.3.3 Exporting Data

Exporting data to comma separated value (csv) format was tested using the `psql` command line tool which is included in PostgreSQL database distribution. The goal was to simulate the data retrieval cases of the visualization graphs as closely as possible, and to find whether possible performance bottlenecks would be in database engine operations or in the visualization scripts.

The data export performance measurements were done using SQL queries equivalent to data retrieval in graphs of Table 6.5. The query of the graph `lidar2` is identical to the query of `lidar3` so only results for `lidar2` is presented in the results. Similarly, the query for graph `mrr2` is identical to that of `mrr1` and the query for `video1` is identical to that of `video2`. The visualization tools do a couple of additional queries to retrieve the number of parameter set and time zone of the instrument location, but they are insignificant for the total performance of the system when retrieving more than a few minutes worth of measurement data at one time.

As in visualization tool measurements, the time ranges used were 4 minutes, 15 minutes, 1 hour, 4 hours, 16 hours and 24 hours.. Each time range was exported 30 times with a starting time selected at random from the test data set. The measured execution times include starting the psql tool and writing the output in a text file. The time needed for randomizing the starting time and constructing the command line was excluded from the measurement.

The results are presented in Tables 6.10 through 6.13. Full details can be found in Appendix B. The confidence ranges were calculated using standard deviation and assuming normal distribution. All measured times are wall-clock time.

Table 6.10: Performance of exporting data, standard version of the database, system not functioning properly. All execution times in seconds.

Graph	15 min	1 h	4 h	16 h
balance	0.32 ± 0.02	0.32 ± 0.02	0.35 ± 0.02	0.46 ± 0.02
heat	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.05 ± 0.00
lidar1	0.14 ± 0.01	0.14 ± 0.01	0.15 ± 0.01	0.17 ± 0.01
lidar2	40.18 ± 0.27	41.23 ± 0.27	44.67 ± 0.30	59.67 ± 0.59
mrr	10.72 ± 0.06	10.85 ± 0.10	11.14 ± 0.08	12.44 ± 0.06
poss	0.11 ± 0.00	0.11 ± 0.00	0.11 ± 0.00	0.13 ± 0.00
video1	0.12 ± 0.00	0.12 ± 0.00	0.12 ± 0.00	0.15 ± 0.03

Table 6.11: Performance of exporting data, array version of the database, system not functioning properly. All execution times in seconds.

Graph	15 min	1 h	4 h	16 h
balance	0.31 ± 0.02	0.32 ± 0.02	0.34 ± 0.02	0.47 ± 0.01
heat	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.05 ± 0.00
lidar1	0.42 ± 0.01	0.42 ± 0.01	0.42 ± 0.01	0.45 ± 0.01
lidar2	0.43 ± 0.01	0.48 ± 0.01	0.66 ± 0.01	1.46 ± 0.02
mrr	0.33 ± 0.00	0.34 ± 0.00	0.36 ± 0.00	0.43 ± 0.00
poss	0.11 ± 0.00	0.11 ± 0.00	0.11 ± 0.00	0.13 ± 0.00
video1	0.12 ± 0.00	0.12 ± 0.00	0.12 ± 0.00	0.15 ± 0.00

Table 6.12: Performance of exporting data, standard version of the database, after vacuuming. All execution times in seconds.

Graph	15 min	1 h	4 h	16 h
balance	0.04 ± 0.01	0.05 ± 0.01	0.13 ± 0.12	0.22 ± 0.00
heat	0.03 ± 0.00	0.03 ± 0.00	0.04 ± 0.00	0.04 ± 0.00
lidar1	0.04 ± 0.00	0.04 ± 0.00	0.09 ± 0.06	0.14 ± 0.01
lidar2	0.24 ± 0.03	0.67 ± 0.04	2.44 ± 0.13	44.92 ± 1.30
mrr	0.09 ± 0.02	0.17 ± 0.03	0.43 ± 0.05	1.49 ± 0.17
poss	0.04 ± 0.01	0.04 ± 0.00	0.06 ± 0.03	0.10 ± 0.00
video1	0.04 ± 0.00	0.04 ± 0.00	0.05 ± 0.00	0.09 ± 0.01

Table 6.13: Performance of exporting data, array version of the database, after vacuuming. All execution times in seconds.

Graph	15 min	1 h	4 h	16 h
balance	0.05 ± 0.02	0.05 ± 0.01	0.10 ± 0.07	0.22 ± 0.00
heat	0.03 ± 0.00	0.03 ± 0.00	0.04 ± 0.00	0.04 ± 0.00
lidar1	0.04 ± 0.00	0.04 ± 0.01	0.06 ± 0.01	0.27 ± 0.20
lidar2	0.05 ± 0.00	0.10 ± 0.00	0.28 ± 0.01	1.27 ± 0.18
mrr	0.04 ± 0.01	0.05 ± 0.01	0.07 ± 0.01	0.16 ± 0.03
poss	0.04 ± 0.00	0.04 ± 0.00	0.06 ± 0.02	0.10 ± 0.00
video1	0.04 ± 0.00	0.04 ± 0.00	0.05 ± 0.03	0.10 ± 0.01

As expected, exporting data using `psql` is considerably faster than plotting it with the visualization tools. Otherwise the performance characteristics are similar to visualization tool results. The difference in performance between the initial sequential search operation (Ta-

bles 6.10 and 6.11) and normal indexed search operation (Tables 6.12 and 6.13) becomes even more apparent now. Again, lidar data retrieval of long time ranges in the standard version of the base (Table 6.12, graph lidar2, 16 h) can be noted as a similar strange exception than in visualization tool results.

6.4 Result Analysis and Scalability Estimates

In data insertion, the performance varied between 16 and 258 row insertions per second depending on the type of instrument, table structure and amount of data. Inserting lidar and mrr data in the standard version was slowest: the number of rows was large and even the time per row was longest. Within groups of instruments with similar table layout (e.g. balance, heat sensor, poss, weather station being one group, lidar and mrr another) the insertion time of one row stayed constant and total time grew linearly with the number of rows, indicating complexity O(n).

Data insertion is normally done only once, so very high performance is not necessary. Spending two hours for inserting a 10 day data set as in the array version is completely acceptable. Even if the number of measurement locations is increased considerably, this will not become a bottleneck. For example, inserting an equal amount of data from 100 locations would take 8.3 days, still shorter than the time period of the dataset itself. On the other hand, insertion times in the standard version are too long for convenient use and leave little margin for processing larger amounts of data.

Data retrieval and visualization performance is more important. Retrieving and visualizing short time ranges is fast enough for all instruments. For longer time ranges, performance of lidar visualizations is relatively poor in the standard version of the base. Exporting or visualizing long time periods increases processing time linearly, with complexity O(n). Using PostgreSQL arrays cuts the times to about a half but does not change the complexity. This is not a big problem, interactive use is expected to concentrate on time periods up to one day and longer processing over months or years will be done as batch runs. The main bottleneck is the processing of data structures using Python. Database load is low even during longer plots so the system would become more responsive by optimizing the visualization tools or simply adding more processing power to the client computers. For most use, the current performance is completely sufficient.

New research projects in future may need to execute large numbers or complex SQL queries to retrieve data from the database. The performance of such queries has not been tested and remains unknown. If the queries use comparison operators or sorting on unindexed columns, manually demanding PostgreSQL to index those columns may improve the performance. Query ordering can also make a big difference, PostgreSQL does not seem to have very advanced automatic optimization algorithms, at least the version used in the Wakasa project.

The discovery that the system was using sequential searches before vacuuming showed how small details can have a decisive role in performance. PostgreSQL seems also to be picky about query syntax, for example not using quotes around values can produce correct result but again revert from indexed search to slow sequential search. The user has to be careful and detect incorrect behavior. With the current amount of data the system would have been usable even with sequential searches, but it would have become a larger and larger problem in the future as more data will be added.

The performance measurements were done using a relatively small data set, each table was small enough to fit in the 1 GB memory of the database server. Performance may suffer if the tables grow larger than the main memory size, but it should not be a problem as long as the indexes still fit in the memory. Also, the system was only tested with one user, stress testing with multiple concurrent users would give better picture of the performance under heavy loads. However, the database is not foreseen to be used by more than a few simultaneous users and most of the load in time-consuming queries was on the visualization tools which are normally executed at the client. Therefore no major scalability problems are expected to appear at least in the near future.

Chapter 7

Application to Analysis of Relationship Between Radar Reflectivity Factor and Snowfall Rate

During winter 2004, B. Sc. student Hideki Aoyama studied the relationship of POSS radar reflectivity factor and snowfall rate using the Wakasa observation dataset stored in the database. His work is described in more detail in this chapter.

7.1 Background

As mentioned in Section 2.3, the $Z - R$ relation is important in estimating precipitation rate, and its relation is often expressed by Equation 2.1 [16]. Many attempts have been made to obtain values of B and β for snowfalls from quasi-simultaneous measurements of radar reflectivity and ground data on snowfall rate or accumulation [16] [49] [50] [51]. However, the various empirically determined coefficients of the $Z - R$ relationship tend to differ largely. This can be explained by targets of radar and reference on the ground not being the same because of large distance between sampling volume for measuring of radar reflectivity factor and ground reference point for snowfall rate. In addition, snowflakes have a larger variety of shapes, densities, and terminal fall velocities compared to those of raindrops [8] [52]. Furthermore intense snowfalls do not tend to last more than a few minutes [8].

In order to obtain the coefficients of the $Z - R$ relationship accurately, radar reflectivity factor and snowfall rate have to be measured simultaneously by instruments located in a small area and do the recording using a high temporal resolution and time synchronization.

7.2 Observation Data, Database and Data Extraction

The data set for this analysis was recorded during the Wakasa observation campaign, described in Section 2.5. A large number of values were obtained with high temporal resolution from a limited spatial range.

The database visualization tools were used to choose a suitable time range of data and divide it into individual snowfall events. To analyze the $Z - R$ relation, snowfall rate and reflectivity data was retrieved from the database and values automatically matched against each other using the timestamps to give an array of pairs (Z, R) during each snowfall event. The type of snowfall in each event was evaluated using imaging data. Cloud conditions were examined using optical lidar data.

7.3 Division Into Individual Snowfall Events

Figures 7.1a and 7.1b show the time series of snowfall rate and radar reflectivity factor observed during a 16 hour period starting on January 28, 2003. All pairs of Z and R data were plotted as shown in Figure 7.2. The coefficients B and β in Equation 2.1 were obtained from the plotted data. The coefficient B in the $Z - R$ relationship is the radar reflectivity factor on the regression line when the snowfall rate is 1 mm/h. The coefficient β is the slope of the regression line. In this case, the coefficient of determination r^2 between the Z and R was 0.60.

