Database for tropospheric product evaluations - implementation aspects

Jan Douša, Gabriel Győri
Research Institute of Geodesy, Topography and Cartography,
Geodetic Observatory Pecný
Ústecká 98, Zdiby 250 66
jan.dousa@pecny.cz

Abstract

The high-performance GOP Tropo database for evaluating tropospheric products has been developed at the Geodetic Observatory Pecný. The paper describes initial database structure and aimed functionality. Special focus was given to the optimizing effort in order to handle billions of records. Evaluation examples demonstrate its current functionality, but future extensions and developments are outlined too.

Keywords: troposphere, zenith delays, database, GNSS, radiosonde, meteorological data

1. Introduction

The potential of GNSS observations for troposphere monitoring has been described in Bevis et al. (1992). Since that time various projects aimed for developing GNSS-meteorology in Europe. First benchmark of near real-time ground-based GNSS tropospheric products – Zenith Total Delays (ZTD) – was provided within the COST Action 716 – Exploitation of Ground-based GNSS for Meteorology and Climatology (1999-2004). The extended routine analyses were supported by the EU FP5 project – Targeting Optimal Use of GNSS for Humidity (TOUGH, 2003-2006). Recently, the E-GVAP I-III (2006-2016) aimed for the establishing operational ZTD estimations and their assimilations in numerical weather models (NWM) and for developing active quality control for GNSS products. Additional effort on enhanced capability of GNSS troposphere monitoring and the exploitation of NWM data for GNSS positioning has been recently prepared and approved within the COST Action ES1206 – GNSS for Severe Weather Events and Climate (GNSS4SWEC, 2013-2017).

The Geodetic Observatory Pecný (GOP) analysis centre has contributed to the above projects since 2000 and provided one of the first operational GPS tropospheric products - near real-time regional GPS solution available officially since 2001 (Douša, 2001). Additional tropospheric products in support of meteorology have been developed at GOP during recent years – near real-time regional multi-GNSS product, 2011-present (Douša, 2012a), first near real-time global product, 2010-present (Douša 2012b) and real-time ZTD product, 2012-present (Douša et al., 2013). GOP routine post-processing tropospheric solution has been available also since 1996 for the part of the EUREF Permanent Network (EPN). The complete European network was reprocessed at GOP recently for the entire period 1996-2012. As being the most accurate and homogeneous during the whole interval, the reprocessing could be used in regional studies for climatology. The regular and long-term evaluation of all these products is an important task for both getting a relevant feedback about the accuracy and studying potential for improvements.

Initial comparison of GOP tropospheric products was done occasionally (Douša, 2003) using

PERL scripts for data stored in internal text format. With increased data period such design was recognized as inconvenient and it was replaced by a simple MySQL database used during 2002-2010 for GOP GNSS-based zenith total delays routine evaluations. Recently, more flexible database structure was requested for fully automating tropospheric data comparisons including the searching of nearby points, filtering, converting, interpolating, generating various statistics and their extracting for web-based plots. New database (labelled as *'GOP Tropo database'*) was required to provide a high-performance system in order to deal with billions of data records. As a free alternative to the enterprise solutions, the PostgreSQL server (POSGRESQL) was selected for this task and the database structure was completely revised in order to support its flexible utilization. The GOP Tropo database structure and functionality design is described in Section 2, the optimization aspects in Section 3 and initial comparison examples in Section 4. The conclusion and outlook is given in the last section.

2. Database design

This section provides a rough description of the data structure with the focus on specific details. The data organization is the most important aspect of any database since it predefines any future utilization (and its flexibility for extensions). The main GOP Tropo database features and functionalities were initially defined as well as requirements for future developments to:

- accommodate and compare different tropospheric data types in a single database,
- provide geo-referencing and collocated point searching for data comparisons and interpolations,
- apply vertical corrections (potentially supported with geoid and orography models),
- generate comparison differences and statistics in a yearly, monthly, weekly and hourly mode,
- support conversion data types (e.g. zenith wet delay to integrated water vapour and others in future),
- interpolate values from grid points and calculate grid points from available data (in future),
- study trends, temporal/spatial variations, correlations (in future).

Target and potential utilizations are foreseen such as a) to compare (near) real-time GNSS tropospheric products with respect to the final ZTD products, b) to assess GNSS results with respect to radio soundings, radiometers or numerical weather models and other independent datasets, c) to compare troposphere estimates from different space geodetic techniques (GNSS, VLBI, DORIS), d) to evaluate different strategies of interpolations or kriging of meteorological data or tropospheric delays, e) to evaluate in-situ meteorological observations provided in M-RINEX, f) to evaluate global pressure, temperature or specific tropospheric models in future.

