OPTIMIZATION OF DATABASE STRUCTURE FOR HYDROMETEOROLOGICAL MONITORING SYSTEM

Ph.D. Robert SZCZEPANEK

Cracow University of Technology
Institute of Water Engineering and Water Management
ul.Warszawska 24, 31-155 Cracow, Poland
e-mail: robert@iigw.pl

Abstract

Recent development in automatic monitoring systems affected in exponential growth of collected data. Simple methods of storage and spatial analysis of hydrometeorological phenomena are not efficient when dealing with big datasets. This paper presents results of research on databases structure for collection of high temporal density data (10 minutes) from several sensors installed on monitoring stations. Number of stations (from 5 to 50) was fitted to conditions of medium size county in Poland. Analysis of different relational database structures for typical scenarios of data request is presented. Optimisation criteria include i.a. request time and database size. At the databases design phase two main factors should be taken into account. First one is time of request, related to crisis management when fast access to data must be ensured. Second aspect is database size, related to planning and research phase when wide scope of data should be accessible and time in not so important. Some methods of database efficiency improvement, not related directly to database structure, are also presented. Two databases were used for comparison (MySQL and PostgreSQL) and their efficiency for different scenarios was tested and described.
Keywords: relational database, hydrometeorological monitoring, crisis management, open source

1. INTRODUCTION

Monitoring of hydrometeorological phenomena in Poland and many European countries in twenty century was dominated by governmental institutions. Rapid development of automatic monitoring systems made them available also for local users. Scale of presented research corresponds to existing local monitoring systems in Poland at county and commune scale. Local government is the main users of such systems for crisis management purposes at different levels of decision making, but also private sector starts installing local meteorological monitoring systems. Probably in few next years, new local monitoring systems will be installed in many places, so information on possible threats and problems should help decision makers to choose the best solution. To make analysis useful also for other potential users (e.g. commercial, recreation sector) two different scales of monitoring installations were analysed – small network with 5 monitoring stations and big network with 50 monitoring stations.

Main core of technology for monitoring network construction is well known from years, but infrastructure for data collection (sensors, telemetry systems, data storage hardware) is just the first step. The second step is more difficult and covers management and analysis of gathered data. Few potential users of such monitoring systems have knowledge how the system will function in 1-5 years in terms of cost to efficiency ratio. Main goal of this paper is to demonstrate how different factors, like database structure and number of data collection points, influence efficiency of data storage and management. In stead of sophisticated methods like database engine tuning, simple methods of overall performance improvement are shown and discussed. In presented paper not optimisation *sensu stricto* is described, but search for optimal combination of available open source database software and database structure build on that software.

2. MATERIALS AND METHODS

As local user usually runs monitoring system on low- or middle-end infrastructure, research was done on Intel Pentium 4, 1.6GHz with 1GB RAM, WD Caviar 80GB hard drive (WD800BB) and Windows 2000 with Service Pack 4 as operating system.

To have comparable environment, several theoretical assumptions concerning data grow were outlined. Concept of research is based on real monitoring hydrometeorological systems working in Żywiec County and Kłodzko County in southern Poland. Monitoring networks consist of 28 and 39 stations respectively, with 2÷13 sensors installed on each station.

For this research purposes two types of hypothetical local networks were assumed:

- "small monitoring network" - 5 stations with 10 sensors each, and
- "big monitoring network" - 50 stations with 10 sensors each.

It means, whole monitoring network under analysis is between 50 and 500 points of data acquisition. Each sensor collects data with 10 minutes interval and measurement time precision is one second. Selected data types for this research were the following: precipitation depth, water table level, air temperature (3 types), air humidity, wind direction, wind speed, sun radiation and air pressure. Data gathering scenarios were prepared for three periods – one year (Y1), five years (Y5) and twenty five years (Y25). Those arbitrary selected periods correspond in my opinion to three stages of monitoring system life and were selected to present time-snapshot situation. From new system after one year of operation, to mature system after twenty five years with sufficient amount of data for hydrometeorological statistical analysis. There are of course existing and available in some places longer records of observations, but due to storage limitation, only databases smaller than 10GB could be analysed.