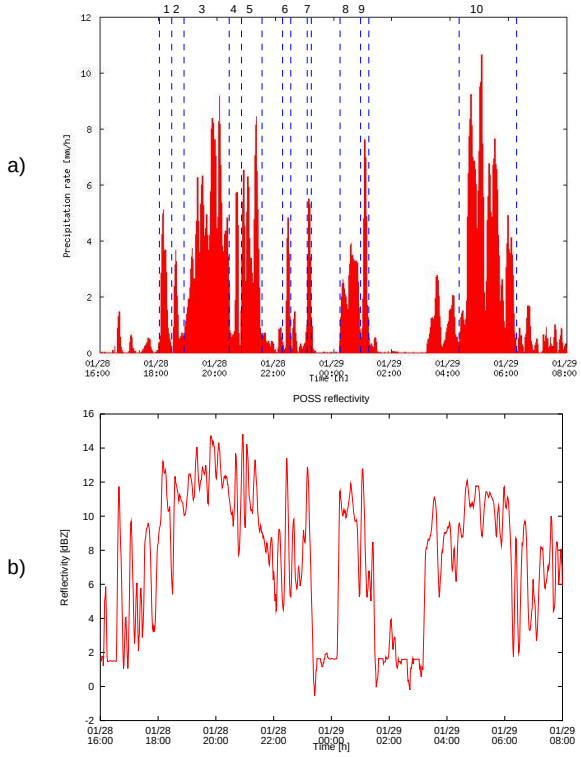

Figure 7.1: Time series of a) snowfall rate and b) radar reflectivity factor from 2003-01-28 16:00:00 until 2003-01-29 08:00:00 Japanese time.

The optical lidar values represent the integrated backscatter up to the height of highest returned signal. The profile of optical lidar values from 0 m up to 3000 m is shown in

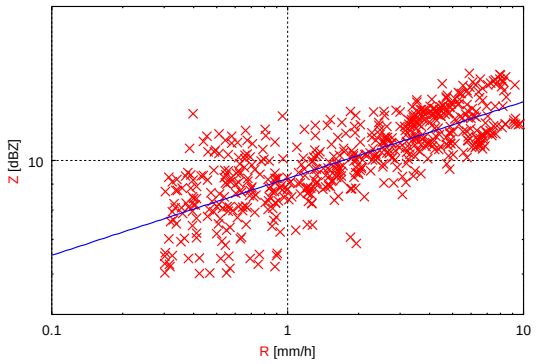

Figure 7.2: The relationship between snowfall rate and radar reflectivity factor during 2003-01-28 16:00:00 – 2003-01-29 08:00:00 Japanese time.

Figure 7.3a. The integrated optical lidar backscatter of the 3 lowest cells from 0 to 90 m in height is shown in Figure 7.3b. The lowest cloud base height is shown in Figure 7.3c.

To obtain a better coefficient of determination for the $Z - R$ relation, the data was divided in shorter time periods by automatically separating individual snowfall events. The algorithm used was:

1. If the balance data indicates a snowfall rate above 3 mm/h continuously for more than 3 minutes and the integrated optical lidar backscatter up to 90 mm (3 cells) is larger than 1500 (10000 srad)-1, there is a snowfall event.
2. The beginning and end of a snowfall event are determined as follows: a snowfall rate below 0.8 mm/h continuously for more than 3 minutes or a rate below 0.3 mm/h for 1 minute.
3. In case the snowfall rate drops lower than 1.0 mm/h during an event, it is divided into two events. The timestamp with the lowest snowfall rate between the events is selected as the separation time.

The data was divided into 10 distinct snowfall events by the algorithm. The division of these events is marked in Figure 7.1a. Table 7.1 shows the coefficients B, β and coefficient of determination r^2 for each snowfall event.

Table 7.1: B, β and r^2 for individual snowfall events.

Event	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Begin	18:03	18:31	18:52	20:34	20:49	22:22	23:02	00:13	00:59	04:18
End	18:22	18:42	20:30	20:45	21:32	22:31	23:16	00:56	01:12	06:13
Duration	20	12	99	12	44	10	15	44	15	116
B	10.60	9.96	10.63	10.19	9.16	8.96	9.34	8.96	7.38	8.56
β	0.12	0.10	0.12	0.13	0.18	0.18	0.14	0.16	0.25	0.14
r^2	0.71	0.12	0.69	0.78	0.69	0.36	0.65	0.63	0.83	0.77

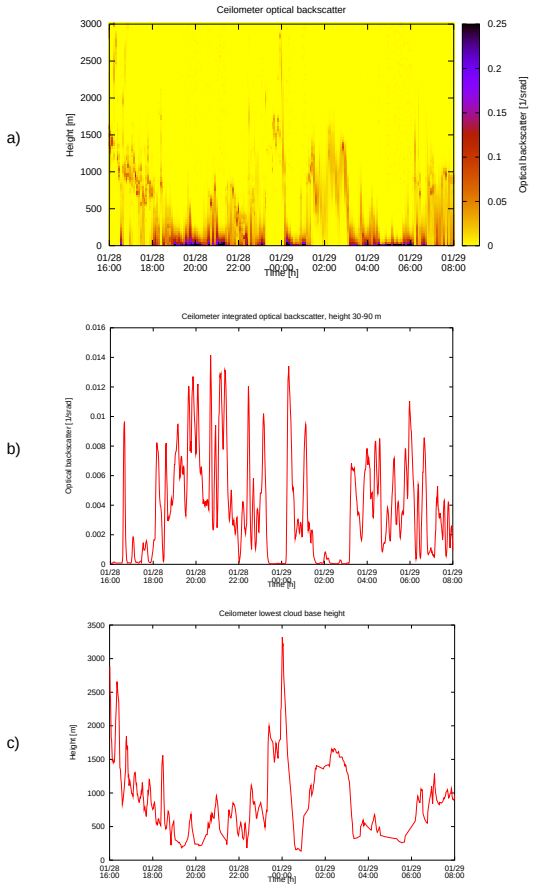

Figure 7.3: Time series of optical lidar data: a) backscatter profile from 0 m up to 3000 m, b) integrated backscatter of the 3 lowest cells from 0 to 90 m in height, c) lowest cloud base height.

7.4 Shortening and Division of Selected Events Using Optical Lidar Data

The coefficient of determination r^2 during events number 2 (18:31 – 18:42) and 6 (22:22 – 22:31) is very low. The durations of these two events were manually shortened using the optical lidar data shown in Figure 7.3. This data provides information about the atmospheric profile between ground and cloud base. The part of $Z - R$ data during periods of atmospheric profile different from the rest of the event was neglected. Figure 7.4 shows the shortening of event 6 to event 6' (22:25 - 22:31) using optical data. The coefficients B and β were again calculated for shortened duration 6' and regression line drawn as shown in Figure 7.5. The coefficient of determination r^2 between Z and R increased from 0.36 to 0.77.

Event number 8 (00:13 - 00:56) was divided manually into two sub-events and event 10 (04:18 – 06:13) into three sub-events. The types of precipitation particles in each event were examined further by using diameter and velocity distribution data recorded by the im-

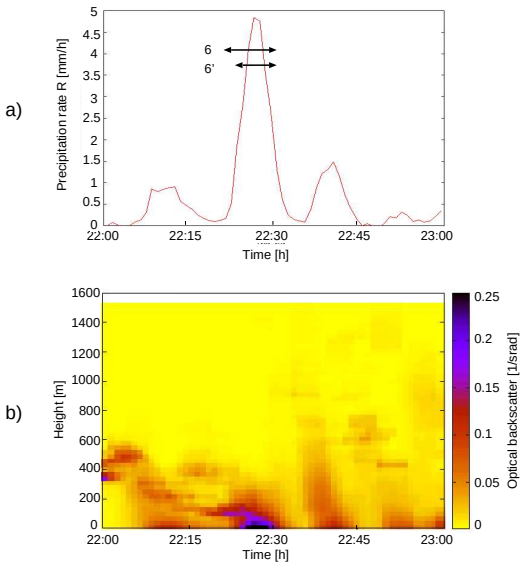

Figure 7.4: Shortening the duration of snowfall event 6: a) snowfall rate, b) the profile of optical lidar backscatter.

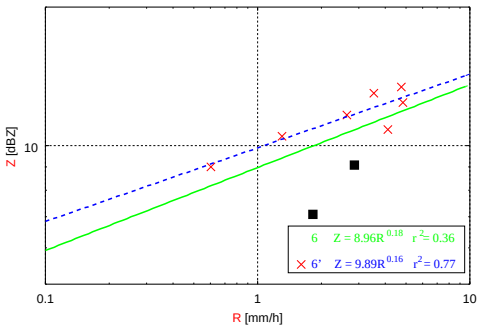

Figure 7.5: Regression lines between snowfall rate and radar reflectivity factor for events 6 (solid line) and shortened event 6' (dashed line).

age processing based snowfall observation system. In event 8, the early part was determined as a graupel shower which changed into snowfall at around 00:30. The coefficients of determination r^2 increased slightly from 0.63 to 0.64 and 0.74. In the case of event 10, snowfall types were almost similar in all sub-events and the average coefficient of determination r^2 increased only marginally from 0.77 to 0.79.

Table 7.2 shows the coefficients B, β and coefficient of determination r^2 for the manually shortened and divided snowfall events. Figure 7.6 shows the time series of B and β. Their respective positions on the $B - \beta$ plane are shown in Figure 7.7.

Table 7.2: B, β and r^2 for shortened events 2', 6' and sub-events 8'a, 8'b, 10'a, 10'b and 10'c.

Event	**2'**	**6'**	**8'a**	**8'b**	**10'a**	**10'a**	**10'c**
Begin	18:32	22:25	00:13	00:29	04:28	05:12	05:53
End	18:38	22:31	00:28	00:56	05:11	05:52	06:13
Duration	7	7	16	28	44	41	21
B	7.70	9.89	7.67	8.97	8.83	8.90	8.73
β	0.38	0.16	0.41	0.15	0.12	0.12	0.16
r^2	0.69	0.77	0.64	0.74	0.83	0.76	0.78

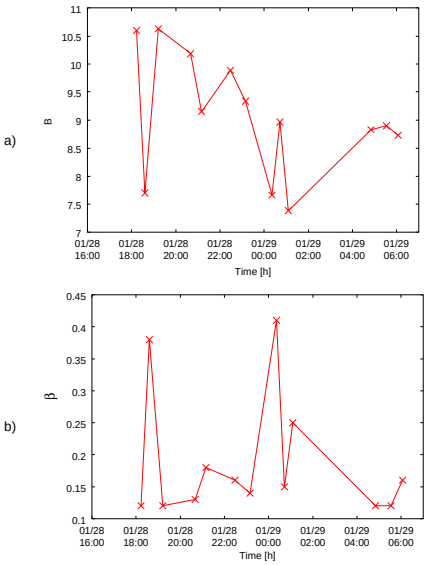

Figure 7.6: Time series of $Z - R$ relation coefficients: a) coefficient B, b) coefficient β.

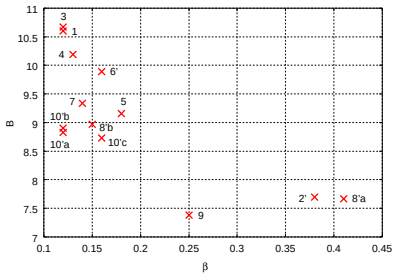

Figure 7.7: Snowfall events on the $B - \beta$ plane.