2.1. Database structure

In any relational database the user data are placed in database tables, which structure is designed optimally for the purpose of the utilization. The data are organized as records (rep-

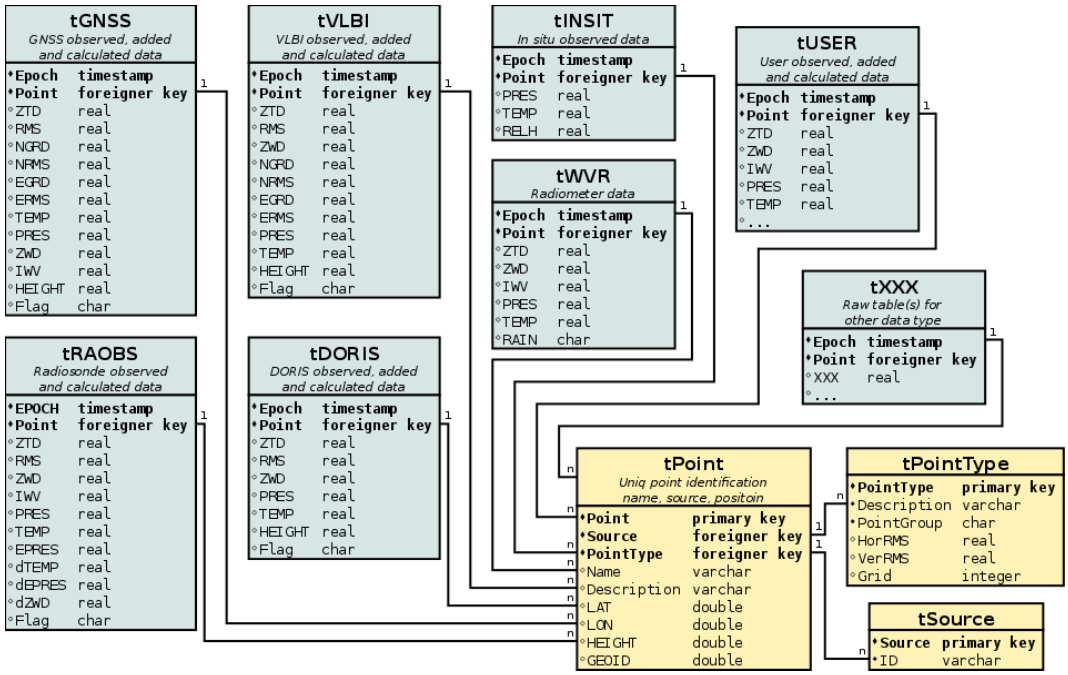

Figure 1: Basic structure of GOP Tropo database including original data representation.

resented by table rows) while each record has specific values (represented by table columns). The data should not be duplicated and, when any duplication occurs, new database table with relevant records is created to unify any dualities. Such unique record is then related to the original table record by using the relation provided with a reference key (that's why such database is called relational).

The GOP Tropo database is designed to accommodate different tropospheric meteo data types such as: GNSS, VLBI, DORIS, radiosondes, radiometers, synoptic data, in-situ meteorological data, data extracted from the NWM and other supporting data/models as e.g. geoid, orography. According to the variability of tropospheric products, meteorological data or other supporting data, it would not be easy and efficient to accommodate them within a single data table, because each data source has its own specific stored values. Fortunately, data from each source could be processed independently and it was identified as useful to define a specific data table for each data type. Different data sources for all data types are also considered, e.g. such as the GNSS ZTDs from EUREF, E-GVAP analysis centres, the International GNSS service and many others. Additionally, for a single analysis centre, e.g. GOP, various products are available too, such as final, reprocessed, near real-time (global, GPS, GPS+GLONASS), real-time or others. All the sources within a single data type are accommodated within a single table providing unique source identification for the data.

The common structure of the data organization within the database is shown in Fig. 1. All data are georeferenced according to the reference key to the *tPoint* table (*'t'* is always used to identify the table name), which provides additional information about the data source (*tSource*), site identification (*tSite*) and point location. Optionally geoid undulation and orography height is provided too. Each record in the *tPoint* table is uniquely defined by its name, source and position (latitude, longitude, height) along with the position accuracy used

Table 1: List of existing input filters (PERL) for data decoding and inserting in database

Input filter	Input format	Procedure	Remarks
tro-snx2DB.pl	Tropo-SINEX	fInsertGNSS	ZTDs, IGS/EUREF products
bsw-trp2DB.pl	Bernese TRP	fInsertGNSS	ZTDs, GOP products
cost-trp2DB.pl	COST-716	fInsertGNSS	ZTDs, E-GVAP
rt-flt2DB.pl	Tefnut output	fInsertGNSS	ZTDs, GOP real-time analysis
raobs2DB.pl	BADC profiles	fInsertRAOBS	Integrated data, radiosondes
wvr2DB.pl	Radiometrics	fInsertWVR	Integrated data, radiometers
met-rnx2DB.pl	Meteo RINEX	fInsertINSIT	In-situ meteo data (GNSS)
cost-met2DB.pl	COST Meteo	fInsertSYNOP	COST 716 meteorological data

in identification of a unique point.