Next assumption is that all data from monitoring sensors are transmitted to one central system and stored in central database server. Database server is able to serve data both to local hydrometeorological models run locally and to remote users connected by web browsers. As reliable estimation of transfer time in real conditions is very complex problem (Lightstone et al. 2007), only time of internal database management system response to query was analysed. This is only part of the time needed to transfer data from database to client, but additional variables like script efficiency and band width could disturb measurement. Testing procedure was prepared in such a way to minimize data caching mechanism and its influence on total request time.

The newest versions of two leading open source relational database engines were tested and analysed: MySQL 6.0.2-alpha and PostgreSQL 8.2.5. As interface to PostgreSQL database, pgAdmin III program was used. In case of MySQL database, two tools provided by MySQL AB were used - MySQL Query Browser to calculate database time response to query and MySQL Administrator for database size estimation.

All monitoring data for this project were generated, but their physical properties of hydrometeorological phenomena were fulfilled. For every type of data, boundary values and specific "data behaviour" were defined. Air temperature for example was slowly, continuous changing. Accuracy of data under investigation also was defined individually. Hydrometeorological data were written into text files by program written in C especially for this purpose. Then data were imported into database system and designed queries were executed.

3. DATABASE STRUCTURE

To focus on main research problem, database structure contained only information critical from performance/size point of view. In real installations number of attributed related to objects (like station or sensor) is much bigger and includes i.a. spatial attributes used for spatial analysis of meteorological phenomena or for GIS visualisation.

All database relational structures and queries presented in this paper were tested on both database engines – MySQL and PostgreSQL. In the case of MySQL only traditional storage engine (MyISAM) was implemented in the test. Not providing support i.a. for transactions (Axmark et al. 2007), this type of engine enables faster database response to query and is still the most popular in web-based database systems. Modern and sophisticated storage engines like InnoDB or Falcon (Axmark et al. 2007) are not widely used and still MyISAM is the most popular one. Problem of optimal indexing is crucial in term of efficiency to database size ratio. In general, additional indexes

speed up query but significantly increase database size (Lightstone 2007). Indexing influence was shown on query example, not database structure level.

For this research purpose two the most frequently used relational structures (schemes) were tested. First database structure (S1) is oriented to monitoring stations. Data from one station are saved in one table, with columns corresponding to sensors (fig.1). This type of structure is variant and can change in time as new stations are run or new sensors are installed. Tables for measurement storage named st1 ... stn on figure 1, have *time* field as primary key which is indexed and contains detailed time when measurement was done. Whole process of database management is usually done with help of additional tables with metadata. External program must take care of table names and their structure – number and type of columns. In this structure columns types can be well tuned to data types.

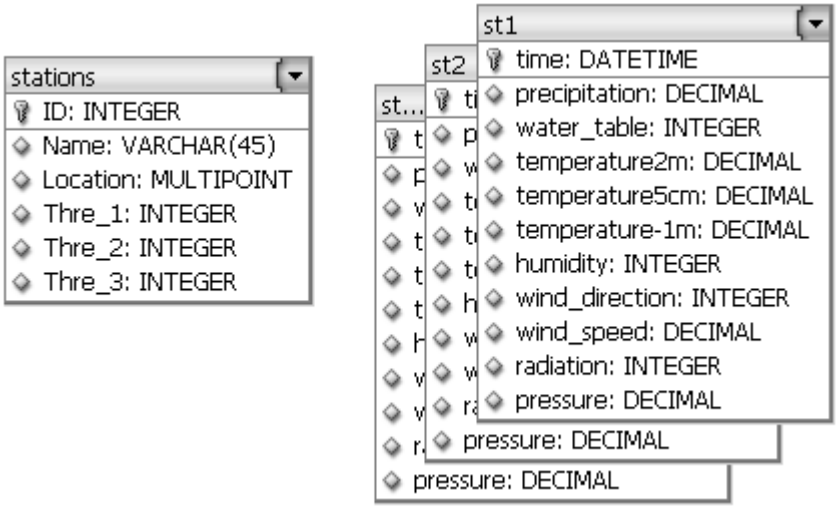

Figure 1. Database structure (S1) oriented to monitoring stations.