7.5 Discussion

From ground surface data only, it is not clear when cloud conditions change and produce snowfall that has different characteristics. If we use optical lidar data, however, it becomes easier to detect the changes. From this time series data, we can obtain information of the growth process of snow particles. To obtain the best-fit $Z - R$ relationship, data has to be analyzed separately for each snowfall type.

The radar reflectivity becomes large when the snow particle size increases by coalescence because reflectivity is proportional to the sixth power of the diameter of snow particles [16]. The average of coefficient B in snowflake events was large and widely distributed compared to graupel events. In the case of narrow diameter and velocity distribution, the coefficient of determination r^2 of $Z - R$ relationship was high. On the other hand, in the case of wide diameter distribution, r^2 was low. It indicates that R and Z should be measured at short intervals. Measurement of R and Z at intervals of 1 minute seemed to be adequate with respect to the speed of change in precipitation conditions.

In this example, only a 16 hour period of data was analyzed and there was a lot of manual involvement on dividing the snowfall into separate events. The visualization tools developed in the database project were useful for this kind of analysis. However, the real power of the database becomes apparent when extending the analysis to longer time ranges and developing automatic snowfall classification algorithms, which will be the next step in this research.

Chapter 8

Future Experiments

In the future, the database will continue to be used at least by the snowfall team in Kanazawa University [20]. New data from upcoming observation campaigns will be inserted in the base. The range of instruments will differ from the observation campaign in Fukui 2003, but many of them will be the same. At least data from the video camera system, electronic balance, the optical lidar, POSS and MRR will be included.

The visualization tools will be useful for introducing new students in the laboratory to weather phenomena. Selected plots can be used to demonstrate how rainfall and snowfall events look like in different instruments and then the students can try making plots themselves. This could be made even easier by adding a graphical web interface to the system.

Three students of the snowfall team are going to study changes of the $Z - R$ relation against time by using a moving time window in the instrument data and develop automatic classification of snowfall events. Details of their approach have not been decided yet. It is also not known whether the system will be adopted by the laboratory of River and Environmental Engineering in Tokyo University [24] or some other group.

Chapter 9

Conclusion

The Wakasa database project was started out of a need to get more convenient and uniform access to the measurement data recorded by a large number of instruments. The design succeeded in this main goal. Accessing the data is very similar for all instruments included in the database and users don't have to worry about esoteric file formats any more. The system supports multiple parameter sets and multiple simultaneous observation sites for all the instruments.

The visualization tools proved to be useful already during the prototype phase of the project. Although simple, in many cases they were more useful for real work than the software provided by the manufacturers of the instruments. They saved a lot of time compared to old methods that had been used in the laboratory to make similar kinds of plots.

Reliability of the PostgreSQL database engine was very good during the whole project, not a single crash or loss of data was experienced with the exception of one hard disk failing in the database server. Some performance issues were encountered and solving them took a significant amount of time. When working with databases even tiny details can affect the performance by orders of magnitude. Profiling commands for SQL queries can be very useful to determine bottlenecks in the database engine.

After finding the reason for poor performance and making appropriate fixes the system is fast enough for current use. Some of the visualization tools are still a bit slow when plotting longer time ranges, but that is mainly due to converting database rows to Python matrices at the client end. Load on the database server is low and no scalability issues are foreseen. However, new experiments and statistical analysis algorithms may use a very large number of small queries or more complex queries which have not been tested yet. In particular, matching data elements from different instruments against each other with a selected time resolution requires going through the values with a script and doing extra processing.

The snowfall team in Kanazawa University didn't have many legacy tools fixed to old file formats so moving the data to a database was not a problem. However, that is not always the case. The database have been designed so that it would be easy to add new tables and devices, but if there are already extensive analysis tools for a certain instrument using a specific data format, it may be not be convenient or even possible to change them. New instruments made at the laboratories themselves are the easiest case because they could be configured to record all data directly to the database.

The concept of using a database as a back end storage for raw measurement data is new

for most researchers which are not doing database research themselves, but it will probably become more popular in the future. Open source database engines already deliver good performance for even large data sets and have moved database technology from the world of database experts to any advanced user — to a standard tool which more and more software is built on.

The Wakasa database project project didn't develop new database technology or new methods for snowfall analysis. However, it provides a convenient platform for working on new algorithms, especially statistical algorithms requiring access to large datasets and several measurement instruments. It will be interesting to see which kind of research will be based on the system in the future and whether it will be adapted and extended to new projects.

Chapter 10

Acknowledgements

The author would like to thank Fukui airport research and planning section for having the possibility to use Fukui airport facilities during the Wakasa observation campaign.

The project would not have been possible without the exchange agreement between Helsinki University of Technology and Kanazawa University. Scholarships to support living in Japan were received from the Association of International Exchange, Japan (part of the Japan Student Services Organization [53] since April 1, 2004) and Helsinki University of Technology.

Appendix A

Database Table Layout

Diagram of the table and column structure of the database are presented in this chapter. Standard version is shown in Figures A.1 and A.2, array version is shown in Figures A.3 and A.4. The only differences are in MRR radar and optical lidar (Ceilometer) table layouts.

Each table is represented as a rectangle with a list of columns and column data types. The primary key of each table is listed at the bottom of the rectangle. Dependencies between tables are represented with arrows pointing to the referred column. The referring columns are additionally marked with letters “FK” (Foreign Key).

For some columns, a default value is specified after a “=” sign. If not specified, the default value is NULL. More detailed descriptions of the purpose and semantics of each column can be found in the Wakasa Database User’s Guide [47]. Column data types are described in PostgreSQL documentation [37].

A.1 Standard version

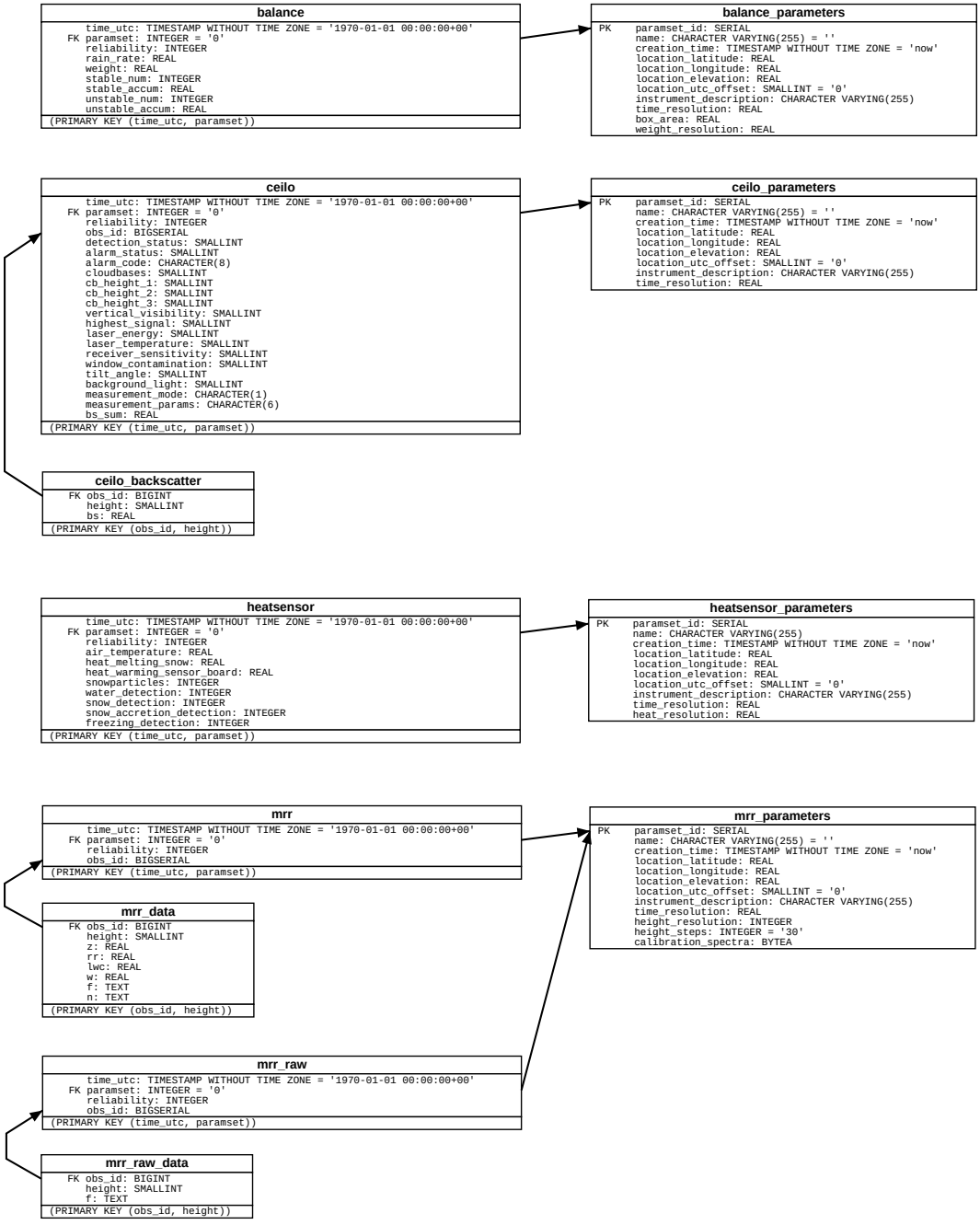

Figure A.1: Wakasa table layout, standard version, page 1(2)

