BRIEF DEVELOPER'S CONCEPTION OF INFORMATION SYSTEM FOR SOFT REAL-TIME PROCESSES

Case study included

J. Polnický, M. Kmínek, Z. Kokoška

Institute of Chemical Technology (ICT) Prague, Department of Computing and Control Engineering
Technická 5, 166 28 Praha 6 - Dejvice, Czech Republic
Tel.: +420 224355021; Fax: +420 224355053
E-mail: Jan.Polnicky@vscht.cz; Web: http://uprt.vscht.cz/polnicky

Abstract: This paper discusses some of the issues related to preliminary analysis, design and other tasks of the **open information system conception for soft real-time processes**, mainly about database system and database structure concepts for on-line process data archiving & process control. As some of the design-phase problems were not well summarized in the publications, the analysis of the process character had to be carried out in the first phase. The objective of this project was to suggest database system and develop suitable relational database structure for the semi-industrial unit - climbing film evaporator. The development of the database structure for this unit is presented, as well as the behavior of the system observed in experimental tests.

Keywords: soft real-time process, information system, database system & structure, RDBMS

Introduction – today's TIS

There are plenty of complex technological-information systems (TIS) on today's market. These systems are covering all needs of industrial automatization like direct controlling; deflect diagnostics; process data gathering, visualization and archiving. Heart of TIS is **real-time process** and also **industrial databases**.

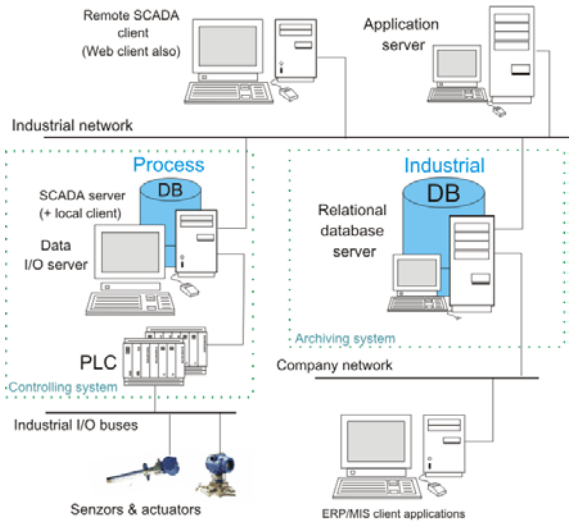

Fig. 1. Generelized architecture of TIS

Databases together with planning and administrative systems from ERP/MIS application group provide vertical flow of information. However, these robust TIS are very expensive and totally integrated, which means unsuitability for using on campus environment. For student's understanding of the merits of the case, we have to have a possibility to decompose the system. Sometimes we can found ourselves in situation, when our TIS has implemented measuring system and visualization&controlling system from SCADA/HMI application category, but it hasn't got system for process data collecting and archiving.

Main questions, which we'd like to clarify, are:

- How to select suitable combination of database software, operating system and hardware?
  - RTDBMS, Oracle or MS SQL?
  - Unix/Linux or Windows? SCSI or IDE?
- What do we know about information's journey from sensor places to database?
- Which information from process do we want to store in the database?
- What algoritms associated with database do we want to use?
- How to design and create physical structure in database environment?

Conception overview – Databases in TIS

Suitable combination of database software, operating system and hardware

Let us start with **hardware** (HW) requirements. If we are building a server from components, we have to construct fast, reliable and stable framework – we must use branded articles. Particulary the database server is very demanding on speed and system memory space. Next to finest mainboard, fast processor(s) and large system memory, we have to ensure fast I/O harddisk operations. For example, using RAID controllers – creating RAID arrays from fast SCSI or slower IDE harddisks, can provide it.
For **operating system** (OS) we have just two possibilities: Linux or Microsoft Windows. If we want reliable, stable and open operating system, we probably choose one of attested distribution of Linux, which generally consumes less resource than Windows systems. Windows has a couple of advantages: their compatibility and user-friendliness.
And what about selection of database? We will discuss this in a small nested chapter…

What do DB, DBMS, RDBMS and RTDBMS acronyms mean?