Second database structure is static - its structure does not changes in time (fig.2). This approach is closer to the traditional normalization process, and corresponds to 3rd normal form (Lightstone 2007). All measurement data are held in one table - *measurement*. Primary key is *time* field, but indexing is done on *time* and *sensor_ID* respectively. Information on sensors properties is located in related tables.

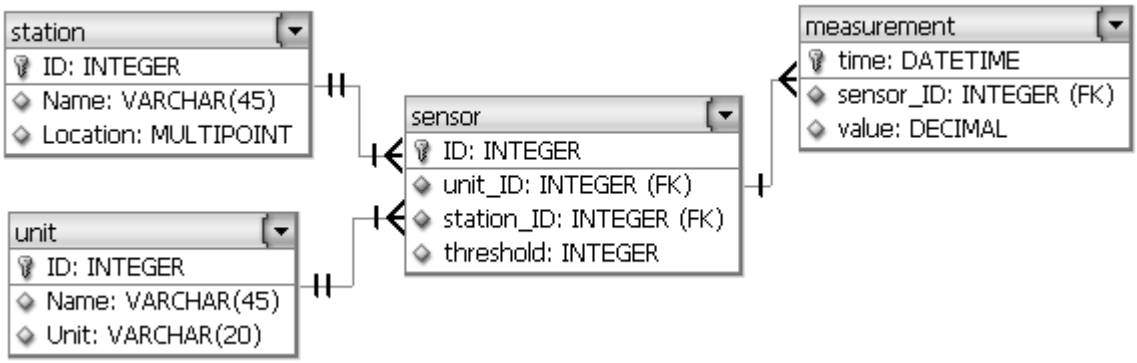

Figure 2. Database structure (S2) oriented to normalization.

Advantage of such structure is that management of such scalable, static structure is much simpler but because data from each sensor is saved in separate row and all observations are saved in one table, database size increases rapidly. For 50 stations during 25 years of observations number of records will be equal to 3 285 000 000! Additionally as there is only one field for all types of observations, *value* field (fig.2) must be wide enough to store different types of data.

4. TEST QUERIES

Three query types were analysed, focusing on three aspects of database functionality. Example SQL for each type is presented only for demonstration purposes and uses MySQL syntax (Axmark 2007) for S2 database structure. All queries are prepared in the simplest possible form. As result, no database specific implementation of SQL standard was tested. Using common syntax made however possible reliable comparison between MySQL and PostgreSQL.

Query Q1

Objective: search for data from all sensors for certain term.

This query uses key index and its goal was to analyse efficiency of indexing on the most frequently used field - time. Often used in crisis management when fast response for simple question is main issue.

Search for data from 2000-07-02 8:00:00

```
SELECT  * FROM measurement JOIN sensor ON measurement.sensor_ID=sensor.ID
WHERE time='2000-07-02 8:00:00'
```

Query Q2

Objective: search for data exceeding certain values.

This query searches for data not using indexing mechanism. Used both in crisis situation and in planning phase. Such *ad hoc* query can be very time consuming, especially when working with big tables. Used in operational mode when searching for locations of station in space, where threshold values are exceeded or values probably are wrong.

Search for air temperature at 2m exceeding 20.1 ºC

```
SELECT * FROM measurement JOIN sensor ON measurement.sensor_ID=sensor.ID
WHERE unit_ID=3 AND value>20.1
```

Query Q3

Objective: calculate statistics for selected, grouped data.

This is typical analytical query used on large databases. Run usually once and results are stored as new object.

Calculate average wind speed for every month-year of observations

```
SELECT YEAR(time), MONTH(time), AVG(value)
FROM measurement JOIN sensor ON measurement.sensor_ID=sensor.ID
WHERE unit_ID=8 GROUP BY YEAR(time), MONTH(time)
```

5. RESULTS AND DISCUSSION

Database size in case of big data sets plays important role in planning. Storage media prices are cheaper every year, but database size influences directly query performance. On figure 3 comparison of database size for different combinations of database engine, number of stations and database structure is shown. Please note that on all following figures values are presented in logarithmic scale. This shows how fast the increase of values under investigation is. Difference between minimal and maximal case are in the order of three levels of magnitude. After one year of observations database size vary between 10MB and 3GB. This depends mainly on monitoring network size (5 or 50 stations), but even for 5 stations the difference is in the order of 300MB. So database size for small networks can be 30 times bigger when implementing certain database and relational structure.