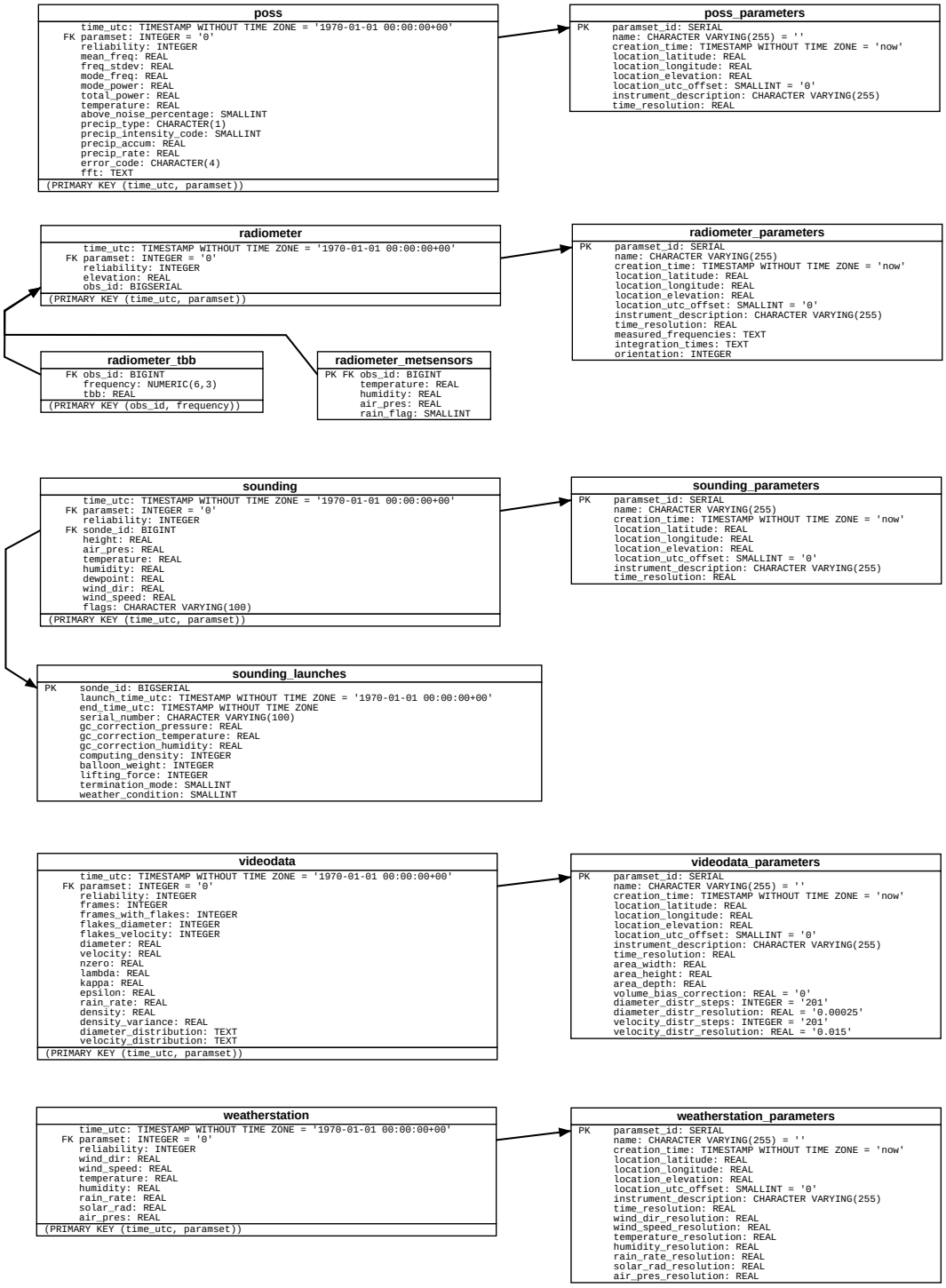

Figure A.2: Wakasa table layout, standard version, page 2(2)

A.2 Array version

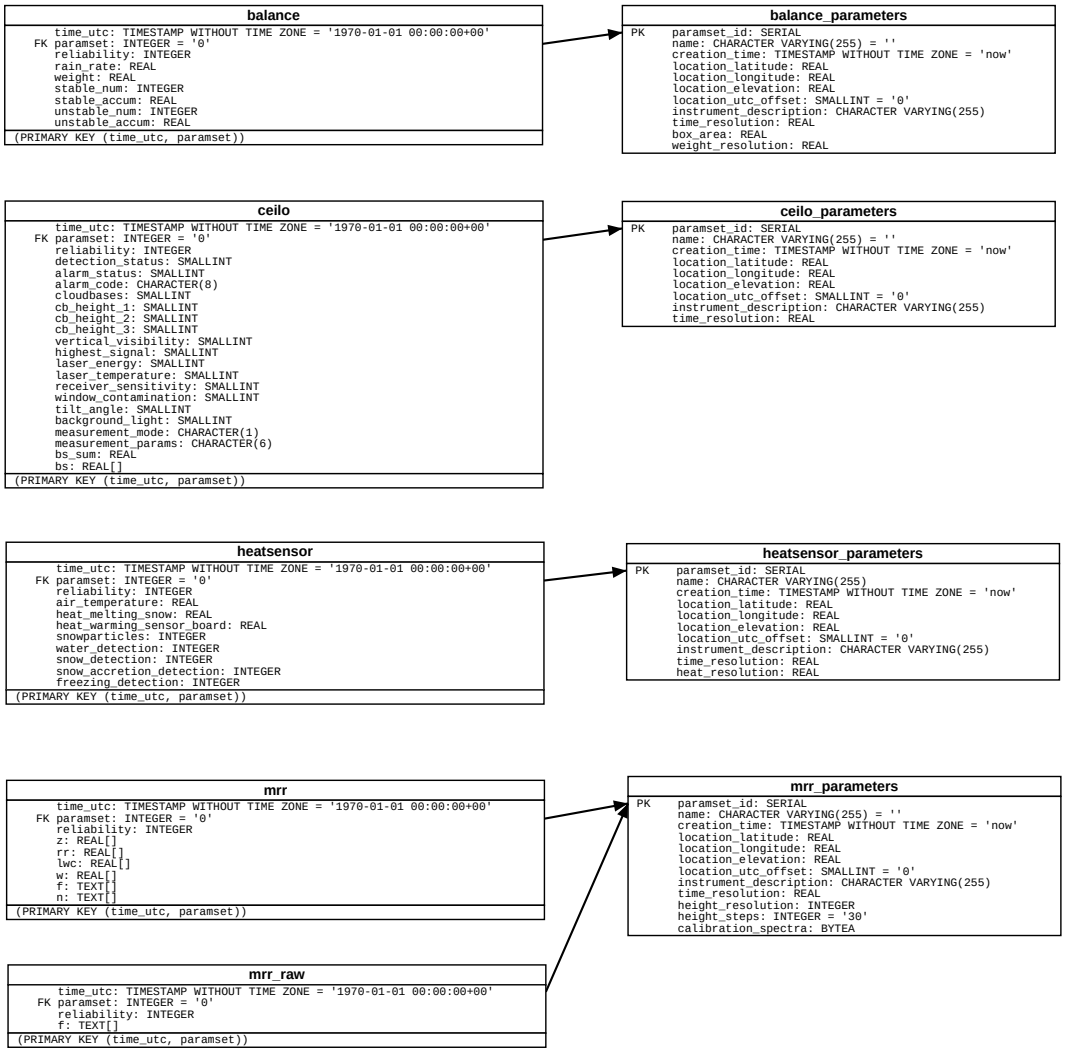

Figure A.3: Wakasa table layout, array version, page 1(2)

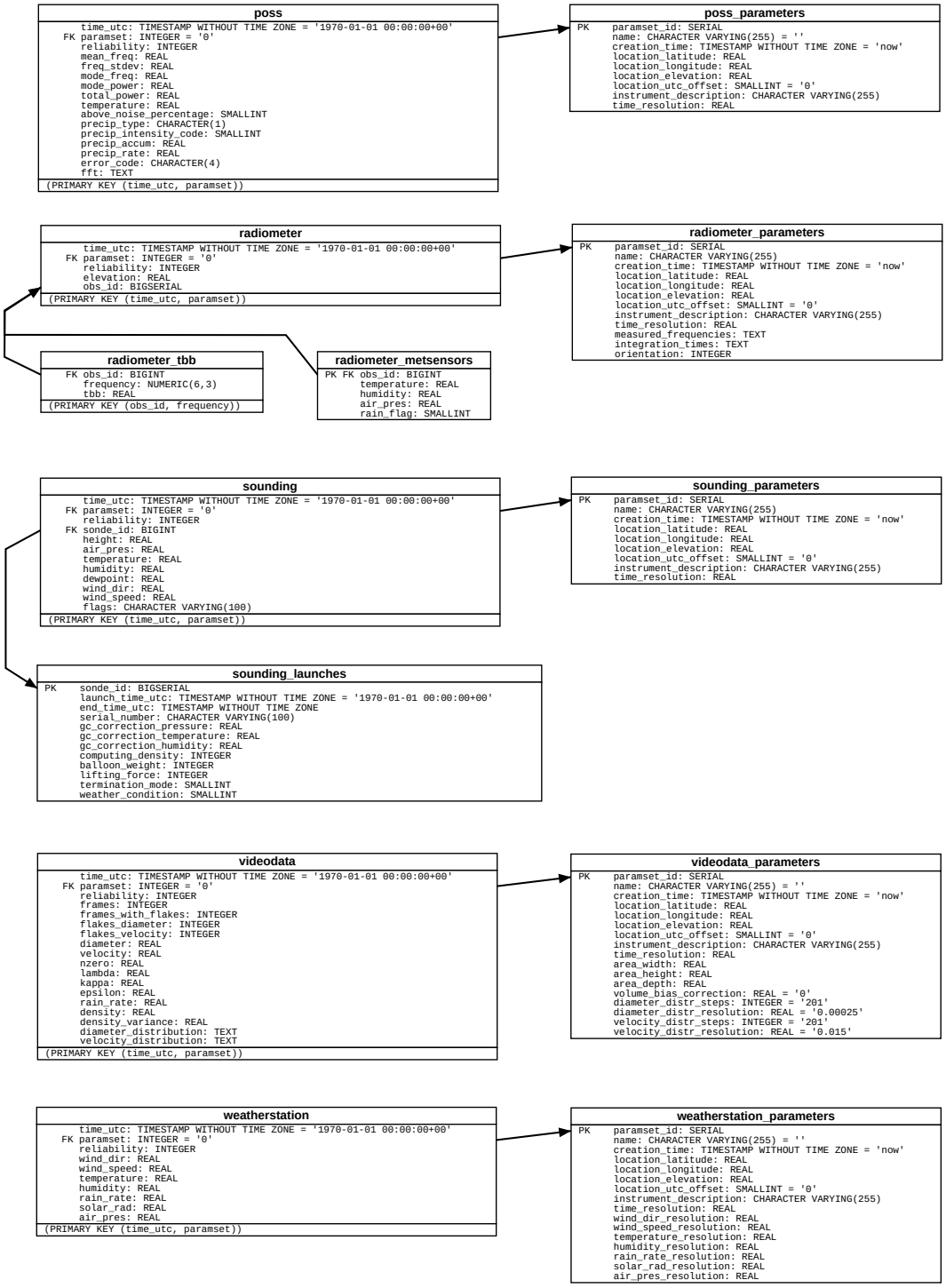

Figure A.4: Wakasa table layout, array version, page 2(2)

Appendix B

Detailed Results of Performance Measurements

The results of performance measurements concerning data visualization and export are presented in detail in this appendix. See Chapter 6 for system setup, general description of the performance measurements and analysis of the most important results. Results of data insertion performance are also presented there.

B.1 Graph Types, Command Lines and SQL Queries

The full set of graph types used in the visualization tool performance measurements is listed in Table B.1.

Table B.1: Graph types used for visualization tool performance measurements.

Graph name	**Instrument**	**Type of visualization**
balance	Electronic balance	Snowfall rate
heat1	Heat sensor	Snow detection flag
heat2	Heat sensor	Heat needed to melt snow
lidar1	Optical lidar	Lowest cloud base height
lidar2	Optical lidar	Integrated backscatter 0–3000 m
lidar3	Optical lidar	Backscatter map 0–3000 m, 30 m height resolution
mrr1	MRR-2 radar	Integrated reflectivity 0–3000 m
mrr2	MRR-2 radar	Reflectivity map 0–3000 m, 120 m height resolution
poss	POSS radar	Integrated reflectivity
wvr	Radiometer WVR1100	Brightness temperature of frequency 23.8 GHz
video1	Video observation system	Snowflake diameter distribution averaged over time
video2	Video observation system	Snowflake diameter distribution against time (map)
video3	Video observation system	Snowflake number concentration
weather1	Weatherstation AWS	Air temperature
weather2	Weatherstation AWS	Wind direction

Command lines for producing each of these graphs are shown in Table B.2. The equivalent SQL queries for retrieving the necessary data from the database are shown in Table B.3. These queries were used in the performance measurements of exporting data. The visualization tools do a couple of additional queries to retrieve the number of parameter set and time zone of the instrument location, but they are insignificant for the total performance of the system when retrieving more than a few minutes worth of measurement data at one time.

