Significance of a Comprehensive Relational Database System for Modern Accelerator Controls

R. Bakker, T. Birke, B. Kuske, B. Martin, R. Müller
BESSY GmbH, Lentzeallee 100, D-14195 Berlin, Germany

Abstract

The advantages of a central 'data warehouse' that holds relevant project data are obvious. Due to their flexibility and given functionalities professional Relational Data Base Management System (RDBMS) (like ORACLE) seem to be better suited for this purpose than Object Oriented Data Base Systems (OODBS) that are optimized for speed and structural sophistication.

In modern accelerator control diverse but multiply connected data areas have to be maintained: Classical DB applications provide the static data necessary to activate real time DBs for hardware control and data acquisition. Recently generic high level software tools have come into use that need complex configuration data. Theoretical models and their connections to operational procedures can be put into a reference framework due to the centralized storage of design data, calibration factors, geometries etc. On the technical side active nodes are increasingly distributed on large (multi-layered) networks. Different computer systems, each optimized for its designed task, provide services different in scope and multiplicity and require appropriate description and configuration data.

RDBM systems support an excellent environment to control data flow and maintenance. Transaction tools allow to import data from genuine sources while conserving ownership and responsibilities. Programmed clients automatically propagate RDBM changes and help to maintain global system consistency. WEB gateways to the RDBM give platform independent and structured access to the data, thus providing high DB transparency. Todays RDBM properties even allow to consider archiving system performance monitor data ('logging') with reasonable update frequency.

1 Introduction

Within the accelerator community DataBases (DBs) and DB applications cover a large variety of functionalities and data types. The International Accelerator DataBase Group (IADBG) has set up a forum for the exchange of ideas in 1994 [1]. A survey of the DB activities within the IADBG reveiled many differences in DB focus and intention, data availability and complexity as well as a common interest to share experiences.

This paper addresses the data flows, tools and DB client applications relevant for the accelerator *off-line control system* [2]. It is restricted to a description of what is done at BESSY. Guiding idea of the DB design at BESSY has been to set up a *single* and *unique reference repository* for all configuration data needed by the control system and its applications.

2 Considerations suggesting a RDBMS

Configuration data necessary for moderate size installations like the control systems for the light sources at BESSY could in principle be mapped into a manageable number of plain files. This approach is fast in the beginning but has serious drawbacks on the long run.

Data integrity, validity, access control is not easy to guarantee in a file based system. Type checking of the data can only be done by dedicated applications. Problems due to incorrectly modified data are hard to trace back. Maintenance cannot be done reliably in a distributed way. Multiple instances like test, backup, old and obsolete versions of the data files exist somewhere in the system simultaneous with the reference data.

Adding a new device to the running accelerator complex is frequently a very cumbersome and lenghty procedure. Devices appear in many different contexts. A device might be subject to a monitoring program handling installation domains and at the same time element of a correction procedure dealing with certain equipment classes or types. Configurations have to be modified at many different places. The variety of configurations is increasing with the number of generic applications available. That problem would be solved by a central and unique data source that allows to generate the different data collections automatically.

At BESSY II a fully distributed three level control system is installed[3]. It consists of console workstations, VME crates and embedded controllers. The relations between configuration data on the different levels are quite complex and are a data area for itself even if configuration selfdetection is implemented wherever feasible.

This type of considerations led in the early phase of BESSY II to the decision to set up a relational data base providing all relevant reference data. The commercial RDBMS ORACLE has been chosen for its powerful environment of additional support packages and numerous third party and public domain products.

3 DB design and implementation at BESSY

In the beginning no clear specification concerning the available and needed data existed. ORACLE has been unknown so far and there has been no idea about the help a CASE tool could give. Instead of adressing a DB design with an information analysis based on formal methods a vague three

step development has been envisaged: (1) In order to get aquainted with the ORACLE tools the design parameter list should be transferred to the DB. (2) During the construction phase the DB should incrementally obtain the references to any type of control system related configuration data. The present status of this step is described in this paper. (3) For the operation phase the storage of data monitoring system performance ('logging') has been foreseen.

3.1 Devices and names

As the only fixed structural element of the DB design any data item relevant for the control system has to be assigned to a specific component. And by convention every reference to a component should be made by a symbolic name. The word 'component' in this sense can refer to a piece of equipment (*device*), a predefined group of such components or a processor connected to the networks (*symbolic* or *virtual device*).