All PostgreSQL databases (with circles on figure 3) are much bigger than corresponding MySQL (squares) ones. Users with limited storage capacity should select MySQL. Database structure S1 (black) where data are divided into stations needs less space than structure S2 (silver). Approximately ten times more information can be stored in structure S1 than in S2 and this does not depend on database type – see overlapping graph for MySQL S1 with 50 stations and MySQL S2 with 5 stations.

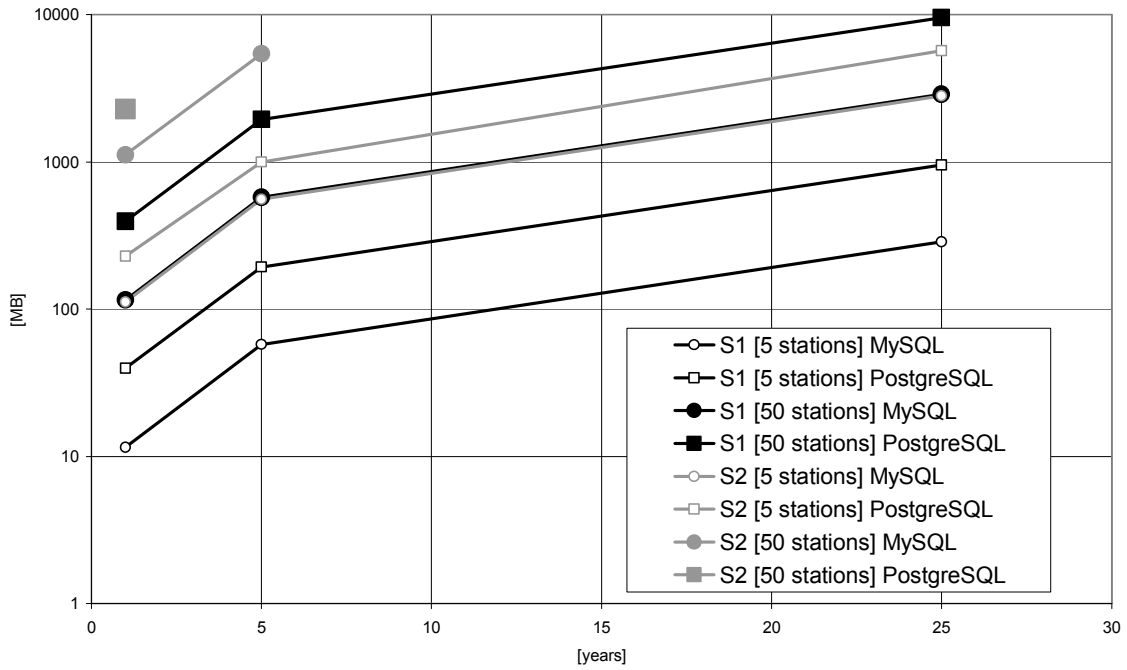

Figure 3. Database size as function of database structure (S1, S2), number of stations (5, 25), database type (MySQL, PostgreSQL) and observation period (1, 5, 25 years).

Increase in database size is quasi linear which is not so clear on figure 3 due to logarithmic vertical scale. So extrapolation of database size seems to be possible and reliable. For extreme analysed case (database structure S2, 50 stations, 25 year of observations) PostgreSQL database will be in the order of 100GB.