Specific data tables are currently predefined for the data types – *tGNSS, tVLBI, tDORIS, tRAOBS, tWVR, tINSIT, tUSER, tGEOID, tSURF* and others where the name can help to identify the data content. Others specific data could be completed later, such as for mapping functions coefficients, slant tropospheric delays, synoptic data, NWP grid data (or more likely their reduction to the specific parameters at a reference surface) and other.

2.2. Database feeding, record uniqueness

The database filling is done in three steps: 1) data download, 2) decoding and converting original data including data preparation for SQL command calling a specific database insert procedure and 3) executing SQL command within GOP Tropo database. Input data are collected from various sources via the standard ftp or http downloads to a local disk, all in original and usually compressed formats. This process is controlled via unix cron job scheduler.

The data decoding and filtering is done by in-house developed input filters written in the PERL scripting langue. Their main tasks consist of the extracting and converting of data from text files and preparing (and executing) SQL commands calling the GOP Tropo database insert procedure. The input filters are designed for various input formats (e.g. for *tGNSS* it is Tropo-SINEX format, Bernese 'TRP' output, COST-716 format) and also specific database insert procedure is called for each data types (e.g. *tGNSS* uses *fInsertGNSS* procedure). The list of supporting formats and input filters is given in Tab. 1. The radio sounding profile is the example of data type, content of which is not fully included into the database, but only selected parameters derived at the surface (e.g. pressure, temperature, water vapour pressure, zenith hydrostatic delay, zenith wet delay, relevant lapse rates for possible vertical conversions etc.). The data decoding and database filling is also regularly started from a cron scheduler.

The internal database stored procedure for inserting is performed either via INSERT or UPDATE SQL command. The former command supports direct inclusion, but works for new records only. The latter consists of an initial check if the record already exists in the database and, if true, it replaces it with the current data. In this context it is important to understand how records are identified as unique within data tables. Database systems use

primary keys to specify the column (or columns) that uniquely identify each record, and these can be either natural or surrogate. A natural key is the one that is composed of columns directly related to the data, while surrogate key is a specific column added to the data table only for serving as a primary key (e.g. a unique identifier, auto-incremented numerical value). In many cases we use surrogate keys, but not for the main data tables, where the primary key is designed as a unique index over two columns – *Epoch* and *tPoint*. The epoch value is of the type 'TIMESTAMP WITHOUT TIMEZONE' handling commonly data and time. The point column refers to the record in the *tPoint* table, where the uniqueness is provided via auto-incremented numerical value. The record in the *tPoint* table is checked and updated automatically anytime when filling new entry into main data table. This is proceeded as follows – requested point is searched within the *tPoint* table and, if found, it returns the reference key and, if not, the new point is created. The uniqueness of the point records is thus managed within the point searching/inserting internal procedure (*fInsertPoint*) taking into account the unique point characteristics: *SiteID, Source, PointType* and *location* (latitude, longitude, height) within predefined *point accuracy* (horizontal and vertical).

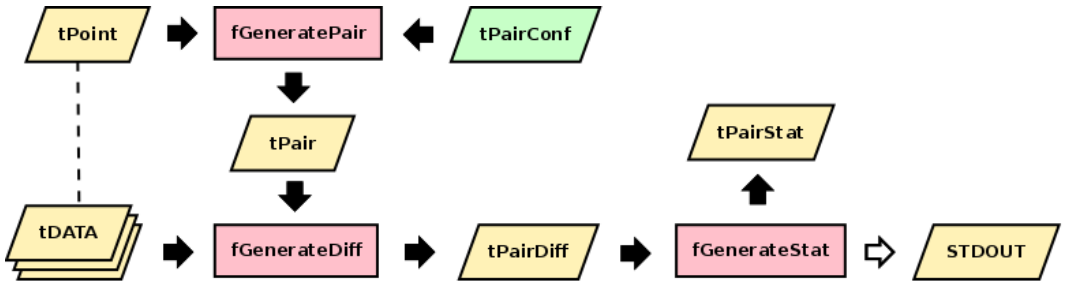

Figure 2: Scheme of the comparison within GOP Tropo database

2.3. Data and product comparison

The main functionality of the current GOP Tropo database implementation is the comparison of various tropospheric data and products. It consists of this sequence of steps (Fig. 2):

- comparison configuration (manual),
- search of collocated points (with respect to the setting criteria),
- generation of data differences for identified pairs,
- statistics over data differences,
- extraction and visualization (provided outside the database).

The user configures data or product comparisons by defining two data sources from one or two data tables. The setting consists of criteria for searching pairs of collocated points – the limit for the horizontal and vertical distance of two points. If applicable, the maximum σ is set for the initial data filtering and the limit of the confidence interval is provided for the statistical procedures to detect and reject outliers. Optionally, mask or explicit site list can be provided if selected stations should be compared only (implicitly all stations). Individual comparison settings are stored as records in the special database table (*tPairConf*) where a surrogate key is set for further referencing.

