# CONCEPT OF THE SOFTWARE FOR ITEP-TWAC CONTROL SYSTEM

P. Alekseev, F. Sizov, ITEP, Moscow, Russia

*Abstract*

The work is in progress on development of new control system for ITEP accelerators complex. All software for the system should be developed from very beginning. Core element of new software is PostgreSQL object-relational database management system. All interactions between programs on device side and on operator side are made utilizing the database functionality. The database is also provides storage space for all configuration data, operational modes, logs and so on.

## INTRODUCTION

ITEP accelerators complex is located in several buildings in the area of the institute. Control functions should be available for staff located at main control room and at local control panels of subsystems. In addition, diagnostic signals and some control parameters should be accessible from outside the complex. Therefore, the control system will be distributed and multiuser.

Common architecture of the control system is shown on Fig. 1. Offered structure allows connecting to the system almost any equipment with any known control interface. Supposed that the equipment may be industry manufactured or self designed.

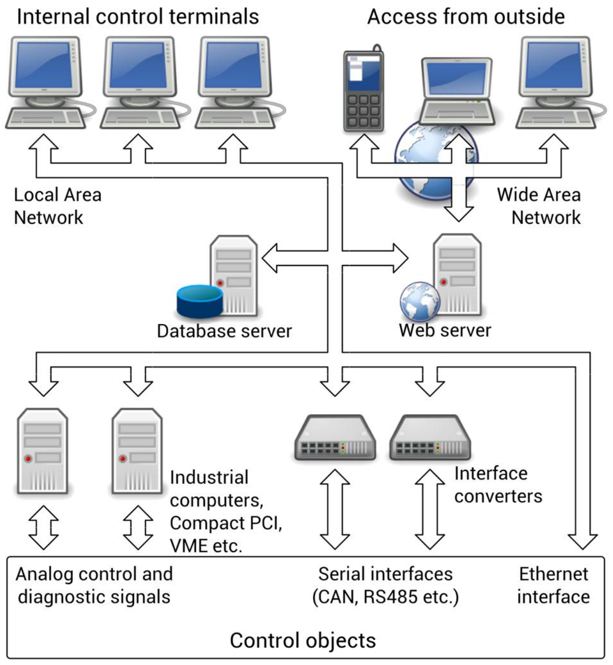
Internal control terminals
Access from outside
Local Area Network
Wide Area Network
Database server
Web server
Industrial computers, Compact PCI, VME etc.
Interface converters
Analog control and diagnostic signals
Serial interfaces (CAN, RS485 etc.)
Ethernet interface
Control objects

Figure 1: Common control system structure.

Front-end programs that user interacts with, may be called from the LAN or even from internet using the web server. Back-end software that interacts with control objects, executes on industrial PCs or on the database server.

The investigation of capabilities of various software products was made relative to our problem. The results coupled with existing resources and expertise allows to offer configuration of software described below.

Control of data flows between front-end and back-end software is implemented by PostgreSQL database. In addition, the database is used to store all the information related to the control system. Main functions of the database are illustrated in Fig. 2.

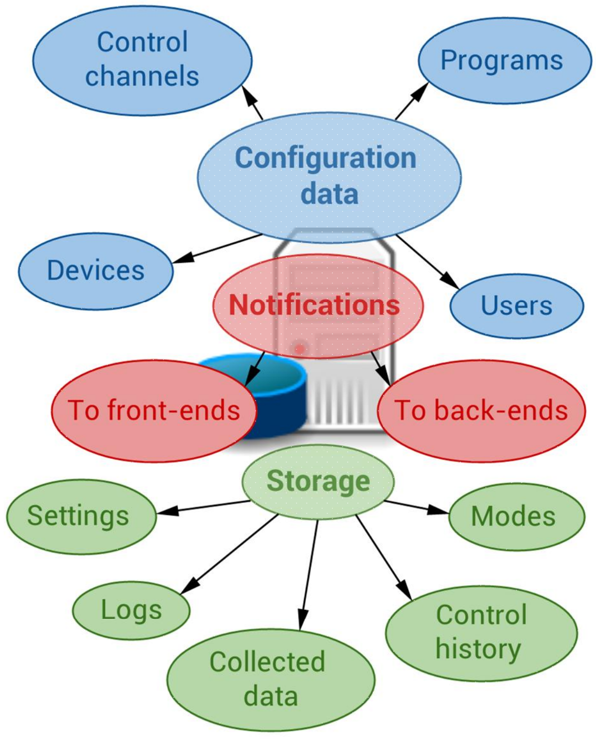

Figure 2: Database functions.

## DATABASE STRUCTURE

There are three types of objects are used to describe the control system structure. These are daemons, devices and channels. Database table DAEMONS contains data about all back-end programs of the control system. Equipment of the control system should be described in the DEVICES data table. Each device such as communication port, dac board, peripheral module etc. has its own database record.

All the physical parameters of the control system are presented as channels. Database table CHS contains an information that used by front-end programs to configure control elements. Each channel should be linked to device that operates with it.