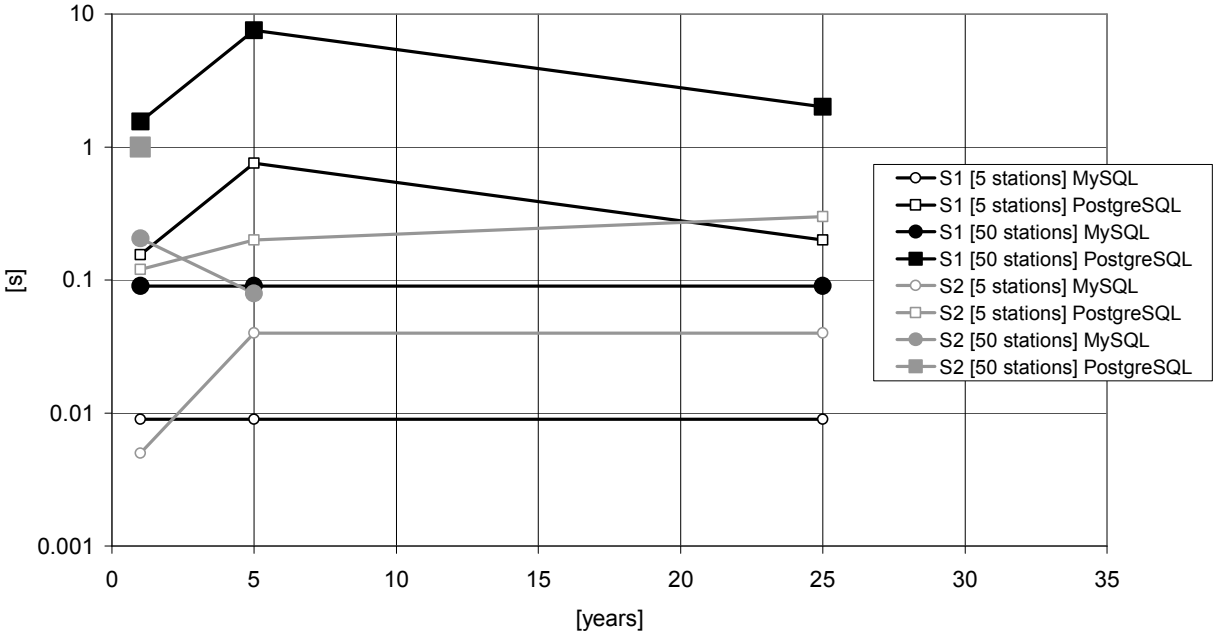

Figure 4. Time response to Query 1 as function of database structure (S1, S2), number of stations (5, 25), database type (MySQL, PostgreSQL) and observation period (1, 5, 25 years).

Second parameter under investigation was request time to query. First query Q1, which uses indexing mechanism gave in general results below 1 second (fig.4). It is acceptable value for real-time monitoring used in crisis management systems. Only requests for 50 stations saved in structure S1 from PostgreSQL were between 1 and 10 seconds. Request time on indexed fields is almost constant and does not depend on number of records for selected case. There is no difference in request time between one year observations and twenty five years observations. This shows how important and efficient can be indexing. Request time depend on selected database structure, database type and number of stations.

Query Q2 syntax is very similar to Q1, but in the case of Q2 we perform search on fields which are not indexed. Only MySQL database with structure S1 gives request time below 1 second in whole range of years, and request time remains almost constant (fig.5). For big dataset (25 years) MySQL request time is below 10 seconds while for PostgreSQL in most cases exceeds this value. Maximal query time for Q2 are close to 1000 seconds (corresponding to 16 minutes) and such values are unacceptable in crisis situations. Database structure for such purpose should be well designed with special attention paid to indexing of often used fields.