As commonly done, names are composed in such a way that the identification of the specific unit out of a appropriate device class and the structural subdomain of its location can be identified (Fig. 1). The final naming convention has then been fixed for the shortest possible strings that can be parsed in a unique way [4].

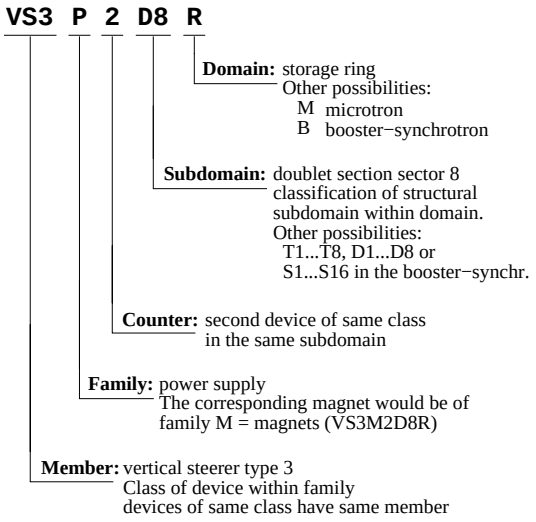

Figure. 1. Elements of a typical Device Name

With respect to the RDBMS the atoms of information contained in the names form a minimal data set that allows to detect analytically any data item related to the name within the data clusters associated ('bootstrap' DB).

3.2 Present DB scope

Along the coarse classification scheme contained within the device names the DB has grown more or less chronologically with the demands of the installation process. Great care has been given to consistency. Primary data are entered once and only once. Derived data are generated automatically by the DB ('constraints', 'procedures'). The data required by the different applications in the specific context are provided by appropriate views.

Today the DB consists of 190 tables, 140 views, 116 triggers, 754 constraints, 232 indices, 28 packages and 7 stored procedures. DB administration applications are 4 forms, 1 report, 150 WEB pages and 100 CGI binaries. Design, implementation, administration, documentation and user instruction is done by a single DB administrator.

3.3 Role of CASE tools

Recently the *PowerDesigner DataArchitect* CASE tool [5] has been applied to the DB. Using the reverse engineering capability of this tool the physical data model has been analysed in terms of entity relationships. Blueprints of the DB structure have been generated (Fig. 2). The results could be inspected and global design and implementation details examined. Some (minor) implementation errors could be identified immediately and a simplified and clearer 'streamlined' structure was found. The required modifications of the DB have been performed by the CASE tool itself.

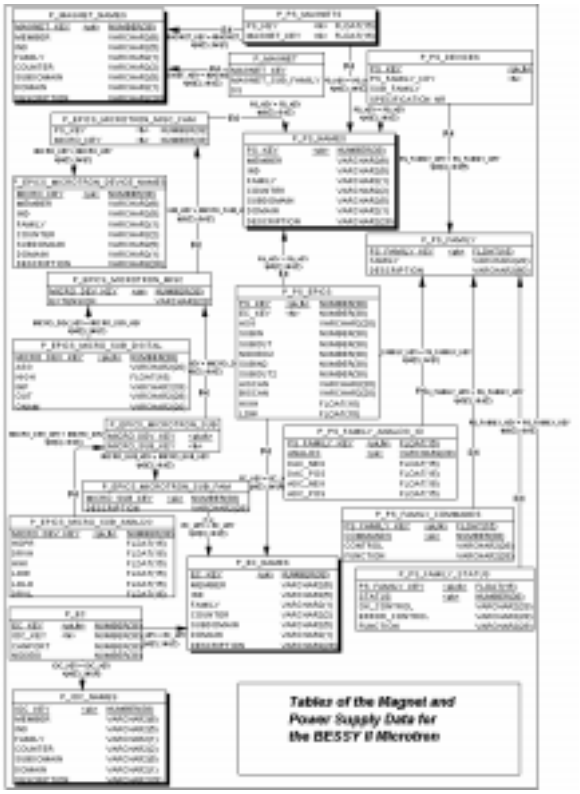

Figure. 2. Blueprint of a Table Space

Even if only used for quality assurance purposes the CASE tool enhances drastically the reliability and integrity of the DB system. Today also the conceptual design of new data areas at BESSY is done with the *PowerDesigner DataArchitect* and the physical data model is generated automatically.