A set of tables that are used in the database to describe all the objects of the control system are shown in Fig. 3.

ISBN 978-3-95450-125-0

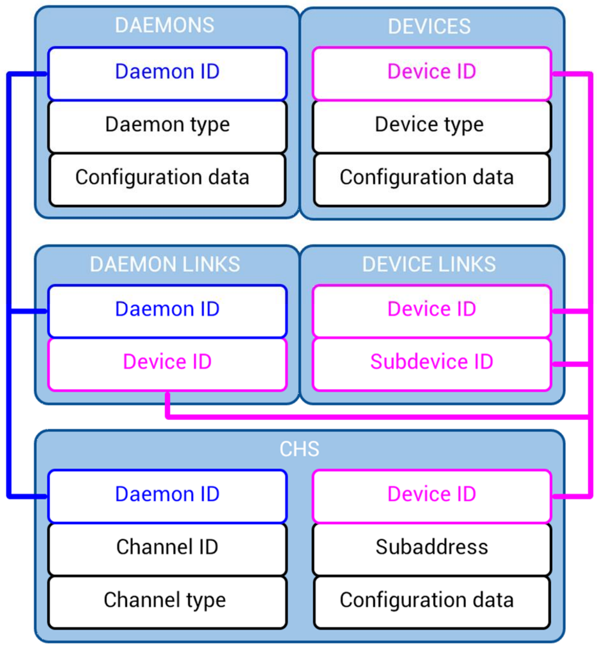

Figure 3: Configuration data.

Tables DAEMON_LINKS and DEVICE_LINKS allows to define what software should service the device and how the devices linked to one another. This way is possible to describe any hierarchy of devices.

The channels could be combined into groups, database tables GROUPS and GROUP_LINKS provides this. Groups of channels may be used to create modes of operations, to handle sets of values etc.

Other database tables are used to store information about the users, front-end programs, schedule, access logs etc.

## DATA MANIPULATIONS

The core element of the control system software is an algorithm for requests processing and transfer of responses. This algorithm could be implemented using native PosgreSQL asynchronous notification commands. PostgreSQL offers asynchronous notification via the LISTEN and NOTIFY commands [1]. A client session registers its interest in a particular notification condition with the LISTEN command. All sessions listening on a particular condition will be notified asynchronously when a NOTIFY command with that name is executed by any session.

We could not find similar functionality in the other database management systems. Sometimes it may be implemented by using server and client programming. In our case, application of PostgreSQL allows to minimize programming time because all the functions are present already.

### *Changing Parameters Values*

Actually we should consider two types of notifications for each of two types of requests. First type of requests is the request to change setting of a channel. Data flow diagram for that request is shown on Fig. 4.

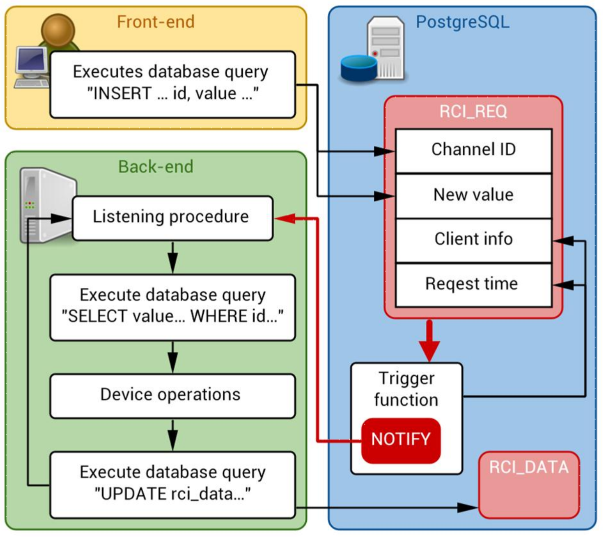
Front-end
Executes database query "INSERT ... id, value ..."
PostgreSQL
RCI_REQ
Channel ID
New value
Client info
Reqest time
Back-end
Listening procedure
Execute database query "SELECT value... WHERE id..."
Device operations
Execute database query "UPDATE rci_data..."
Trigger function
NOTIFY
RCI_DATA

Figure 4: Data flow of control request.

Database table RCI_REQ is responsible for processing of request for parameter changing. Client should insert its request into this table with only channel id and new value. This operation will trigger database function that finds id of the back-end program, sends the notification to it and writes some additional information about the request into the database. Whenever the notification will be delivered to the back-end it selects new tasks from RCI_REQ and process them. If at one time more then one request will be found in RCI_REQ only the recent one will be handled, but all the clients that listen should be notified about the changes. Data flow of daemon's response is illustrated on Fig. 5.