The candidates for identifying collocated points are searched within the *tPoint* table for the two specified sources from the setting. The station pairs are generated by the *tGeneratePair* procedure and the pair list is stored in the *tPair* table used afterwards for data differences generation (*fGeneratePairDiff*) within the period of request. Compared products have usually different sampling rates and the values for comparing could be generally referred to different epochs. For this case, the database supports sampling rate argument defining the interval within which all single product values are extracted and the mean value is calculated for the comparison. An optimum difference sampling step should be set up as the higher data sampling of both products in order to grab common values and, on the other hand, below one hour since the variation of the troposphere will be significantly smoothed by averaging product values over a longer time. The setup between 10 and 60 minutes is usually reasonable. In future, we consider supporting other functional fitting of the single product values for differencing instead of a simple averaging. Generated data differences are always stored in the *tPairDiff* table enabling to accommodate differences from recently collected data or analyzed products.

The statistical procedure (*fGeneratePairStat*) applies three iterations to estimate biases, standard deviations and root mean squares (r.m.s.) for each compared pair individually. The first iteration serves for the median estimation as a robust initial mean value. It is used in the second step for calculating differences reduced by mean value in order to estimate reliable standard deviation for the outlier detection. In the third step, final statistics – bias, standard deviation, r.m.s., number of all observation and outliers – are calculated after outliers were rejected using the confidence interval based on standard deviations from the second iteration and, optionally, data excluding r.m.s. limit from the setting. For individual comparisons, five statistic modes are provided for a period of request (defined by *'beg'* and *'end'* arguments) – all, yearly, monthly, weekly and hourly. The first one calculates the statistics over the requested period, the last for the same period but providing statistics for data filtered according to hour of day. All other modes calculate statistics individually for windows as specified within the whole period. Because the differences are saved in the *tPairDiff* table, all statistic modes can be efficiently repeated in a regular update for generating time-series as demonstrated in Fig. 3. Finally, the extractions and visualizations of results are described in the section with sample evaluations. Table 2 summarizes the comparison procedures. Note that each comparison procedure uses a key to the configuration record in the *tPairConf* as its first argument which has been omitted in the table.

Table 2: List of comparison procedures/functions (*'f'*) input/output tables (*'t'*)

Procedure	**Arguments**	**Input**	**Output**	**Remarks**
fGeneratePair		tPairConf	tPair	generate pairs
fGeneratePairDiff	*'beg','end','sample'*	tPair,tData	tPairDiff	generate differences
fGeneratePairStat	*'beg','end','type'*	tPairStat		generate statistics

2.4. Other procedures for data maintenance

Since data are structured using various database tables, any of the removing, viewing, extracting and statistic operations require more complex SQL commands, which are implemented as specific stored procedures in GOP Tropo database. According to the relationships between

Table 3: List of other procedures for database content maintenance

Procedure	**Arguments**	**Category**	**Remarks**
vPoint	*'source','site','description'*	View	List filtered point list
vSource	*'source'*	View	List filtered source list
vSitel	*'source','site','description'*	View	List filtered site list
vData	*'source','site','description'*	View	List filtered data
vDiff	*'source','site','description'*	View	List filtered differences
fDataInfo		Info	Information on data table
fPairInfo		Info	Information on pair tables
fStatInfo		Info	Information on statistics tables
fDeletePoint	*'source','site','description'*	Delete	Remove selected data and site
fDeleteSite		Delete	Remove unreferenced sites
fDeletePair	*'source','site','description'*	Delete	Remove pairs and differences

records in various data tables, such procedures combine data for a transparent and user-friendly output. Additionally, for easy data selection, three common arguments for name or mask input (via a simple regular expression using asterix) are supported for most common maintenance procedures. Table 3 provides an incomplete list of maintenance procedures with respect to three category operations: view, delete or info.

While view procedures are designed to extract specific data combinations from various tables, the info procedures extract general information about table contents – e.g. start/end of data, number of records etc. All these procedures are, however, only minimum implementations to simplify common queries while any other specific query can still be requested using a standard SQL command.

On the contrary, delete procedures could be very tricky, in particular due to a possibility to lose a consistency within tables and their relations. Any removal is implicitly driven by the 'DELETE CASCADE' attribute used for most tables. The attribute specifies that when referenced row is deleted, row(s) referencing it should be automatically deleted as well. However, the cleaning of any specific point-/pair-related data or differences should be done with the relevant delete procedure. For cleaning sites in the *tSite* table which are not anymore referenced a specific stored procedure exists too.

Table 4: List of selected important settings from *postgresql.conf* configuration file

Name of variable	**Value**
shared_buffer	3000 MB
work_mem	512 MB
max_connections	10
maintenance_work_mem	256 MB
default_statistic_target	300
effective_cache_size	5000 MB
constraint_exclusion	partition

3. Database optimization

The database is running on a dedicated server with GNU Linux operating system Debian 6.0.7. and, currently, has reserved 12Gb memory and 8 thread 64-bit Intel(R) Xeon(R) CPU. This hardware configuration is sufficient for current operations. However, more than 750 millions of records are stored in each of the largest tables – tGNSS and tRAOBS – and there are more than one billion records total in the database, utilizing almost 100Gb of hard drive space. This amount of data can cause lack of performance and rapidly decrease execution speed of some SQL queries. The highest performance can be reached by optimal configuration of several parameters which are stored in *postgresql.conf* settings file, because they are set to extremely low values by default and, in most cases, they are not related to the available hardware configuration. For that reason, we discuss recommended and applied settings in the next paragraph.