Standard **database system (DS)** is a specific kind of information system consisting of database management system (DBMS) and its own database (DB). DBMS designates rules and data structure in whole database (datafile – datastore) and it is providing access to and ability of data manipulation. In developer's point of view there are two database systems: **file** and **client/server** databases:
For *file-databases*, application spooling files on local or remote file system using internal database libraries. This type of databases is hardly accessible for several parallel users at the same time, because there is no co-ordinative process. One of advantages of file-databases are rapid I/O data operations.
For *client/server databases*, application communicate with superior process. This process coordinates access of individual users and ensures data integrity and system security. The most frequent DSMS is **relational database management systems** (**RDBMS**, known also as SQL databases), which are using relational data model for storing information to organized structures – relational tables. Classical RDBMS process data in systematic and organized structure dependent on specific applications. They are prioritizing logical data consistency as the only one criterion. To guarantee data consistency, databases are using locking procedures for reading (selecting) and writing (updating) raw information – procedures are blocking other database transactions or database users.
There are also **real-time database systems (RTDBS)** for very fast, real-time processes. RTDBS is closely specialized database system, which has big requirements on performance and accuracy. In connection with RTDBS, we are talking about "*critical terms*". Database translations are evaluated according to occurrence of *lost transactions*, which missed their critical terms. Not only logical evaluation, but also time of translation gives evidence of translation correctness. Real-time processes can be divided to **hard** and **soft real-time systems**. In case of hard real-time systems, we can't allow any *lost transaction*, in contrast to soft real-time systems. In any case, we can reuse same words as for operating systems: DS has to be fast, reliable, stable, open and compatible.
Generally, classic relational database systems *aren't* suitable for real-time process data archiving, *but* depend upon number of variables and requisite period of storing. *So, why not try to use classic RDBMS for small soft real-time process?* We are now trying to realize real-time database by optimizing classical relational database system. Optimization steps like databuferring or suppression of standard relational database's checking and logging algorithmsare unimportant for, and inhibiting real-time data archiving. Also locking algorithms are redundant for one process user in one time.

Information journey from sensor places to databases

Complex logical controlling is most often realized by using programmable logical controller (PLC). Measured process information from each analog senzor is transmitted by standard electric signal to PLC, which (in periodic cycles) processes, transfers and handles this information in numerical format. This numerical information is then sent through industrial I/O buses over to another superior PLC or straight to operator's controller PC, where the information from each senzor is converted to real numbers with correct units and displayed on operator's panel (SCADA/HMI applications). Middle-sized history of heterogenous process data is directly stored in local historical databases (usualy fast file database) for SCADA/HMI application's disposals – **process database**. The oldest data from process database are archived to **industrial databases** over the network using standard database interfaces (e.g. ODBC, OLE DB,...). Process data in raw or aggregated format are used for further data processing such as data analysis, presentation.

Process data for storing to database

We can divide information from process into three basic groups: **dynamic, relatively static data** and **other information**:
Dynamic data, measured variables from senzors, are acquisited by PLC in very short moments – like a couple of miliseconds. If we wanted to store all data for each cycle, we would have to have very large storage mediums. That would be wasting of

resources! We should try to save disk space by choosing **proper period** for cyclic data archiving and store all information with minimal space requirements.
Relatively static data – process parametres (e.g. constants of PID controllers) should be stored only after having been chaged - once in a while. For this change type of data storing, we have to specify the **thresholds** in numerical value as well as in the time scope. There are two methods for data archiving – **cyclic** and **change** option.
Other information from the process (e.g. user notes, alarms and other various events) should be also archived to the database. All information or information package going to the database must be stored with exact date and time (known as timestamp) and value precision. More information will be given in the chapter "*relational database-model building*".

Algoritms associated with databases

There should be many various algorithms for data archiving, hi-level process controlling (e.g. based on artificial intelligence - neural networks for predictive controll), on-line calculations based on technical data sheets in database tables or simple variable conversions (conversion factors in database tables). Main algorithms associated with database serve in heterogenous data storing and archiving to the prepared structures in real-time process database and manufacturing databases. This executive unit used to be located on the side of data server or operator's computer as part of SCADA/HMI application. Other algorithms using stored data can be placed directly in the database as database objects – functions and procedures. All of them are using database interfaces for communication with any **data sources**. Database interface forms the software layer between database system and client application. In fact database interface consists of low-level system libaries – drivers, which enable duplex data exchange. At this time we have a couple of database interfaces such as ODBC, OLE DB, JDBC, BDE, API.