Table B.2: Visualization tool command lines equivalent to graph types of Table B.1. Start and end times of each plot were selected at random by the measurement script.

Graph name	**Command line**
balance	`./plotbalance.py -p wakasa2003`
heat1	`./plotheatsensor.py -p wakasa2003 -d snow_flag`
heat2	`./plotheatsensor.py -p wakasa2003 -d heat_melting`
lidar1	`./plotceilo.py -p wakasa2003 -d lowestcb`
lidar2	`./plotceilo.py -p wakasa2003 -d bsint --height=0:3000`
lidar3	`./plotceilo.py -p wakasa2003 -d bsmap --height=0:3000 --centerheights`
mrr1	`./plotmrr.py -p wakasa2003_120m -d zint --height=0:3000`
mrr2	`./plotmrr.py -p wakasa2003_120m -d zmap --height=0:3000 --centerheights`
poss	`./plotposs.py -p wakasa2003 -d reflectivity`
wvr	`./plotradiometer.py -p wakasa2003-wvr -d 'bt(23.8)'`
video1	`./plotvideo.py -p wakasa2003 -d ddistr`
video2	`./plotvideo.py -p wakasa2003 -d ddistrmap`
video3	`./plotvideo.py -p wakasa2003 -d number`
weather1	`./plotweather.py -p wakasa2003 -d temperature`
weather2	`./plotweather.py -p wakasa2003 -d wind_dir`

The following common part was added at the end of each command line:

`-l --timeres=60`
`-c plotter.conf.noarrays`
(`-c plotter.conf.arrays` for the array version of the base)
`--gpcmdfile=/tmp/perftest-plot.cmd`
`--gpdatafile=/tmp/perftest-plot.data`
`--outputfile=/tmp/perftest-plot.eps`
`--starttime='<starttime>'`
`--endtime='<endtime>'`
`1>logs/<graphtype>.stdout`
`2>logs/<graphtype>.stderr`

Table B.3: SQL queries equivalent to graph types of Table B.1. Start and end times of each query were selected at random by the measurement script.

Graph name	Query

balance	`SELECT time_utc, rain_rate FROM balance`
	`WHERE paramset = '1'`
	`AND time_utc >= '<startdate>'`
	`AND time_utc <= '<enddate>'`
	`AND rain_rate IS NOT NULL`
	`ORDER by time_utc;`

heat1	`SELECT time_utc, snow_detection FROM heatsensor`
	`WHERE paramset = '1'`
	`AND time_utc >= '<startdate>'`
	`AND time_utc <= '<enddate>'`
	`ORDER by time_utc;`

heat2	`SELECT time_utc, heat_melting_snow FROM heatsensor`
	`WHERE paramset = '1'`
	`AND time_utc >= '<startdate>'`
	`AND time_utc <= '<enddate>'`
	`AND heat_melting_snow IS NOT NULL`
	`ORDER by time_utc;`

lidar1	`SELECT time_utc, cb_height_1 FROM ceilo`
	`WHERE paramset = '1'`
	`AND time_utc >= '<startdate>'`
	`AND time_utc <= '<enddate>'`
	`AND cb_height_1 IS NOT NULL`
	`ORDER by time_utc;`

lidar2 (standard version)	`SELECT time_utc, height, bs FROM ceilo, ceilo_backscatter`
	`WHERE ceilo.paramset = '1'`
	`AND time_utc >= '<startdate>'`
	`AND time_utc <= '<enddate>'`
	`AND ceilo.obs_id = ceilo_backscatter.obs_id`
	`AND ceilo_backscatter.height > '0'`
	`AND ceilo_backscatter.height <= '3000'`
	`ORDER by time_utc, height;`

lidar2 (array version)	`SELECT time_utc, bs[1:100] FROM ceilo`
	`WHERE paramset = '1'`
	`AND time_utc >= '<startdate>'`
	`AND time_utc <= '<enddate>'`
	`ORDER by time_utc;`

lidar3	Equivalent to the query of graph lidar2.

(Continued on next page)

(Continued from previous page)

Graph name	Query
mrr1 (standard version	`SELECT time_utc, height, z FROM mrr, mrr_data WHERE mrr.paramset = '2' AND time_utc >= '<startdate>' AND time_utc <= '<enddate>' AND mrr.obs_id = mrr_data.obs_id AND mrr_data.height > '0' AND mrr_data.height <= '3000' ORDER by time_utc, height;`
mrr1 (array version)	`SELECT time_utc, z[1:25] FROM mrr WHERE paramset = '2' AND time_utc >= '<startdate>' AND time_utc <= '<enddate>' ORDER by time_utc;`
mrr2	Equivalent to the query of graph mrr1.
poss	`SELECT time_utc, total_power FROM poss WHERE paramset = '1' AND time_utc >= '<startdate>' AND time_utc <= '<enddate>' ORDER by time_utc;`
wvr	`SELECT time_utc, tbb FROM radiometer, radiometer_tbb WHERE radiometer.paramset = '1' AND time_utc >= '<startdate>' AND time_utc <= '<enddate>' AND radiometer.obs_id = radiometer_tbb.obs_id AND radiometer_tbb.frequency = 23.8::numeric AND tbb IS NOT NULL ORDER by time_utc;`
video1	`SELECT time_utc, frames, diameter_distribution FROM videodata WHERE paramset = '1' AND time_utc >= '<startdate>' AND time_utc <= '<enddate>' AND diameter_distribution IS NOT NULL ORDER by time_utc;`
video2	Equivalent to the query of graph video1.
video3	`SELECT time_utc, frames, flakes_diameter FROM videodata WHERE paramset = '1' AND time_utc >= '<startdate>' AND time_utc <= '<enddate>' AND flakes_diameter IS NOT NULL ORDER by time_utc;`

(Continued on next page)

(Continued from previous page)

Graph name	Query

weather1	`SELECT time_utc, temperature FROM weatherstation`
	`WHERE paramset = '1'`
	`AND time_utc >= '<startdate>'`
	`AND time_utc <= '<enddate>'`
	`AND temperature IS NOT NULL`
	`ORDER by time_utc;`

weather2	`SELECT time_utc, wind_dir FROM weatherstation`
	`WHERE paramset = '1'`
	`AND time_utc >= '<startdate>'`
	`AND time_utc <= '<enddate>'`
	`AND wind_dir IS NOT NULL`
	`ORDER by time_utc;`

B.2 Visualization Tools

Visualization tool performance results for each of the graphs in Table B.1 are presented in Tables B.4 through B.7.

Table B.4: Performance of visualization tools, standard version of the database, system not functioning properly. All execution times in seconds.

Graph	4 min	15 min	1 h	4 h	16 h	24 h
balance	0.55 ± 0.02	0.58 ± 0.02	0.67 ± 0.02	1.08 ± 0.02	2.63 ± 0.02	3.74 ± 0.02
heat1	0.26 ± 0.04	0.25 ± 0.04	0.26 ± 0.04	0.28 ± 0.04	0.33 ± 0.01	0.35 ± 0.00
heat2	0.24 ± 0.04	0.26 ± 0.04	0.25 ± 0.04	0.27 ± 0.05	0.33 ± 0.02	0.36 ± 0.00
lidar1	0.39 ± 0.03	0.41 ± 0.02	0.42 ± 0.01	0.49 ± 0.04	0.81 ± 0.12	0.97 ± 0.14
lidar2	38.78 ± 0.41	39.59 ± 0.20	43.51 ± 0.21	58.64 ± 0.25	119.63 ± 0.32	160.54 ± 0.39
lidar3	38.74 ± 0.28	39.86 ± 0.22	44.17 ± 0.24	61.53 ± 0.26	131.25 ± 0.31	177.90 ± 0.50
mrr1	8.83 ± 1.03	8.72 ± 0.01	9.02 ± 0.02	10.21 ± 0.03	15.08 ± 0.11	18.29 ± 0.18
mrr2	8.66 ± 0.02	8.76 ± 0.01	9.21 ± 0.02	10.93 ± 0.07	17.98 ± 0.14	22.58 ± 0.22
poss	0.36 ± 0.05	0.35 ± 0.00	0.37 ± 0.00	0.46 ± 0.00	0.81 ± 0.00	1.04 ± 0.00
wvr	0.57 ± 0.08	0.57 ± 0.00	0.59 ± 0.03	0.70 ± 0.06	1.16 ± 0.12	1.46 ± 0.16
video1	0.39 ± 0.04	0.41 ± 0.04	0.46 ± 0.04	0.67 ± 0.08	1.48 ± 0.24	1.97 ± 0.26
video2	0.42 ± 0.06	0.61 ± 0.08	1.25 ± 0.26	3.55 ± 1.29	14.70 ± 3.05	18.75 ± 3.13
video3	0.39 ± 0.08	0.41 ± 0.09	0.41 ± 0.03	0.52 ± 0.08	1.00 ± 0.18	0.97 ± 0.21
weather1	0.30 ± 0.06	0.31 ± 0.00	0.32 ± 0.01	0.37 ± 0.06	0.56 ± 0.23	0.63 ± 0.28
weather2	0.32 ± 0.05	0.31 ± 0.02	0.32 ± 0.02	0.34 ± 0.03	0.51 ± 0.18	0.64 ± 0.27

Table B.5: Performance of visualization tools, array version of the database, system not functioning properly. All execution times in seconds.

Graph	4 min	15 min	1 h	4 h	16 h	24 h
balance	0.56 ± 0.07	0.58 ± 0.02	0.67 ± 0.02	1.07 ± 0.02	2.62 ± 0.02	3.75 ± 0.02
heat1	0.25 ± 0.04	0.25 ± 0.04	0.27 ± 0.04	0.26 ± 0.04	0.33 ± 0.01	0.35 ± 0.00
heat2	0.25 ± 0.04	0.26 ± 0.04	0.25 ± 0.04	0.27 ± 0.04	0.34 ± 0.02	0.36 ± 0.00
lidar1	0.68 ± 0.03	0.69 ± 0.02	0.71 ± 0.03	0.78 ± 0.04	1.07 ± 0.14	1.24 ± 0.16
lidar2	0.83 ± 0.01	1.19 ± 0.02	2.71 ± 0.05	8.82 ± 0.17	33.19 ± 0.41	49.33 ± 0.40
lidar3	0.87 ± 0.01	1.37 ± 0.02	3.41 ± 0.05	11.55 ± 0.20	44.29 ± 0.39	66.27 ± 0.45
mrr1	0.62 ± 0.00	0.65 ± 0.00	0.79 ± 0.00	1.31 ± 0.00	3.44 ± 0.01	4.84 ± 0.02
mrr2	0.64 ± 0.00	0.70 ± 0.00	0.97 ± 0.00	2.04 ± 0.02	6.32 ± 0.04	9.17 ± 0.04
poss	0.34 ± 0.00	0.35 ± 0.00	0.37 ± 0.00	0.46 ± 0.00	0.81 ± 0.00	1.04 ± 0.00
wvr	0.55 ± 0.02	0.56 ± 0.02	0.60 ± 0.00	0.68 ± 0.08	1.17 ± 0.08	1.46 ± 0.15
video1	0.39 ± 0.04	0.41 ± 0.03	0.47 ± 0.02	0.68 ± 0.07	1.47 ± 0.24	1.83 ± 0.26
video2	0.44 ± 0.04	0.56 ± 0.12	1.22 ± 0.30	3.37 ± 1.15	13.06 ± 2.47	19.14 ± 2.86
video3	0.37 ± 0.03	0.38 ± 0.03	0.40 ± 0.04	0.48 ± 0.02	0.73 ± 0.08	0.86 ± 0.07
weather1	0.29 ± 0.03	0.31 ± 0.00	0.31 ± 0.01	0.35 ± 0.05	0.45 ± 0.16	0.48 ± 0.20
weather2	0.28 ± 0.04	0.30 ± 0.01	0.31 ± 0.01	0.35 ± 0.05	0.46 ± 0.16	0.60 ± 0.28

Table B.6: Performance of visualization tools, standard version of the database, after vacuuming. All execution times in seconds.