The first important parameter – *work_mem* – defines the maximum limit of memory which can be used for one sorting operation. The amount of memory usage increases with each additional sorting. Each client connected to the same database server typically uses a maximum of two sorting operation at one time. This implies that the value of *work_mem* could be set to the amount of unused memory divided by the maximum number of connections while divided by two. The number of connections can be reduced in *max_connections* setting. When sorting, PostgreSQL estimates first the amount of memory for possible use. If *work_mem* value is not high enough, the system will use swap operations along with free space on the hard-drive, decreasing the performance. The parameter *shared_buffer* then determines the maximum amount of memory which can be used for cached data in memory after they are read from the hard drive. The higher value, the bigger the set of data is possible to store in memory, which reduces the number of swap operations. Optimal value is quite difficult to set, but 30% of the available memory is recommended for a dedicated server.

The VACUUM operation is one of the most important commands in PostgreSQL database server. It is a kind of garbage collector which releases allocated space from invalid records. The VACUUM operation is closely related to the ANALYZE operation which is used to collect statistics for optimizing a server query plan. The automatic maintenance of database is provided by the AUTOVACUUM command, which is turned on by default. The *maintenance_work_mem* parameter in the setting then defines the amount of memory which can be used for AUTOVACUUM. The last important parameter is *default_statistic_target*, value of which determines the quantity of data used for statistics. The higher the value, the higher the CPU utilization generated by server. On the other hand, a higher value of *default_statistic_target* likely yields more precise statistics and thus possibly increase the speed of the next SQL query. Table 4 summarizes the important setting of the PostgreSQL configuration file.

As it has been already mentioned above, the biggest table in GOP Tropo database contains more than half a billion of rows, which can increase easily when new data are introduced (e.g. such as from E-GVAP project). The SQL queries usually work with data in specific time range, for example, generating monthly statistics between different products. The PostgreSQL supports basic table partitioning while splitting single large table into several smaller pieces. This is done by applying an inherited scheme, which defines a parent table similarly as the original table, while data are stored in a sequence of child tables (partitions). Such

partitioning can rapidly improve system performance, in particular when the query works with data in a single partition. On the other hand, additional overheads are relevant to all 'INSERT' commands calling a special trigger function when the table is partitioned. The inherited implementation guarantees that the partitioning has no direct influence to scripts or applications used within the database since data stored in child tables inherits behaviour from the empty parent table. In GOP Tropo database, range partitioning was applied in a yearly scheme on all large tables and each *tGNSS* child table thus contains about 20 millions records on average. New partitions are created by specific trigger PL/SQL functions called when data are inserted in the table. Table 5 summarizes the execution time of computing average value from data stored in a single partition restricted by WHERE clause. It is obvious that such query is much faster on the partitioned table than on the table without partitioning.

```
EXPLAIN ANALYZE SELECT AVG(ztd) from tGNSS as t1
where t1.epoch >= '2007-01-01 00:00:00' AND
          t1.epoch < '2008-01-01 00:01:00'
AND  t1.FK_Point = XX;
```

Table 5: The statistics from the partitioning test via analyze function for specific (repeated) command

Sequence number	**Station**	**Source**	**Execution time (partitioned table)**	**Execution time (single table)**
First	GOPE	EUREF Repro1	1363 [ms]	1300 [ms]
Second	GOPE	EUREF Repro1	13	48
Third	GOPE	EUREF Repro1	13	48
First	ALBH	IGS Repro1	47	125576
Second	ALBH	IGS Repro1	0.1	4422
Third	ALBH	IGS Repro1	0.1	4422

The database performance was tested on several different clusters as defined in Tab. 6. The original cluster is labelled as A and to this variant additional optimization steps were applied sequentially. In the first step the original file system (ext3) was replaced with newer file system (ext4) to improve swap operations (cluster B). The PostgreSQL setting was then revised according to that described above (cluster C). The *tGNSS* table was divided into yearly partitions (cluster D). Monthly partitioning produced a lot of tables and this variant was rejected from the comparisons. The *tPairDiff* table was also partitioned applying the yearly scheme (cluster E). The comparison averaging interval was decreased from 60 to 10 minutes (cluster F). The PostgreSQL was updated from 8.4 to recently newest 9.2 version (cluster G) with applying original settings and, finally, the settings was tuned also in PostgreSQL 9.2 in a similar way as for the version 8.4 (cluster H).