3.4 Data import and maintenance

With ORACLE a large variety of tools is available that allows to enter data into the DB. At BESSY the following sequence of procedures has shown to be very efficient.

On a request to add a new data area the DB administrator sets up appropriate tables for the data, the tags (modification time, DB user etc.), relations and access permissions using the Oracle ServerManager.

The data already existing in a variety of storage formats on different computers are converted to the intermediate standard CSV (Comma Separated Values) table format (Fig. 3). For this purpose either the appropriate storage mode is selected (EXCEL) or custom filters are set up. The geometrical reference data for example have been extracted from the approved AutoCad drawings with lisp scripts and then postprocessed by a Turbo Pascal program.

The ASCII tables are mostly dumped into the DB by the DB administrator using embedded SQL in tiny C programs or the Oracle Loader. In some cases the person responsible for the data content uses *oratcl* scripts [6] to transfer the data from a primary source to the DB.

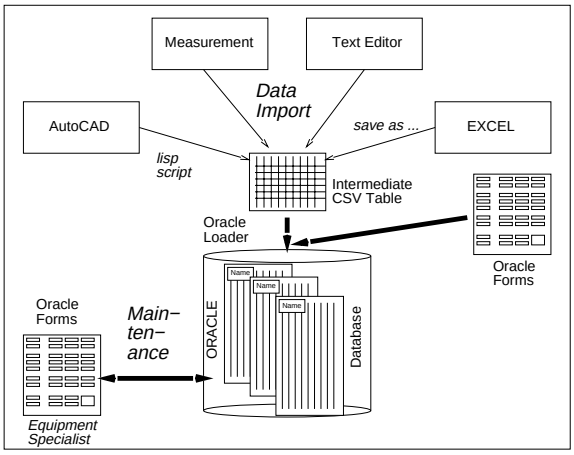

Figure. 3. Initial Data Import and DB Maintenance

Once the tables have been set up maintenance of the data is supposed to be done by the person in charge of the validity of the data. Typically the equipment specialists use Oracle Forms applications prepared by the DB administrator in their MS-Windows environment. Experienced UNIX programmers take advantage of the OCI (Oracle Call Interface) library and a programming language or use an *oratcl* script.

3.5 Data export and queries

For the queries of all day development work the *oratcl* application *oddis*[7] has shown to be easy to use and powerful. *Oddis* is a lightweight tool for administrating ORACLE databases. It combines DB system inspection capabilities with SQL input facilities sufficient for the DB user. It has a subset of the Oracle Server Manager functionalities but is far less complex.

Static configuration files for distinct control system modules are generated by custom *oratcl* scripts or C-programs. These are part of the *make* procedure associated with the appropriate module. The *make* instructions take care of a possible postprocessing and of the staging to the proper installation place.

C/C++ -programs perform online queries through OCI. MS-Windows users utilize the ODBC (Open Data Base Connectivity) interface to load the data of interest directly into EXCEL.

In order to make the content of the DB as widely available as possible the key views have been made WEB accessible and browsable [8] with html forms. The Common Gateway Interface (CGI) has been implemented with C programs. Mostly the output is standardized to html tables using a small set of C macros. There have been experiments with graphical navigation pages too. The biggest advantage of the http type of DB access is that data can be retrieved and checked by everybody concerned wherever there is a computer attached to the network independent of the individual computer skills.

3.6 Backup strategy

The whole DB is exported to another hardware device every night. These snapshots cover the period of one month. The ‘hot’ backup procedure available with ORACLE will be set up as soon as the file server capabilities make it feasible. In addition the DB files are included in the standard procedure of file system backup to a tape robot.

4 DB applications at BESSY

Certainly DB applications are the least portable part of a control system since they map the specific configuration of the accelerator complex. Nevertheless methods applied and structural requirements are comparable.

Today the embedded controllers run only generic IO software. The programs initialize according to the actual hardware configuration detected. The only varying DB entries for this set up of the controllers are basically CAN network, CAN node number and name of attached device. This simple situation will change as soon as table driven autonomous control tasks or complex data preprocessing have to be executed by the embedded controllers. Then the individual programs and the corresponding data have to be prepared for the download to the controller assigned to the related device(s).