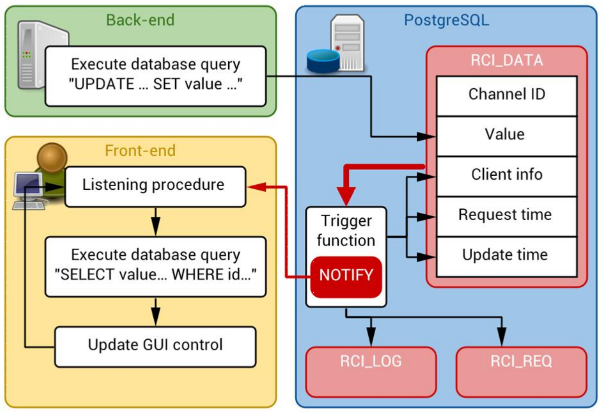

Figure 5: Data flow of daemon response.

Database table RCI_DATA keeps actual values of all the control system parameters. When any of the parameters changed by request or by the other reason the daemon, servicing the channel, should update RCI_DATA. Such update will trigger database function that notifies the clients, clears requests and may even

ISBN 978-3-95450-125-0

store the operation details into the log. The only thing that client shall do, receiving the notification, is to select new value from RCI_DATA.

### *Data Acquisition*

The other type of requests is request to get measured values or collected data. The main difference from control requests is that measurement requests from different clients to the same channel should be serviced simultaneously.

All the data collected from measuring channels are stored in DAQ_DATA table. This table has field named REQ that should be positive in case of any client is waiting for new value of a channel. When REQ is set to zero corresponding back-end program may stop the cycle of measurements. Thus, the only thing that client should do to start measurement is ensure that REQ is more than zero by updating it with positive value. On the other hand, REQ should be decreased by one each time the daemon updates data.

To distinguish front and back-end commands on DAQ_DATA, virtual table, so called "view", named DAQ_REQ is used. DAQ_REQ should be updated only by client programs while DAQ_DATA by daemons.

When client writes positive value into REQ field of table DAQ_REQ triggered function that copies this value into DAQ_DATA and sends notification to back-end program. If at that moment the program is inactive it will wake up and continue data acquisition procedure until any of the channels linked to it has non zero REQ field. This process is illustrated in Fig. 6.

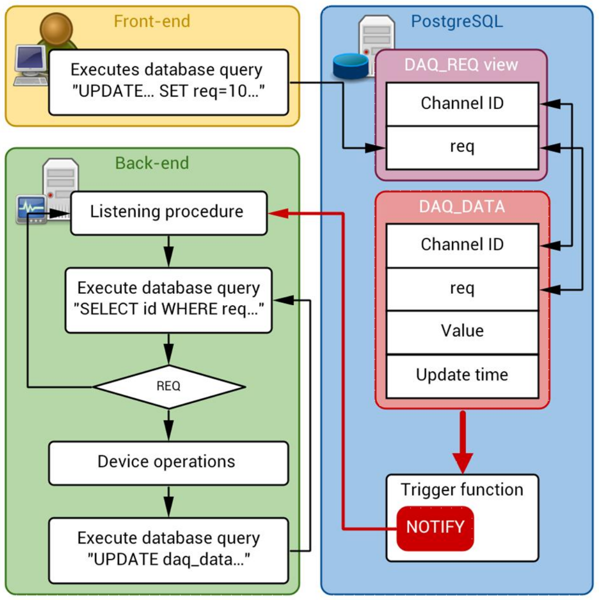

Figure 6: Request for measurement

The second part of the data acquisition algorithm is illustrated in Fig. 7.

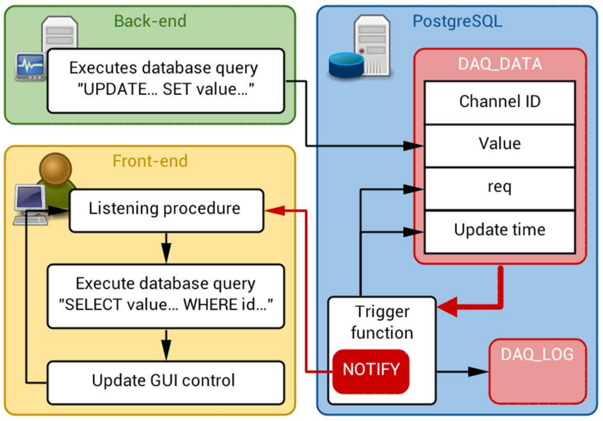

Figure 7: Data ready notification

The same way as in control algorithm measured data is placed into DAQ_DATA table. This operation triggers the function that notifies clients and automatically decreases the value of REQ or copies new value into log table if needed.

## CONCLUSION

All the algorithms and methods described was implemented and tested on subsystems of old control system. They were used to control master oscillators, current sources for beam lines magnets, to broadcast of diagnostics signals of U-10 proton synchrotron. It allows to expect successful application of these methods in the new control system.

It should be noted that the algorithms described may be implemented rather simple inside the database with minimal programming. The clients are utilize standard database control functions to perform queries. Thus, almost any programming language may be used to interact with the system. It gives an opportunity to involve wide range of specialist to development of the control system software.

## REFERENCES

[1] The PostgreSQL Global Development Group, “PostgreSQL 8.3.20 Documentation”, 1996-2012, http://www.postgresql.org