Table 6 summarizes the settings and performance of tested clusters. The statistics clearly show the importance of PostgreSQL configuration tuning, which improved overall performance by a factor of 2-4, in particular for all highly time-consuming procedures (e.g. inserting). That is true for the old as well as the new system version. The new file system also did not provide any improvements as expected, but even a slight degradation. In this general performance test, partitioning did not cause an increased performance when sets of procedures applying more complex SQL queries were used. This was not expected, but in general we decided to

keep partitioning since only very small degradations and overhead costs were found. Further benefit can be expected for new PostgreSQL releases as well as in with growth of the data in database. The latter was the primary argument for preserving partitioned tables for large datasets.

Table 6: Variant settings of the database optimization and their performance

Cluster	File system	Partition data/diff	PostgreSQL vers/tuning	Compared samples	Insert [s]	Total [s]	Diff [s]
A	ext3	1/1	8.4 / no	60 min	4712	1130	835
B	ext4	1/1	8.4 / no	60 min	5050	1361	864
C	ext4	1/1	8.4 / yes	60 min	1348	944	660
D	ext4	Yearly/1	8.4 / yes	60 min	1663	1324	1033
E	ext4	Yearly / Yearly	8.4 / yes	60 min	1653	1377	1079
F	ext4	Yearly / Yearly	8.4 / yes	10 min	1637	7855	7563
G	ext4	Yearly / Yearly	9.2 / no	60 min	5759	1568	1228
H	ext4	Yearly / Yearly	9.2 / yes	60 min	1739	1554	1207

4. Evaluation examples

In order to demonstrate the initial database functionality, we provide several samples from routine evaluations with a focus on GNSS tropospheric products comparisons. The following products were used in the demonstration figures:

- IGS operational and re-processed (Repro1) tropospheric products (Byram, 2012),
- EUREF combined tropospheric results from the operational and re-processed (Repro1) solutions (Soehne and Weber, 2005),
- GOP global near real-time tropospheric product (Douša 2012b),
- GOP near real-time GPS and multi-GNSS(GPS+GLONASS) tropospheric products (Douša, 2012a).

We do not intend to go into details in the examples and we do not thus discuss details about product differences, e.g. applied software, processing strategy, models, constraints, precise products and many others affecting the ZTD solution. It is out of the scope of this paper to study effects of various changes clearly visible in statistics. Sample comparisons demostrate the calculated biases and standard deviations, either for the entire period or within period split into regular discrete intervals (e.g. years, months, weeks). The statistic results are provided with the database procedures for predefined configurations and the results are extracted from the database and visualized with Gnuplot (Williams and Kelley, 2011) or GMT (Wessel and Smith, 1998) plotting tools. Various figures are generated – total bias and standard deviations for the whole period and all common stations, the geographical distribution of the values or time-series of mean statistics over all stations from the comparison. The samples are given in Figs. 3-6.

Figure 3 shows the assessment of the GOP near real-time multi-GNSS ZTD solution with respect to the post-processed EUREF combined ZTD product over three months in 2011. For

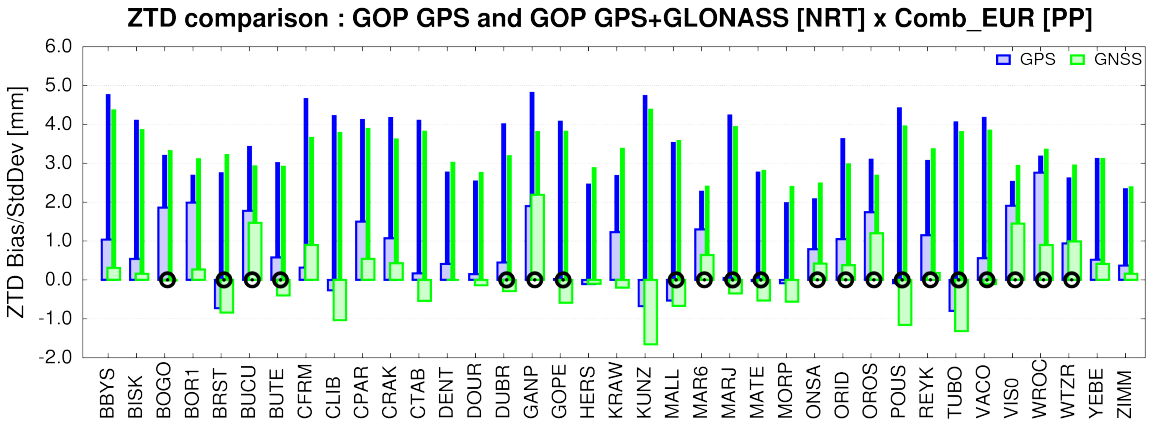

Figure 3: Example total bias and standard deviations for GOP GPS and GNSS ZTD near real-time solutions with respect to EUREF combined ZTD solution (three months in 2011). The circles indicates multi-GNSS stations.