Graph	4 min	15 min	1 h	4 h	16 h	24 h
balance	0.29 ± 0.01	0.31 ± 0.00	0.41 ± 0.02	0.82 ± 0.01	2.38 ± 0.01	3.42 ± 0.01
heat1	0.25 ± 0.04	0.24 ± 0.04	0.26 ± 0.05	0.26 ± 0.05	0.34 ± 0.02	0.36 ± 0.00
heat2	0.25 ± 0.04	0.25 ± 0.04	0.26 ± 0.04	0.28 ± 0.05	0.33 ± 0.01	0.35 ± 0.00
lidar1	0.31 ± 0.06	0.31 ± 0.02	0.34 ± 0.01	0.40 ± 0.05	0.71 ± 0.09	0.93 ± 0.16
lidar2	0.65 ± 0.01	1.44 ± 0.02	4.75 ± 0.05	18.07 ± 0.14	104.55 ± 1.57	139.82 ± 0.52
lidar3	0.69 ± 0.02	1.62 ± 0.02	5.44 ± 0.04	20.87 ± 0.13	116.10 ± 0.91	158.45 ± 0.67
mrr1	0.36 ± 0.01	0.44 ± 0.02	0.75 ± 0.04	1.92 ± 0.09	6.57 ± 0.27	11.95 ± 4.65
mrr2	0.37 ± 0.01	0.48 ± 0.02	0.93 ± 0.04	2.62 ± 0.07	9.49 ± 0.16	17.05 ± 5.13
poss	0.29 ± 0.01	0.29 ± 0.00	0.31 ± 0.00	0.40 ± 0.00	0.76 ± 0.00	0.98 ± 0.00
wvr	0.30 ± 0.02	0.30 ± 0.03	0.32 ± 0.04	0.43 ± 0.06	0.90 ± 0.13	1.15 ± 0.21
video1	0.31 ± 0.04	0.33 ± 0.04	0.39 ± 0.05	0.62 ± 0.06	1.35 ± 0.24	1.67 ± 0.17
video2	0.33 ± 0.07	0.48 ± 0.14	1.12 ± 0.30	3.19 ± 1.31	12.53 ± 2.83	18.86 ± 2.97
video3	0.30 ± 0.04	0.31 ± 0.03	0.33 ± 0.03	0.40 ± 0.03	0.65 ± 0.08	0.81 ± 0.07
weather1	0.26 ± 0.04	0.28 ± 0.03	0.30 ± 0.01	0.35 ± 0.06	0.56 ± 0.24	0.53 ± 0.25
weather2	0.25 ± 0.04	0.29 ± 0.00	0.30 ± 0.02	0.32 ± 0.03	0.47 ± 0.19	0.53 ± 0.21

Table B.7: Performance of visualization tools, array version of the database, after vacuuming. All execution times in seconds.

Graph	4 min	15 min	1 h	4 h	16 h	24 h
balance	0.29 ± 0.01	0.31 ± 0.00	0.41 ± 0.00	0.82 ± 0.00	2.38 ± 0.01	3.43 ± 0.01
heat1	0.25 ± 0.04	0.24 ± 0.04	0.25 ± 0.05	0.27 ± 0.05	0.34 ± 0.02	0.36 ± 0.00
heat2	0.23 ± 0.04	0.25 ± 0.04	0.26 ± 0.04	0.28 ± 0.04	0.33 ± 0.01	0.35 ± 0.00
lidar1	0.30 ± 0.03	0.31 ± 0.02	0.33 ± 0.02	0.40 ± 0.04	0.74 ± 0.11	0.91 ± 0.16
lidar2	0.44 ± 0.00	0.82 ± 0.01	2.31 ± 0.05	8.44 ± 0.21	32.79 ± 0.43	48.94 ± 0.49
lidar3	0.49 ± 0.01	0.99 ± 0.01	3.02 ± 0.05	11.20 ± 0.19	44.03 ± 0.45	66.17 ± 0.50
mrr1	0.33 ± 0.01	0.36 ± 0.01	0.50 ± 0.01	1.04 ± 0.02	3.17 ± 0.04	4.57 ± 0.02
mrr2	0.34 ± 0.00	0.41 ± 0.00	0.67 ± 0.00	1.75 ± 0.02	6.09 ± 0.05	8.94 ± 0.07
poss	0.29 ± 0.00	0.29 ± 0.00	0.31 ± 0.00	0.40 ± 0.00	0.75 ± 0.00	0.98 ± 0.00
wvr	0.30 ± 0.01	0.29 ± 0.04	0.32 ± 0.03	0.44 ± 0.05	0.93 ± 0.05	1.23 ± 0.11
video1	0.31 ± 0.04	0.33 ± 0.04	0.39 ± 0.05	0.59 ± 0.10	1.43 ± 0.21	1.75 ± 0.23
video2	0.34 ± 0.07	0.52 ± 0.10	1.15 ± 0.30	3.73 ± 0.68	13.00 ± 2.65	18.66 ± 2.98
video3	0.31 ± 0.03	0.30 ± 0.04	0.33 ± 0.02	0.40 ± 0.03	0.66 ± 0.08	0.79 ± 0.08
weather1	0.27 ± 0.04	0.29 ± 0.00	0.30 ± 0.02	0.34 ± 0.07	0.49 ± 0.21	0.50 ± 0.22
weather2	0.26 ± 0.04	0.29 ± 0.00	0.30 ± 0.03	0.34 ± 0.05	0.48 ± 0.17	0.53 ± 0.26

B.3 Exporting data

Performance results of exporting the data equivalent to each of the graphs in Table B.1 into a file are presented in Tables B.8 through B.11. The SQL query for graph `lidar3` is identical to the query of graph `lidar2`, the query for `mrr1` is identical to that of `mrr2` and the query for `video1` is identical to that of `video2`. Therefore data export performance is also identical in these cases and the redundant lines are not shown in the results.

Table B.8: Performance of exporting data, standard version of the database, system not functioning properly. All execution times in seconds.

Graph	4 min	15 min	1 h	4 h	16 h	24 h
balance	0.31 ± 0.02	0.32 ± 0.02	0.32 ± 0.02	0.35 ± 0.02	0.46 ± 0.02	0.67 ± 0.02
heat1	0.05 ± 0.01	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.05 ± 0.00	0.05 ± 0.00
heat2	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.05 ± 0.00	0.05 ± 0.00
lidar1	0.14 ± 0.01	0.14 ± 0.01	0.14 ± 0.01	0.15 ± 0.01	0.17 ± 0.01	0.19 ± 0.01
lidar2	40.07 ± 0.64	40.18 ± 0.27	41.23 ± 0.27	44.67 ± 0.30	59.67 ± 0.59	69.56 ± 0.36
mrr1	10.79 ± 0.53	10.72 ± 0.06	10.85 ± 0.10	11.14 ± 0.08	12.44 ± 0.06	13.28 ± 0.06
poss	0.11 ± 0.01	0.11 ± 0.00	0.11 ± 0.00	0.11 ± 0.00	0.13 ± 0.00	0.15 ± 0.00
wvr	0.32 ± 0.05	0.31 ± 0.00	0.32 ± 0.00	0.34 ± 0.01	0.41 ± 0.01	0.46 ± 0.02
video1	0.12 ± 0.00	0.12 ± 0.00	0.12 ± 0.00	0.12 ± 0.00	0.15 ± 0.03	0.19 ± 0.04
video3	0.12 ± 0.03	0.12 ± 0.00	0.12 ± 0.00	0.12 ± 0.00	0.14 ± 0.00	0.14 ± 0.00
weather1	0.07 ± 0.01	0.06 ± 0.00	0.06 ± 0.00	0.07 ± 0.00	0.08 ± 0.01	0.08 ± 0.02
weather2	0.06 ± 0.00	0.06 ± 0.00	0.06 ± 0.00	0.07 ± 0.00	0.07 ± 0.01	0.08 ± 0.02

Table B.9: Performance of exporting data, array version of the database, system not functioning properly. All execution times in seconds.

Graph	4 min	15 min	1 h	4 h	16 h	24 h
balance	0.32 ± 0.06	0.31 ± 0.02	0.32 ± 0.02	0.34 ± 0.02	0.47 ± 0.01	0.67 ± 0.01
heat1	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.05 ± 0.00	0.05 ± 0.00
heat2	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.05 ± 0.00	0.05 ± 0.00
lidar1	0.42 ± 0.01	0.42 ± 0.01	0.42 ± 0.01	0.42 ± 0.01	0.45 ± 0.01	0.46 ± 0.01
lidar2	0.42 ± 0.01	0.43 ± 0.01	0.48 ± 0.01	0.66 ± 0.01	1.46 ± 0.02	1.96 ± 0.03
mrr1	0.33 ± 0.00	0.33 ± 0.00	0.34 ± 0.00	0.36 ± 0.00	0.43 ± 0.00	0.49 ± 0.00
poss	0.11 ± 0.00	0.11 ± 0.00	0.11 ± 0.00	0.11 ± 0.00	0.13 ± 0.00	0.15 ± 0.00
wvr	0.31 ± 0.00	0.31 ± 0.00	0.32 ± 0.00	0.34 ± 0.00	0.40 ± 0.02	0.45 ± 0.03
video1	0.11 ± 0.00	0.12 ± 0.00	0.12 ± 0.00	0.12 ± 0.00	0.15 ± 0.00	0.19 ± 0.04
video3	0.11 ± 0.00	0.12 ± 0.00	0.12 ± 0.00	0.12 ± 0.00	0.14 ± 0.01	0.15 ± 0.01
weather1	0.06 ± 0.00	0.06 ± 0.00	0.06 ± 0.00	0.07 ± 0.00	0.07 ± 0.01	0.09 ± 0.02
weather2	0.06 ± 0.00	0.06 ± 0.00	0.06 ± 0.00	0.07 ± 0.00	0.07 ± 0.01	0.08 ± 0.01

Table B.10: Performance of exporting data, standard version of the database, after vacuuming. All execution times in seconds.