Relational database-model building

We can desing a relational database structure using case-based applications from entity-relational (ER) models, which can also generate executable code, but we'd like to show you how to create a database structure step-by-step manually from the scratch:
In the first step, we have to build up an empty **database instance** (usually we have to set system environment variables and instance configuration parameters in advance), create a **datafile** and **tablespace** for user data and than define **database users** and their **workspaces** (database schemas). Now we can set up privilegies – **roles** for each user or group of users. Concretely, we should sett up permisions for accessing and modifying the whole foreign workspaces or single database objects. We should also imaginary distribute data for storing to several categories, which should be separated by different user access rights.
In the second step, we have to make **database tables**, define **data-types** and **precision** (reserved length of array) for all information, which will be stored in the database. There are predefined datatypes as numbers, strings/texts or dates for most of the relational databases. Next to definitions of datatypes and precision, we should also specify **database constraints** as primary and foreign keys for linked tables, "not-null" and other conditional constraints for table's columns. Relational structure of tables – of stored information – is advantageous for database's translation, but not for simple user data reading. That's why we can create **database views**, which are easier to browse.
In the third step, we can make some database programs like **functions** and **stored procedures**. These programs are formed from blocks of PL/SQL code, which is procedural modification of standard database query language (SQL). There are predefined database programs such as **sequences** and **triggers.** The sequence is a generator of unique series of numbers, used as values for primary keys, while the trigger is a stored procedure, which activates itself when defined action occurs over one database object. Finally, we can prepare SQL scripts for creating whole database structure – from instance to user procedures.

Case study – databases in TIS for small soft real-time system

Short description of original TIS

Laboratory station, climbing film **evaporator unit** (UOP1, Armfield, UK) was equipped with a control system consisting of a programmable logic controller **Satt Control OP-45 SB** (Alfa Laval, Sweden) and supervisory PC running visualization and data acquisition software **TomPack** (ProjectSoft, CZ).

Table 1 Type of I/O for PLC OP-45 SB, AlfaLaval

Type of I/O	Total
Digital input (DI)	16
Digital output (DO)	16
Analog input (AI)	16
Analog output (AO)	6

TomPack for Windows as a SCADA/HMI software consists of TPServer, TPView, TPConfig and UniServer:
TPServer is a central component for comunication between TPViewer and DDE server. It is also managing internal variables and is able to store middle-sized process data history (trends, alarms) to process database based on local, Microsoft Access file-databases.

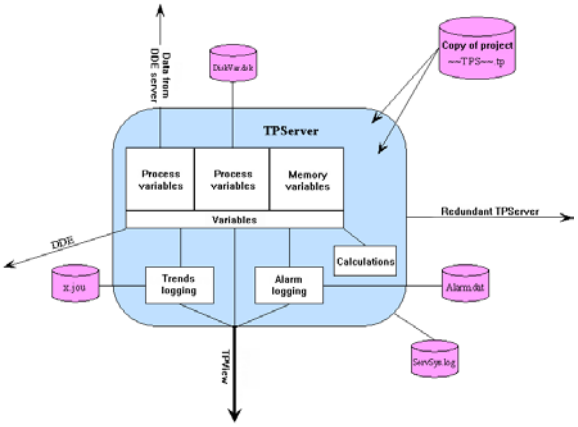

Fig. 2. TomPack system architecture, ProjectSoft

TPView is visualisation software, which displays information received from TPServer and sends back control commands.

TPConfig provides project development enviroment and its configuration. Produced project can be sent either inside the computer or over TPC/IP to TPServer and TPView. TomPack also contains easy script language based on C programming language. Through the use of this script language, we can define mathematic equations; work with TP windows, system and other special functions such as timers, counters and also ODBC function for communication with datasources.

Our research group has set an objective – to enhance existing TIS for on-line data archiving and develop some controlling algorithms associated with database. We had to prepare **technical equipment** – server's hardware, **database system including relational structure** and make a **script** code inside the TomPack for communication with RDBMS.

Technical specifications

There are 18 senzors for measuring the temperature, pressure, liquid level and flow volume in our system. Some of them are utilized in control circuits/loops.