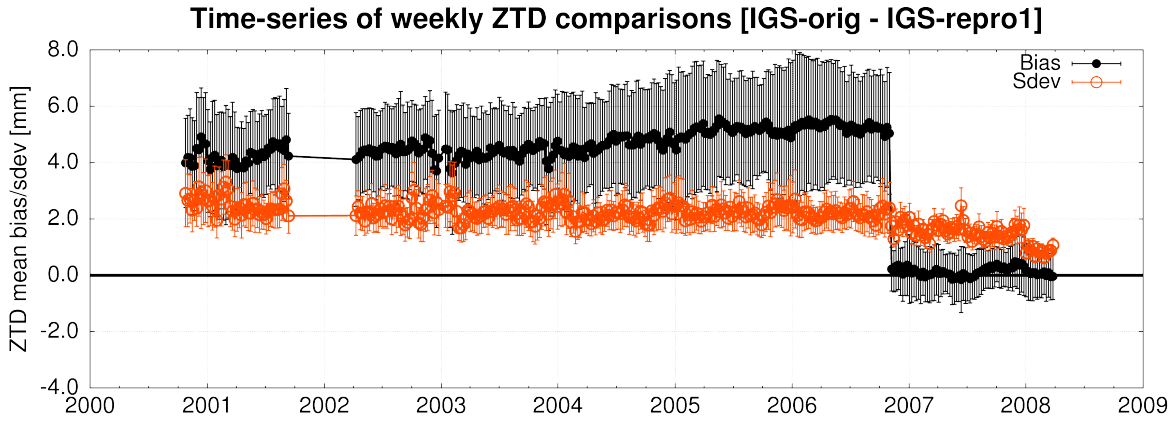

Figure 4: Weekly mean and its r.m.s. of ZTD biases and standard deviations from all stations

each station, which was identified as common to both products, the bias and standard deviation is calculated and plotted. Such plot provides information about the internal accuracy of GNSS ZTD products on a station by station basis. This is useful to assess a consistency of various strategies (and models) for comparison pairs when different product timeliness, input products, GPS or multi-GNSS observations and others are used.

Figure 4 and 5 show time-series of a long-term comparison of two products on a weekly and monthly intervals, respectively. The biases and standard deviations were calculated as mean values over all common stations for each interval individually so that any effect common to all stations can be visualized. Additionally, r.m.s. of such mean is calculated and plotted. Figure 4 compares historical and homogeneously reprocessed IGS ZTD products. The evolution of models and strategies within the historical product are clearly seen when compared to the re-processed ZTDs using the same strategy and models over entire period.

Figure 5 compares two different re-processing products – IGS (global) and EUREF (regional) while common stations in Europe could be compared only. Although both products are

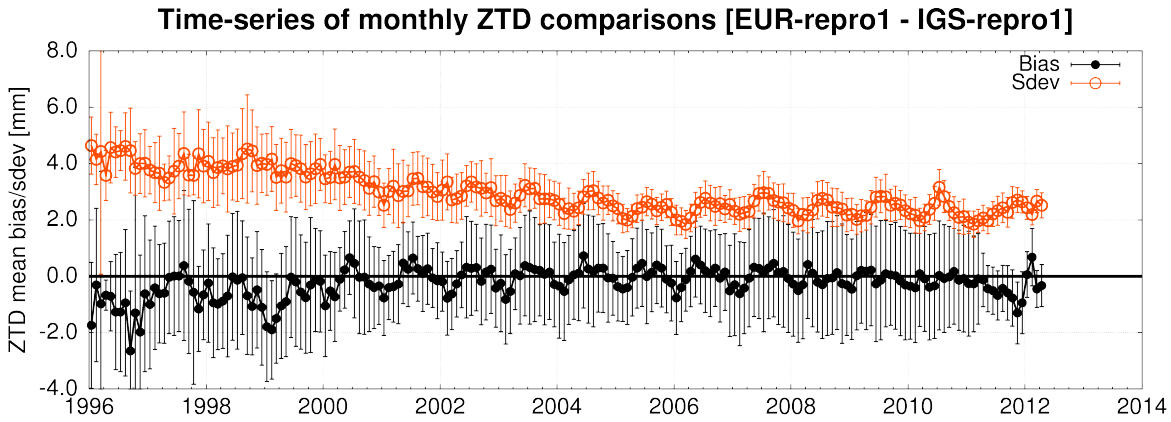

Figure 5: Monthly mean and its r.m.s. of ZTD biases and standard deviations from all stations

consistent over all time, the standard deviation clearly shows improvement in time, which can be attributed to the steadily increasing quality of data and products when more permanent stations in global and European networks are involved.

Finally, Figure 6 shows a comparison of IGS Final and IGS Repro 1 ZTD values for all dates covered by Repro 1. (Whereas Fig. 4 compared IGS Final and IGS Repro 1 ZTD estimates according to date, Fig. 6 compares them according to site.) The ZTD standard deviation is typically lower (about 2 mm) at low North hemisphere as well as in low latitudes in general. The effect of isolated stations, e.g. in central Asia, Africa and oceans, could be easily identified from the figure with standard deviations up to 4-5 mm. This can be attributed to the effect