Graph	4 min	15 min	1 h	4 h	16 h	24 h
balance	0.05 ± 0.03	0.04 ± 0.01	0.05 ± 0.01	0.13 ± 0.12	0.22 ± 0.00	0.31 ± 0.00
heat1	0.04 ± 0.02	0.03 ± 0.00	0.03 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00
heat2	0.03 ± 0.00	0.03 ± 0.00	0.03 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00
lidar1	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.09 ± 0.06	0.14 ± 0.01	0.16 ± 0.01
lidar2	0.11 ± 0.02	0.24 ± 0.03	0.67 ± 0.04	2.44 ± 0.13	44.92 ± 1.30	48.66 ± 0.77
mrr1	0.07 ± 0.01	0.09 ± 0.02	0.17 ± 0.03	0.43 ± 0.05	1.49 ± 0.17	4.35 ± 4.70
poss	0.04 ± 0.01	0.04 ± 0.01	0.04 ± 0.00	0.06 ± 0.03	0.10 ± 0.00	0.12 ± 0.00
wvr	0.05 ± 0.01	0.04 ± 0.01	0.05 ± 0.01	0.24 ± 0.01	0.32 ± 0.02	0.36 ± 0.01
video1	0.03 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.05 ± 0.00	0.09 ± 0.01	0.11 ± 0.01
video3	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.08 ± 0.02	0.09 ± 0.00
weather1	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.05 ± 0.01	0.06 ± 0.01	0.06 ± 0.01
weather2	0.03 ± 0.00	0.03 ± 0.00	0.04 ± 0.00	0.04 ± 0.01	0.06 ± 0.01	0.06 ± 0.02

Table B.11: Performance of exporting data, array version of the database, after vacuuming. All execution times in seconds.

Graph	4 min	15 min	1 h	4 h	16 h	24 h
balance	0.05 ± 0.03	0.05 ± 0.02	0.05 ± 0.01	0.10 ± 0.07	0.22 ± 0.00	0.31 ± 0.00
heat1	0.04 ± 0.01	0.03 ± 0.00	0.03 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00
heat2	0.03 ± 0.00	0.03 ± 0.00	0.03 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00
lidar1	0.04 ± 0.01	0.04 ± 0.00	0.04 ± 0.01	0.06 ± 0.01	0.27 ± 0.20	0.47 ± 0.01
lidar2	0.04 ± 0.00	0.05 ± 0.00	0.10 ± 0.00	0.28 ± 0.01	1.27 ± 0.18	1.89 ± 0.02
mrr1	0.04 ± 0.01	0.04 ± 0.01	0.05 ± 0.01	0.07 ± 0.01	0.16 ± 0.03	0.22 ± 0.05
poss	0.04 ± 0.01	0.04 ± 0.00	0.04 ± 0.00	0.06 ± 0.02	0.10 ± 0.00	0.12 ± 0.00
wvr	0.05 ± 0.02	0.05 ± 0.01	0.05 ± 0.01	0.24 ± 0.00	0.32 ± 0.02	0.36 ± 0.01
video1	0.03 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.05 ± 0.03	0.10 ± 0.01	0.11 ± 0.01
video3	0.04 ± 0.01	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.08 ± 0.01	0.09 ± 0.00
weather1	0.04 ± 0.01	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.01	0.06 ± 0.01	0.06 ± 0.01
weather2	0.03 ± 0.00	0.03 ± 0.00	0.04 ± 0.00	0.04 ± 0.01	0.06 ± 0.01	0.06 ± 0.01

Appendix C

Visualization examples

The visualization tools are a set of command line scripts which can be used to see graphically the data stored in the database. The scripts retrieve the data from the base and do necessary preprocessing, the actual output is produced by the Gnuplot plotting tool [43]. There is one visualization script for each type of instrument and they accept various parameters to select data type, time range and type of plot. Time resolution, smoothing using moving average and other options can also be selected by the user. This appendix presents some examples and the effect of a few important command line options to give an overview of the visualization tool capabilities. For a complete description, see the Wakasa Database User's Guide [47].

Simplest type of graph is a two dimensional line or bar graph which is suitable for plotting a single measurement value against time. This type of graph is available for most instruments. For electronic balance the only available graph is precipitation rate, but in most other cases several different types of data can be plotted. For example in the case of weather station, the user can select between temperature, humidity, wind direction, wind speed, air pressure, solar radiation and rain rate plots. Time resolution adjustment and smoothing are useful for eliminating constant small variations in values. Figure C.1 shows an example of precipitation rate plot using 10 second time resolution (maximum provided by the device) and Figure C.2 shows the same time range with one minute time resolution and smoothing by 3 minute moving average. The y axis range in Figure C.2 is forced to same than in Figure C.1 to make comparison easier.

For radio soundings it is more useful to plot values against height than time. Figure C.3 shows how wind direction changes with height during one radio sounding launch. For instruments such as the optical lidar which produce a height spectra of values, color map plots are available. One such plot for the optical lidar data was shown in Figure 5.1 and another one for MRR radar data is presented in Figure C.4. In this case, the user needs to manually adjust the color range and point size to produce a smooth color map without white space between the points. This is a limitation of the Gnuplot plotting tool. The white areas seen in the graph are actual gaps in MRR data.

The video camera based observation system measures snowflake size and velocity distributions. The visualization tools allow to plot the distributions either as a bar graph averaged over a certain time period or as a color map plot showing the change of one minute distributions against time. Figures C.5 and C.6 show an example of diameter distribution plotted using these two methods for the same time range.

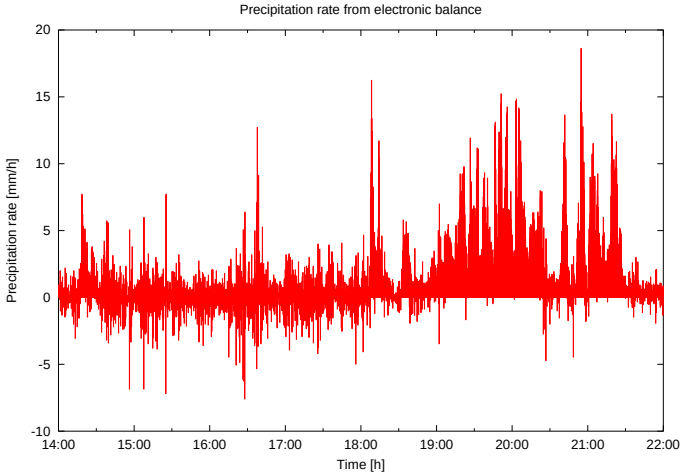

Figure C.1: Example output of `plotbalance.py`. This plot was produced using the following command line: `./plotbalance.py -p wakasa2003 -s "2003-01-28 14:00:00+09" -e "2003-01-28 22:00:00+09" -l`

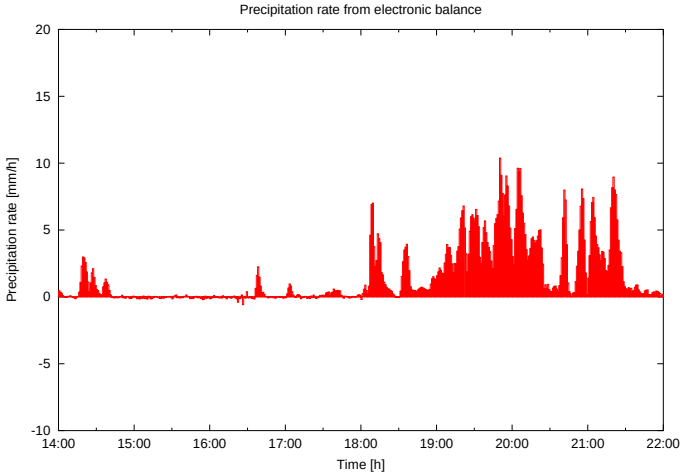

Figure C.2: Example output of `plotbalance.py`. This plot was produced using the following command line: `./plotbalance.py -p wakasa2003 -s "2003-01-28 14:00:00+09" -e "2003-01-28 22:00:00+09" -l --timeres=60 --avg=3 --yrange="-10:20"`

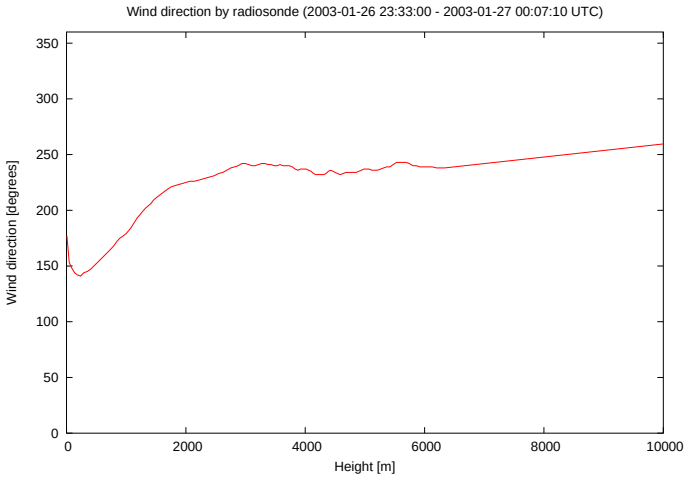

Figure C.3: Example output of `plotsonde.py`. This plot was produced using the following command line: `./plotsonde.py -p wakasa2003-fukui -d wind_dir -s "2003-01-27 08:00:00+09" -e "2003-01-27 10:00:00+09" --xrange=0:10000 --yrange=0:360`

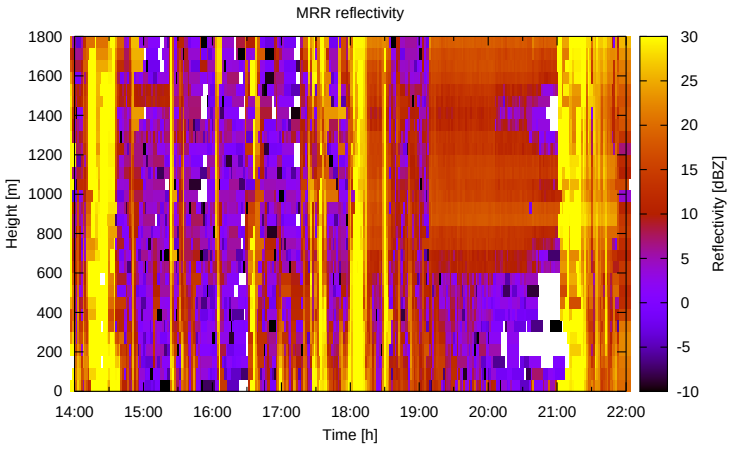

Figure C.4: Example output of `plotmrr.py`. This plot was produced using the following command line: `./plotmrr.py -p wakasa2003_60m -d zmap -s "2003-01-28 14:00:00+09" -e "2003-01-28 22:00:00+09" -l --timeres=60 --centerheights --cbrange=-10:30 --pointsize=1.7`