For new developments of console level applications the *cdev* API [9] has been made mandatory. The device description files connecting *cdev* attributes with the underlying services are generated from ORACLE with simple *oratcl* scripts.

Presently applications of considerable complexity focus on the configuration of the real time DB and the generic console level applications.

4.1 Real time DB for data aquisition

At BESSY the EPICS control system toolkit is utilized. The real time database (RT-DB) is a fully distributed function block database residing on the VME front end computers called IOC (IO Controller). Functionality of the RT-DB can be graphically constructed with a schematic editor. The ASCII RT-DB description files are generated by postprocessing. The templates describing a certain class of devices are created with the graphical editor by the controls group.

The RT-DB configuration process supports 'templates', i.e. static data varying from device to device with equal functionality are configurable via placeholders that are replaced by the appropriate values at RT-DB start time of the RT-DB. The individual values assigned to the placeholders have to be available in a separate file downloaded during instantiation. These static data assigned to the specific unit of a class are retrieved from ORACLE with *oratcl* scripts.

Maintenance of this configuration part of the RT-DB is fully automatized (Fig. 4). Responsibility for data validity as well as modification permission is given to the equipment specialist. No mediating action of the controls group is required. The person responsible for a certain device performs necessary modifications directly to ORACLE (e.g. with Oracle Forms).

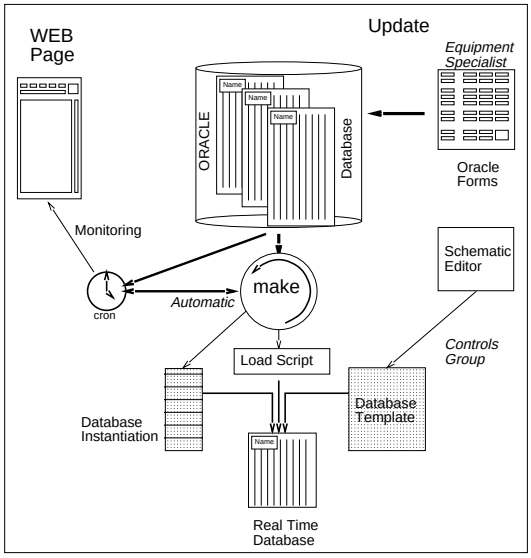

Figure. 4. Automatic Update of the RT-DB

When the changes are committed a DB trigger starts a make run that generates the new configuration files as well as a download script and puts them into place. Whenever operations allow the download of the new values or a reboot of the affected IOC(s) the modification is completed. From data entry into ORACLE to installation the whole process can be monitored on a WEB page (Fig. 5).

4.2 Configuration of generic applications

Today several generic applications are available within the accelerator controls community. They cover certain standard

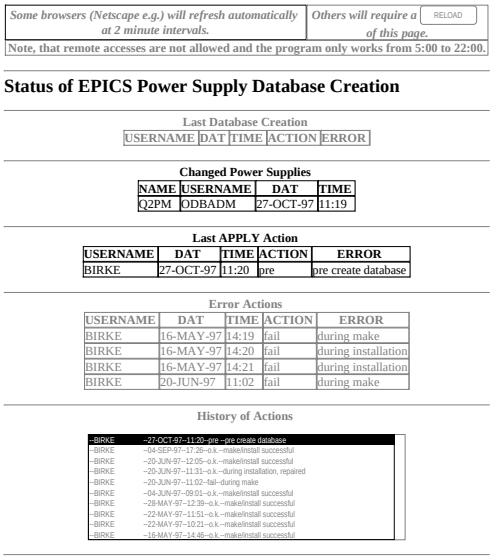

Figure. 5. WEB Page Monitoring DB Changes

requirements if reasonably configured. The following examples illustrate the different data structures involved.

4.2.1 Static

Programs from the EPICS toolkit like *Parameter Page*, *Save/Restore* and *Alarm Handler* need configuration files in different formats. The static file(sets) of these configurations are generated with *oratcl* or C-programs from ORACLE.

Common characteristic features are the assignment of a fixed set of IO channels to a device and the two dimensional structure of collection of devices: Pieces of equipment out of one functional area of the accelerator and one specific device class are grouped into configuration 'atoms'. These elementary configurations are then set up into the flexible groupings and hierarchical trees required for operation demands.