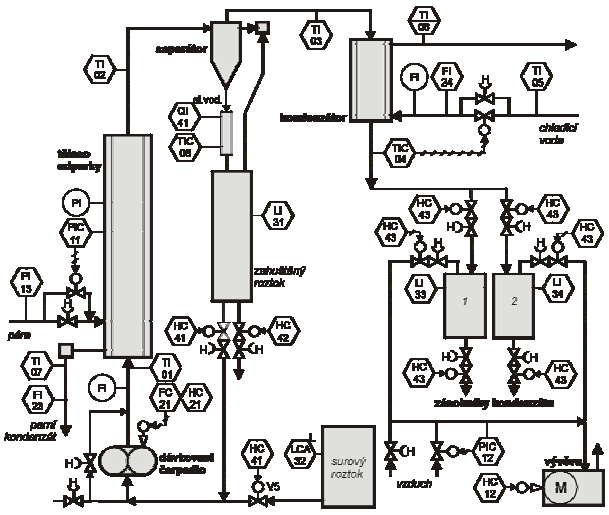

Fig. 3 Measurement and control circuits of laboratory evaporator

Each senzor is linked with programmable logical computer OP-45 SB. This PLC is connected to operator's computer though industrial I/O bus – Sattbus. A special communication adapter Sattbus (ISA card) is mounted in the operator's computer and there are also installed all modules of TomPack software. Created TomPack's scripts, containing ODBC functions, communicate over intranet (protocol TCP/IP) with primary database server Oracle.

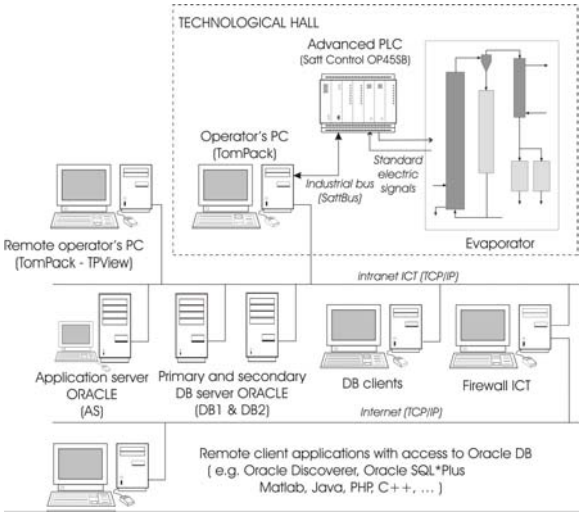

Fig. 4. Communication architecture in TIS

As you can see on figure above, we are using two database servers Oracle for educational purposes. *Primary database server* (DB1) is optimized for communication (on-line data archiving and other data transactions) only with operator's computer. We are now considering scheduled data replication between primary and *secondary database server*. After synchronization, data shall be available for remote database clients without any load on primary server. There is another server - *application server*. This server is able to publish data in many various formats (HTLM, XLS, XML, PDF, text-only…) and is also able to keep server-based applications.

Table 2 Parameters of application and database servers Oracle

Parametr	Value		
	DB Server 1 (Oracle DB1)	**DB Server 2 (Oracle DB2)**	**Application server (Oracle iAS)**
Platform	Intel PC	Intel PC	Intel PC
Procesors (CPU)	1 x CPU AMD K6, 233 MHz	1 x CPU Athlon, 1700 MHz+	**2** x CPU Athlon, 1700 MHz+
System memory (RAM)	128 MB	1024 MB	1024 MB
Harddisk (HDD)	15 GB	2 x 60 GB (RAID 0)	80 GB
Other HW	CD-ROM	U-IDE 100 RAID, CD-ROM	U-IDE 100, CD-ROM
Operating system (OS)	Linux RedHat 6.2	MS Windows 2000 Server	MS Windows 2000 Server
RDBMS / AS (version)	Oracle8iDB EE (v8.1.7)	Oracle9iDB EE (v9.2.0.2)	Oracle9iAS BI (v9.0.2)

Database structure

We divide designed database structure to four imaginary sections. Main section is called "**Control data section**", which contains all executive database objects (PL/SQL procedures, functions, triggers, sequences) and synchronization tables. TomPack's scripts manipulate over ODBC only with this synchronization tables. In 5 second-long cycles (fine cycle period for current system), TomPack's scripts perform buffering of selected values to packets, which are sent to database - synchronization tables. First synchronization table ***Sync0*** provides temporary storage for dynamic process data. This table has *tstamp* column (text field for time-stamp), *experiment_id* column (unique number of experiment) and pairs of *senzor_id_XX* and *value_XX* columns (pair of identification senzor number and numeric measured value for each senzor). After insertion of new data row (to *Sync0* table), representing actual measured values, trigger ***Tr_sync0*** is activated. This trigger is containing PL/SQL procedure, which distribute data to a predefined structure (see below – table *Value*). There are another two synchronization tables ***Sync1, Sync2*** for on-line calculations – see also chapter „Algorithms for on-line computing".