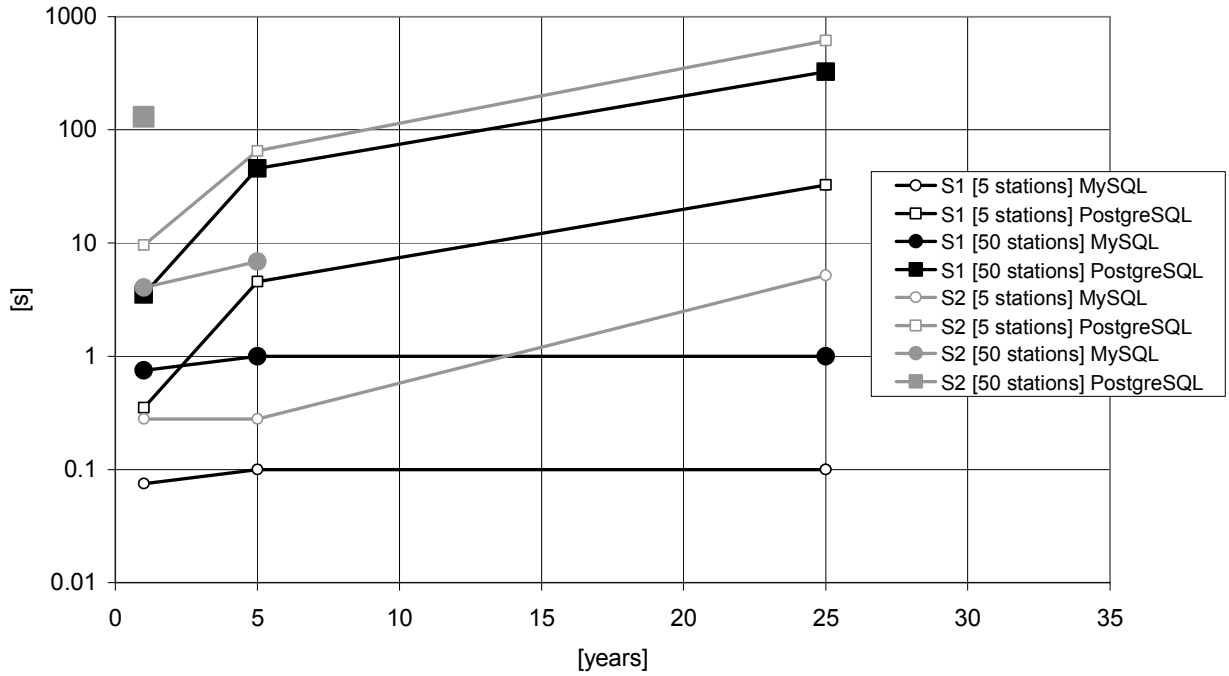

Figure 5. Time response to Query 2 as function of database structure (S1, S2), number of stations (5, 25), database type (MySQL, PostgreSQL) and observation period (1, 5, 25 years).

Last query Q3 has only analytical purpose so fast answer is not expected. One interesting observation is that for structure S2 and 50 stations results of MySQL and PostgreSQL are very similar (fig.6). Due to big database size it was impossible to analyse data for more than 1 year in PostgreSQL.

Query Q3 is much more complicated than Q2. It includes functions for date transformation and statistical calculation. But response times are similar except structure S2 for 50 stations in MySQL.