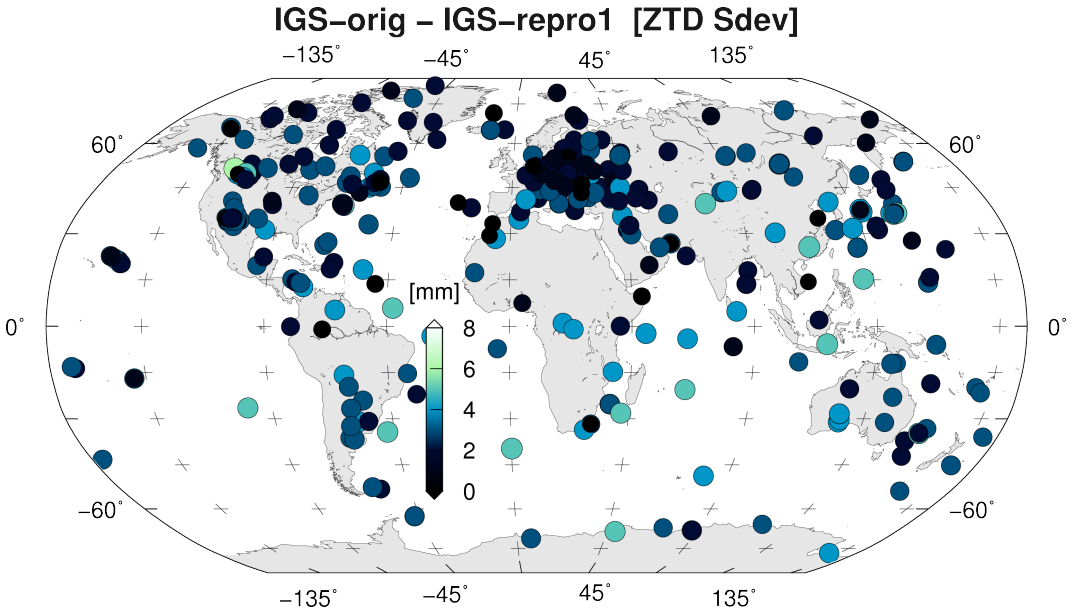

Figure 6: Displays geographical representation of ZTD standard deviations from two IGS global solutions

of lower accuracy of precise global orbit and clock products due to the lack of contributing stations, in particular during 90th.

5. Summary and outlook

The developments of the highly performance GOP Tropo database for the evaluation of tropospheric data and products were described with a special focus on its implementation aspects. The structure was designed in a flexible way to fulfil requirements specified in the introduction. Although the initial and primary motivation aimed for routine comparisons and evaluation of GNSS tropospheric products, current implementation functionality already goes beyond this scope. A special effort was given to the database optimization to support billions of records, which is already easy to achieve with several products. The optimization shows the need for revision of PostgreSQL default settings which could improve the overall performance by a factor of 2-4. Although careful partitioning did not improve performance, it was decided to keep it for the future since it is expected to become very important in handling huge quantities of data. Finally, samples of results were provided in order to demonstrate currently implemented functionality on selected interesting examples.

Acknowledgements

The database development was supported by the Czech Science Foundation (No. P209/12/2207). The authors also thank to Dr. Christine Hackman from U.S. Naval Observatory and two anonymous reviewers for useful comments improving the manuscript.

References

[1] Byram, S., Hackman, C. and Tracey, J. 2012. Computation of a High-Precision GPS-Based Troposphere Product by the USNO. Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011). 2012, Portland, OR, September 2011, pp. 572-578.

[2] Douša, J. (2001): Towards an Operational Near-real Time Precipitable Water Vapor Estimation, Physics and Chemistry of the Earth, Part A, 26/3, pp. 189-194.

[3] Douša, J. (2003): Evaluation of tropospheric parameters estimated in various routine analyses, Physics and Chemistry of the Earth, 29/2-3, pp. 167-175.

[4] Douša, J. and G.V. Bennitt (2012): Estimation and evaluation of hourly updated global GPS Zenith Total Delays over ten months, GPS Solutions, Springer, Online-First.

[5] Douša, J. (2012): Development of the GLONASS ultra-rapid orbit determination at Geodetic Observatory Pecný, In: Geodesy of Planet Earth, S. Kenyon, M.C. Pacino, U. Marti (eds.), International Association of Geodesy Symposia, Vol 136, pp.1029-1036.

[6] Douša, J., Válavovic, P. and Gyori, G. 2013. Development of real-time GNSS ZTD products. presentation at the EGU 2013 General Assembly, April 7-12, 2013.

[7] Williams, T. and C. Kelley (2011). Gnuplot 4.5: an interactive plotting program. URL: http://www.gnuplot.info.

[8] POSTGRESQL, http://www.postgresql.org/docs/8.4/static/reference.html.

[9] Soehne, W. and G. Weber (2005): Status Report of the EPN Special Project "Troposphere Parameter Estimation", EUREF Publication No. 15, Mitteilungen des Bundesamtes fuer Kartographie und Geodaesie, Vienna, Austria, Band 38, pp. 79-82.

[10] Wessel, P. and W.H.F. Smith (1998): New improved version of the Generic Mapping Tools Released, EOS Tans. AGU, 79, 579.