Second section "**Experimental data section**" contains three tables. Main table for this section is ***Experiment*** table. It has *id_experiment, username, user_type, start_time, stop_time* and *desc* columns, where the information about each experiment is stored. Table ***Notes*** has only *experiment_id, tstamp* and *text* columns, where user notes with timestamps can be saved on-line. Last table ***Value*** has *id_value, experiment_id, tstamp, senzor_id* and *value* columns, which are fullfiled by ***Tr_sync0*** trigger for each measured value.

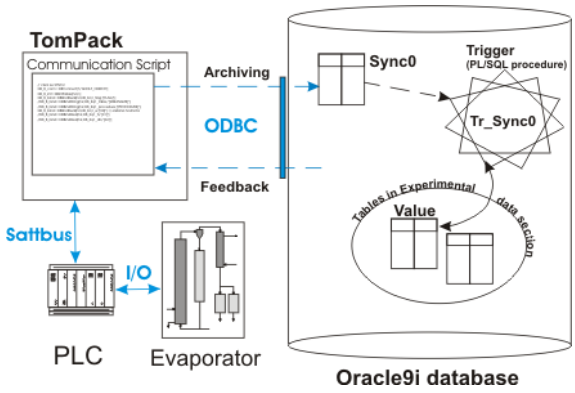

Fig. 6 On-line data archiving– data flow schema

Third section "**System data section**" contains four tables: Senzor, Location, Measurement and Variable. Main table ***Senzor*** has *id_senzor, measurement_id, location_id, senzor_range, senzor_precision, manufacturer* and *serial_number* columns, which predicate about used senzor. Table ***Location*** has *id_location, location* and *senzor_sign* columns, which correspond with senzor locating on evaporator. Table ***Measurement*** has *id_measurement, variable_id* and *measurement* columns, which tell details about measurement type for each senzor. Table ***Variable*** has *id_variable, variable, var_desc* and *unit* columns, where the variable-unit pairs are stored.

Fourth section "**Physical-chemical data section**" contains "TOC", "Table1" and "Table2" tables. Table "***TOC"*** is descriptive table for remaining tables: ***"Table1"*** and ***"Table2"***, where the physical-chemical data sheets are stored.

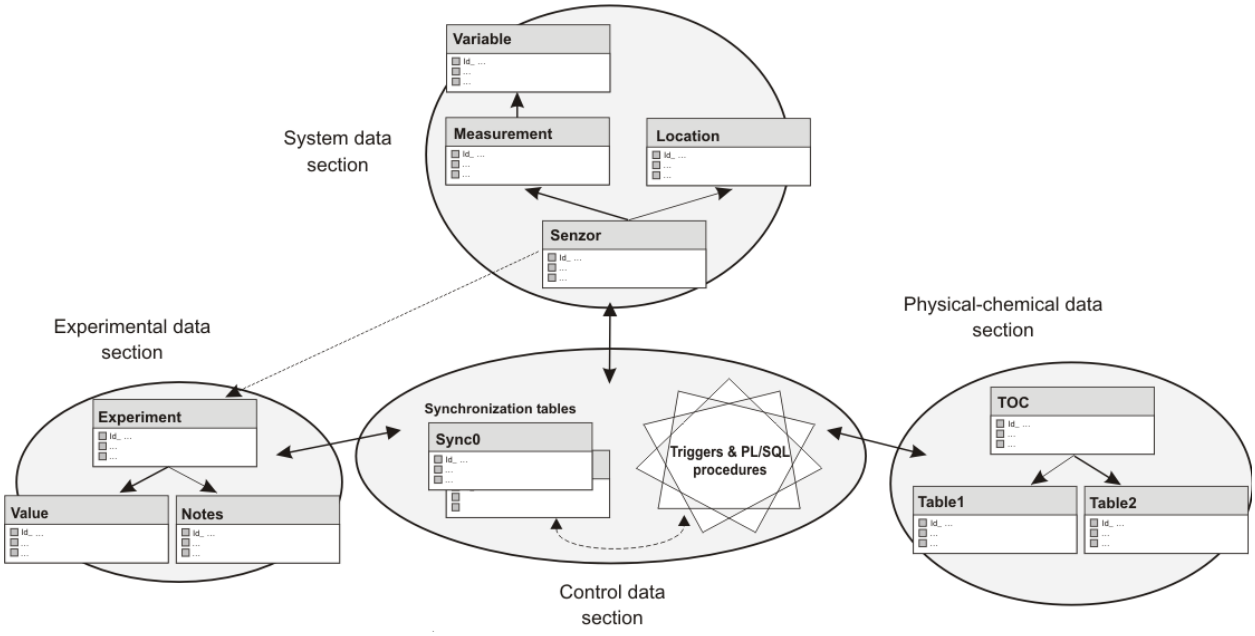

Fig. 5 Relational database structure

Algoritms for on-line computing

We have set a target – to make an application for **on-line estimation** of concentration of the sugar solution (w), based on two measured variables: the boiling point of sugar solution (t_v) and the evaporator pressure (p). Therefore we founded two tabulated relations (f_1, f_2), which were translated into database tables.