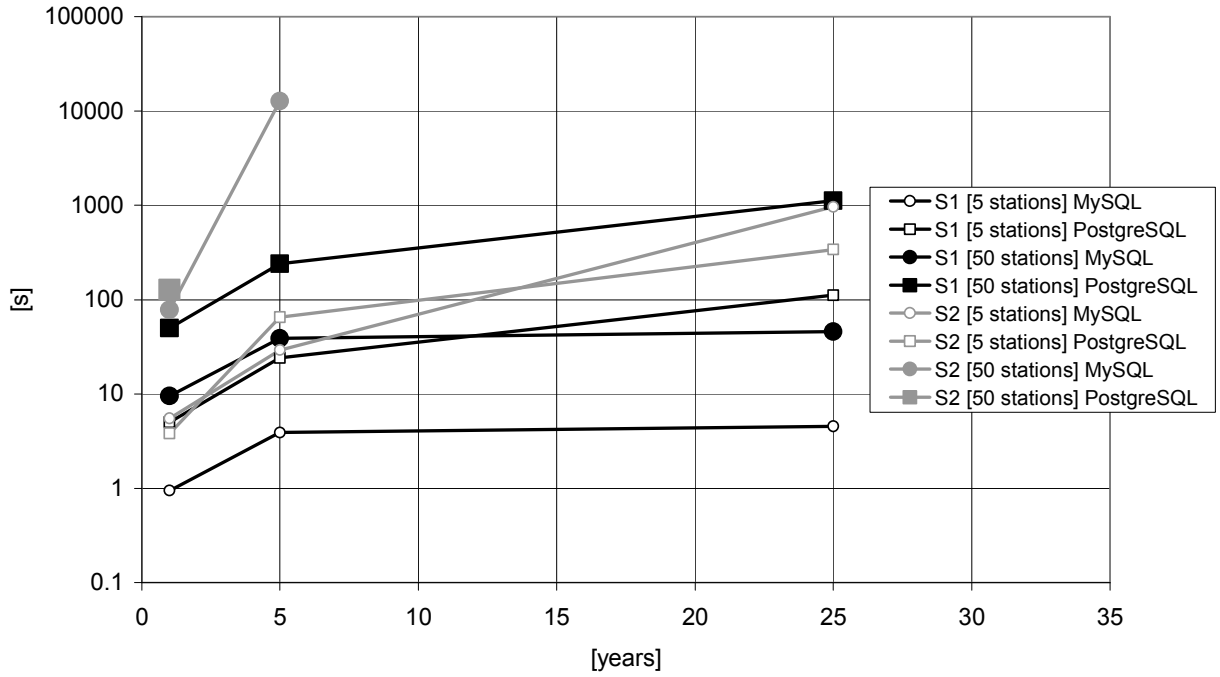

Figure 6. Time response to Query 3 as function of database structure (S1, S2), number of stations (5, 25), database type (MySQL, PostgreSQL) and observation period (1, 5, 25 years).

To overcome problems related to linear increase of response time to query several solutions can be implemented. First of all, data sets should be divided into operational part and analytical part. In high-end database systems all data are located in one warehouse but this needs a lot of space for data storage. Data warehouses are still too sophisticated and expensive for local government. Require also well trained staff to operate and support.

More pragmatic and cheaper solution is to operate only on recent data (e.g. one year) and take use of pre-calculated statistics from previous years held in separate database. This way there is fast access to new data from monitoring stations but at the same time historical data in form of statistical values are available. Possible solution is also fine tuning of database caching mechanism and database structure tuning with special emphasis on indexing.

To get an idea on what is relation between database size and its performance figure 7 and figure 8 were prepared. They both are based on one scenario where data are collected from 5 stations during 5 years of observations.

Expected database size is between 50MB and 1GB (fig.7). When analysing database type separately (MySQL – PostgreSQL), response time to query for all presented queries is directly related to database size (fig.7, fig.8). Bigger the database means longer time to access data. So first point in developing database should be related to minimisation of database size. This can be done be simple measures which are very often omitted – proper definition of field type and size strictly related to data be held. Next steps should focus on documentation of working database, especially aspects of performance and optimisation.

For all queries except Q3 it is clearly seen that response times of MySQL are two times shorter than from PostgreSQL. But one should remember that for this comparison old type of MySQL storage engine (MyISAM) was used which do not support typical nowdays functionality like transactions, which PostgreSQL tables used in this research does.

Both presented database structures S1 and S2 can be used, but there is significant differences in their performance. Implementation of S1 structure require more programming work because of dynamic nature of database structure. It should however provide smaller databases, better performance and easier from human point of view structure to read.