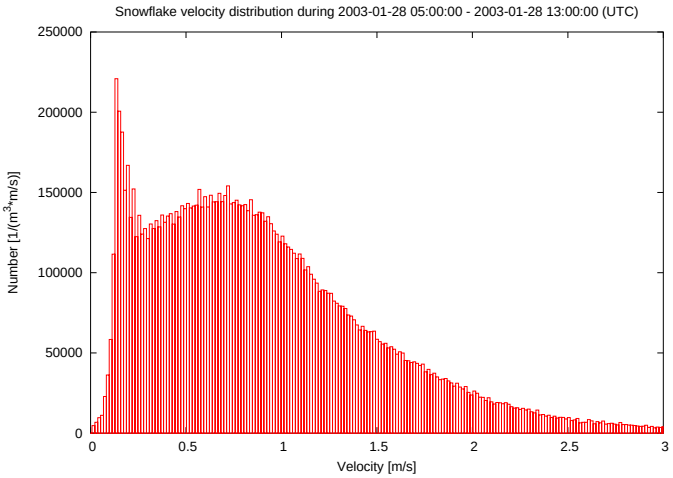

Figure C.5: Example output of `plotvideo.py`. This plot was produced using the following command line: `./plotvideo.py -p wakasa2003 -d vdistr -s "2003-01-28 14:00:00+09" -e "2003-01-28 22:00:00+09" -l`

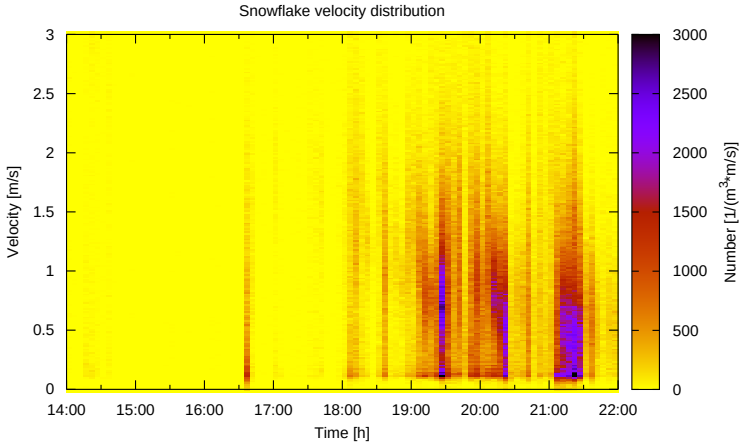

Figure C.6: Example output of `plotvideo.py`. This plot was produced using the following command line: `./plotvideo.py -p wakasa2003 -d vdistrmap -s "2003-01-28 14:00:00+09" -e "2003-01-28 22:00:00+09" -l --timeres=300 --cbrange=0:3000 --palette="negative"`

Bibliography

[1] Philip Eden, Clint Twist, “Weather Facts”, ISBN 0-7513-5322-1, Doring Kindersley, 1995.

[2] R. R. Rogers, M. K. Yau, “A Short Course in Cloud Physics”, 3rd edition, ISBN 0-08-034864-5, Pergamon Press, 1989.

[3] Kenneth G. Libbrecht, “Online guide to snowflakes, snow crystals, and other ice phenomena”, web site. `http://snowcrystals.com`. Cited May 24, 2004.

[4] Toshio Harimaya, “A Climatological Study on the Mechanism of Graupel Formation”, Journal of the Faculty of Science, Hokkaido University Ser. VII (Geophysics), Vol.8, No.5, pp.437–447, 1990.

[5] Horace R. Byers, “Elements of Cloud Physics”, Second Impression, ISBN 0-226-08697-6, The University of Chicago Press, 1973.

[6] Vitaly I. Khvorostyanov, Judith A. Curry, “Terminal Velocities of Droplets and Crystals: Power Laws with Continuous Parameters Over the Size Spectrum”, Journal of the Athmospheric Sciences, Vol.59, No.11, pp.1872–1884, 2002.

[7] Weather Underground, web site. `http://www.wunderground.com/`. Cited May 24, 2004.

[8] Ken-ichiro Muramoto, Kohki Matsuura, Toshio Harimaya, Tatsuo Endoh, “A computer data base for falling snowflakes”, Annals of Glaciology, Vol.18, pp.11-16, 1993.

[9] NNDC Climate Data Online, Description and Walk-through. `http://cdo.ncdc.noaa.gov/cdo/article-description.pdf`. Cited May 24, 2004.

[10] O. Kelley, J. Stout, M. Kafatos, "Content-based Browsing of Data from TRMM", 13th International Conference on Scientific and Statistical Database Management, pp.270-273, 2001.

[11] Johanneum Research 2D Video Distrometer home page. `http://www.distrometer.at/`. Cited May 24, 2004.

[12] Eszter Barthazy, Raphael Schefold, “A new ground-based optical instrument to measure snowflakes”, 30th International Conference of Radar Meteorology, AMS, München, 2001.

[13] David Atlas (editor), “Radar in Meteorology: Battan Memorial and 40th Anniversary”, ISBN 0-933876-86-6, American Meteorological Society, 1990.

[14] Henri Sauvageot, “Radar Meteorology”, ISBN 0-89006-318-4, Artech House, 1992.

[15] S. Y. Matrosov, “Radar Reflectivity in Snowfall”, IEEE Trans. GeoSci. Remote Sens., vol.30, no.3 pp.454–461, 1992.

[16] J. S. Marshall, K. L. S. Gunn, “Measurement of snow parameters by radar”, Journal of Atmospheric Sciences, Vol.9, pp.322–327, 1952.

[17] S. Y. Matrosov, “A Dual-Wavelength Radar Method to Measure Snowfall Rate”, Journal of Applied Meteorology, Vol.37, No.11, pp.1510–1521, 1998.

[18] S. M. Sekelsky, W. L. Ecklund, J. M. Firda, K. S. Gage, R. E. McIntosh, “Particle Size Estimation in Ice-Phase Clouds Using Multifrequency Radar Reflectivity Measurements at 95, 33, and 2.8 GHz”, Journal of Applied Meteorology, Vol.38, No.1, pp.5–28, 1999.

[19] C. Kummerov et al., “The Status of the Tropical Rainfall Measuring Mission (TRMM) after Two Years in Orbit”, Journal of Applied Meteorology, Vol.39, No.12, pp.1965–1982, 2000.

[20] Image Information Science Laboratory, Kanazawa University, Snowfall Team home page. http://wis.ec.t.kanazawa-u.ac.jp/research/sf/. Cited May 24, 2004.

[21] H. Servomaa, K. Muramoto, T. Shiina, “Snowfall Characteristics Observed by Weather Radars, an Optical Lidar and a Video Camera”, IEICE Trans. Inf. & Syst. Vol.E85-D, No.8, pp.1314-1324, 2002.

[22] Coordinated Enhanced Observing Period home page. http://www.ceop.net. Cited May 24, 2004.

[23] AMSR/AMSR-E validation program home page. http://www.ghcc.msfc.nasa.gov/AMSR/validation.html. Cited May 24, 2004.

[24] River and Environmental Engineering Laboratory, University of Tokyo, home page. http://aqua.t.u-tokyo.ac.jp/. Cited May 24, 2004.

[25] Meteorological Research Institute of Japan home page. http://www.mri-jma.go.jp. Cited May 24, 2004.

[26] Japan Aerospace Exploration Agency home page. http://www.jaxa.jp. Cited May 24, 2004.

[27] National Aeronautics and Space Administration home page. http://www.nasa.gov. Cited May 24, 2004.

[28] NASA Joins Snow Study Over the Sea of Japan, article on the NASA web site. http://www.gsfc.nasa.gov/topstory/2003/0122japansnow.html Cited May 24, 2004.

[29] Fukui University home page. http://www.fukui-u.ac.jp. Cited May 24, 2004.

[30] Sankosha Corporation home page. http://www.sankosha.co.jp. Cited May 24, 2004.

[31] Yamada Giken Corporation home page. http://www.yamada-giken.co.jp. Cited May 24, 2004.

[32] Japan Science and Technology Agency home page. http://www.jst.go.jp. Cited May 24, 2004.

[33] R. Elmasri, S. Navathe, “Fundamentals of Database Systems”, 3rd edition, ISBN 0201542633, Addison Wesley, 2000.

[34] T. Connolly, C. Begg, “Database Systems: A Practical Approach to Design, Implementation, and Management”, 3rd edition, ISBN 0201708574, Addison Wesley, 2001.

[35] M. Hernandez, “Database Design for Mere Mortals: A Hands on Guide to Relational Database Design”, 1st edition, ISBN 0201694719, Addison Wesley, 1997.

[36] MySQL database engine home page. http://www.mysql.org. Cited May 24, 2004.

[37] PostgreSQL database engine home page. http://www.postgresql.org. Cited May 24, 2004.

[38] GNU General Public License, version 2. http://www.gnu.org/licenses/gpl.html. Cited May 24, 2004.

[39] The BSD License. http://www.opensource.org/licenses/bsd-license.php. Cited May 24, 2004.

[40] Firebird database engine home page. http://firebird.sourceforge.net/. Cited May 24, 2004.

[41] MaxDB database engine home page. http://www.mysql.com/products/maxdb/. Cited May 24, 2004.

[42] Python programming language home page. http://www.python.org. Cited May 24, 2004.

[43] Gnuplot plotting tool home page. http://www.gnuplot.info. Cited October 2, 2003.

[44] A Teras, K. Muramoto, H. Aoyama, M. Tamura, T. Koike, H. Fujii, T. Pfaff, “Database of Snowfall for Analyzing Vertical Structure of Cloud”, Proceedings of the SICE Annual Conference, session TAII-2-6, pp.1878-1883, 2003.

[45] SYNOP Present Weather codes. http://www.weather.org.uk/resource/wwcode.htm. Cited May 24, 2004.

[46] Thomas Pfaff, “Remote Sensing of Snowfall and Cloud Liquid Water using Passive Microwave Radiometry”, Master’s Thesis, Department of Civil Engineering, University of Tokyo, August 2003.

[47] Arto Teräs, “Wakasa Database User’s Guide”, Image Information Science Laboratory, Kanazawa University, September 2003.

[48] Shridhar Daithankar, Josh Berkus, “Tuning PostgreSQL for performance”. http://www.varlena.com/varlena/GeneralBits/Tidbits/perf.html. Cited May 24, 2004.

[49] R.J. Boucher, J.G. Wieler, “Radar determination of snowfall rate and accumulation”, Journal of Applied Meteorology, Vol.24, No.1, pp.68-73, 1984.

[50] R.E. Carlson, J.S. Marshall, “Measurement of snowfall by radar”, Journal of Applied Meteorology, Vol.11, pp. 494-500, 1972.

[51] Y. Fujiyoshi et al., "Determination of a Z-R relationship for snowfall using a radar and high sensitivity snow gauges", Journal of Applied Meteorology, Vol.29, No.2, pp.147-152, 1990.

[52] Gunn, K. L. S. and J. S. Marshall, 1958: The distribution with size of aggregate snow particles, Journal of the Atmospheric Sciences, Vol.15, No.5, pp.452-461, 1958.

[53] Japan Student Services Organization home page. `http://www.jasso.go.jp`. Cited May 24, 2004.