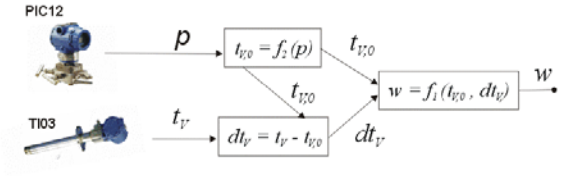

Fig. 7 Algorithm - block diagram

It was necessary to prepare data background and application logic on the **database-side**, but also compose displays and communication scripts for reading/writing information (stored in the database) on the **TomPack-side**.

Two tables were created in the "Physical-chemical data section". „*Table1*" contains tabulated data for f_1 relation [$\mathbf{w = f_1 (t_{V,0}, dt_V)}$]: „Boiling point elevation of pure sucrose solution" and „*Table2*" contains tabulated data for f_2 relation [$\mathbf{t_{V,0} = f_2 (p)}$]: „Boiling point of pure water on ambient preasure". Two synchronization tables (***Sync1, Sync2***) and one trigger (***Tr_calc***) were created in "Control data section". Both synchronization tables have the same structure, but serve for the different puproses – different data transactions. TomPack insert new data row (unique *tstamp* of current calculation, several values for calculation and *flag* indicating state of evaluation) only to table ***Sync1***. This values are processed immediately by trigger ***Tr_calc*** (exactly PL/SQL procedure in trigger) and its result is returned to the table ***Sync2***. *Why there are two sync. tables for this easy algorithm?* Trigger, which is standing above table ***Sync1***, can't write results back to this opened table – it is a database convention of table locking. So, trigger wrote results to the table ***Sync2*** and also update the *flag* of calculation. In this time (of calculation), TomPack is trying to get this results, but it has to wait for regular flag (evaluation is finished). Just in this case – for **on-line estimation** of concentration of the sugar solution, has had to be done two calculations with the interpolation method.

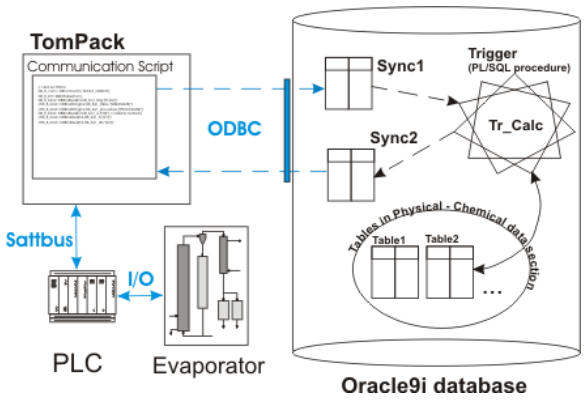

Fig. 8 On-line computing– data flow schema

On-line calculations like this, should be executed once in a while, but also periodic. We verified on-line calculations interleaving with real-time cyclic process data archiving and it seems to be applicable.

Conclusion

We built up a suitable combination of original technological-information system and database server for process data storing. Designed relational structure of database for real-time archiving was successfully tested under semi-plant scale conditions. Our TIS is able to store nearly twenty measured – dynamic process variables in 5 seconds cycles without any problems. It is also able to on-line calculate unknown quantity based on measured and tabulated variables (stored in Oracle database). In the next phase, we would like to expand our system for relatively static variables – system parametres (constants of PID controllers) and create user-friendly environment for data publication & exporting to third party applications.

References

Polnický, J. (2002). *On-line databáze procesních dat pro laboratorní odparku (Diploma thesis).* ICT Prague.

Kmínek, M. (2001). *Měřící a řídící technika (Electronic coursebook;* ICT Prague. *http://uprt.vscht.cz/cz_uc.php).*

Materials of ProjectSoft company, Hradec Králové, CZ (2000-2001). *Manuály k programu TomPack.*