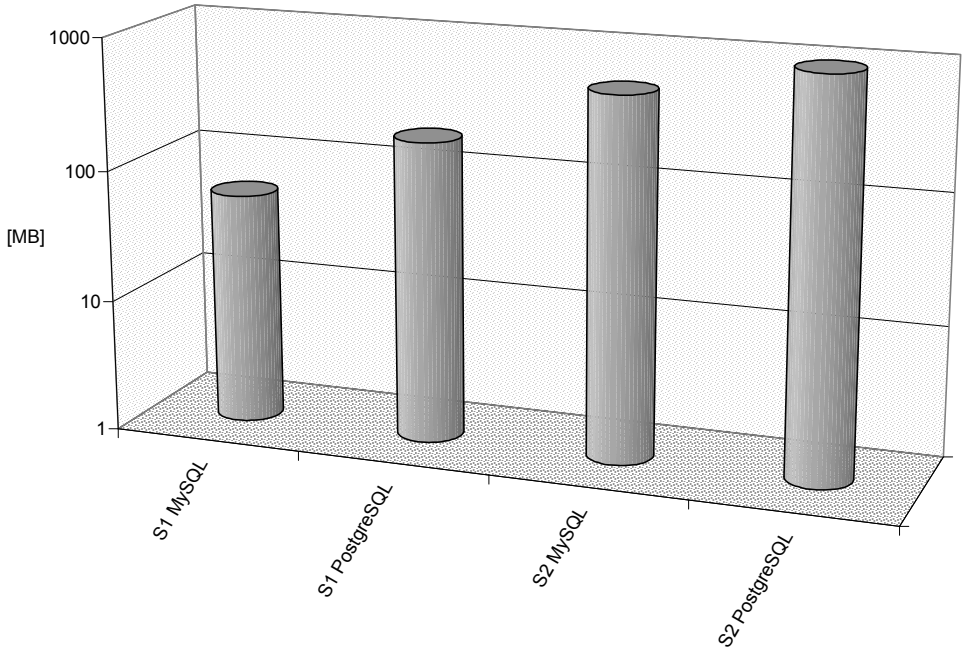

Figure 7. Database size for database structures S1 and S2 in MySQL and PostgreSQL (5 years of observation from 5 stations).

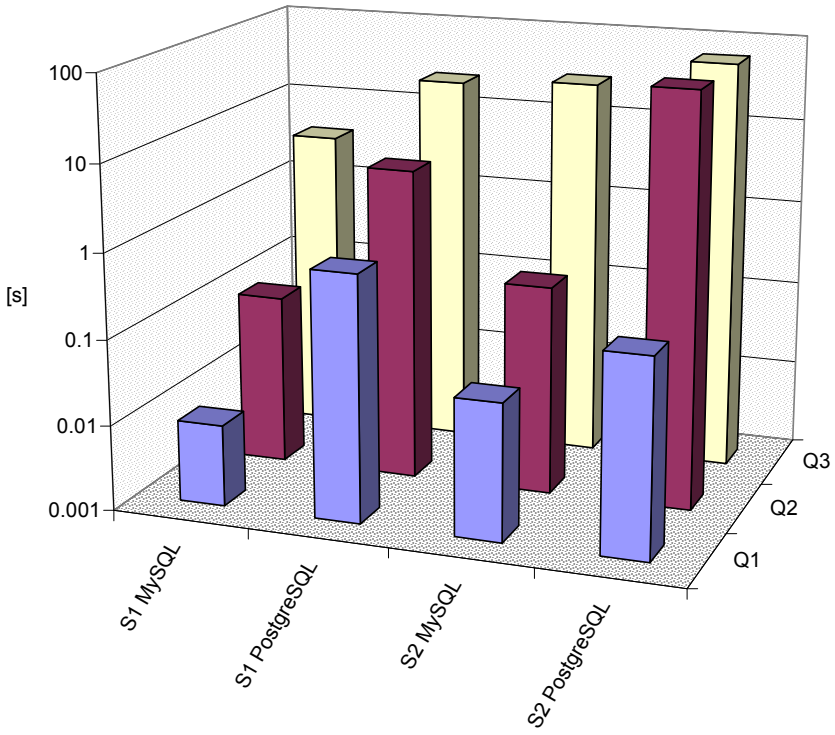

Figure 8. Database time response to query (Q1-Q3) for database structures S1 and S2 in MySQL and PostgreSQL (5 years observation from 5 stations).

In this research several variables which influence overall performance could not be tested. The most important are processor performance, memory size and hard disk performance.

MySQL was always synonym of simplicity and speed while PostgreSQL synonim of advances in database technology. MySQL is the main database used by web portals to fast deliver of required data. If well indexed, they can be very fast and can provide excellent speed/size ratio. For users looking for modern database, who do not care too much about space occupied by databases, PostgreSQL should be good solution.

6. REFERENCES

Douglas K. (2005) PostgreSQL (2nd Edition). Sams

Lightstone S., Teorey T., Nadeau T. (2007) Physical Database Design: the database professional's guide to exploiting indexes, views, storage, and more. Morgan Kaufmann

Welling L., Thomson L. (2004) PHP and MySQL Web Development (3rd Edition). Sams

Axmark D., Widenius M., DuBois P., Hinz S., Stephens J., Brown M., Lavin P. (2007) MySQL 6.0 Reference Manual. <http://dev.mysql.com/doc/refman/6.0/en/>

The PostgreSQL Global Development Group (2006) PostgreSQL 8.2.5 Documentation <http://www.postgresql.org/docs/8.2/interactive/>