The *Archiver* program in addition has to take care of the different data collection mechanisms (monitor, frequency, etc.). *Synoptic* screens have to arrange control elements and status displays in geometrical correspondence to the real arrangement. Here filter programs derive the screen positions from real locations found in the DB. The synoptic instances displayed by the *Medm* program are generated with *adl_gen*[10].

For the static configurations system consistency has to be taken into account at modification time. Whenever new data are committed to the DB a make run has to be initiated that propagates the changes with an update of all affected files.

4.2.2 Dynamic

Generic applications newly developed at BESSY query the DB directly for configuration data. These programs are less portable but automatically consistent whenever they connect to the DB.

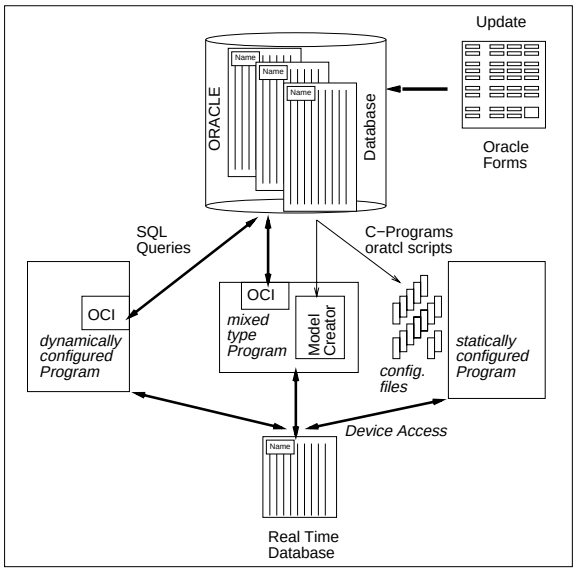

Figure. 6. Configuration Types of Generic Applications

4.3 Mixed type

Arrangement of devices, magnet length, positions, i.e. the static part of the modelling toolkit are extracted from the DB by C programs (using embedded SQL) and stored in C++ class creator files. Due to conceptual reasons of the modelling toolkit these static informations have to be compiled into the code. Programs utilize the modelling toolkit via shared libraries. Modifications of relevant data require an update of these libraries. After a restart the modelling applications are consistent again.

At run time, i.e. working with the actual set points of the accelerator, the programs retrieve the data still necessary for a proper configuration (conversion factors, calibration curves, association of power supplies and magnets etc.) from the DB using OCI.

4.4 Discontinued side–activities

The very first DB project at BESSY was the conversion of the reference list of project design parameters from a word processor file to the appropriate DB structure. It has been meant to be a test ground for shaping of access control and user interfaces. It should add the advantages of the DB data retrieval capabilities. Once set up and corrected maintenance has been abandoned basically due to a lack of interest.

All relevant geometrical information in the DB have been derived from AutoCad drawings. Since most drawings are hierarchical compositions of elementary drawings there has been the intention to reverse the direction of data flow. Approved reference data should be entered into ORACLE and the drawings generated from coordinates found in the DB by lisp scripts. First studies promised a better consistency of the reference drawings. For the time being this plan has been suspended.

5 Plans for the future

The various filters and scripts generating the diverse configurations should be converted into generic forms. The goal is to use the DB also for stuctural information now embedded in the code of the extraction tools or of the dynamically configured applications. Eventually a proper analysis finds out that this meta information can be automatically derived from the DB structure itself.

It is attractive to use the RDBMS for the data stream of the long term archiver. The retrieval tool could then take advantage of the DB extraction speed and the flexibility of SQL queries. Performance tests and tuning of the DB will check the feasibility of this approach very soon.

6 Summary

The early set up of the RDBMS proved to be very helpful. A single repository of reference data and automatic procedures obtaining derived entities made it feasible to generate a system with a high level of consistency. Distributed maintenance turned out to be easy and reliable. Excellent data transparency is provided through retrieval tools that are applicable in nearly any computer environment. The WEB interface in particular does not require specific prerequisites. This turns verification and display of the reference data into an easy procedure that is common wherever questions concerning installation parameters arise.

References

[1] http://www.cern.ch/IADGB/Welcome.html.

[2] According to the definition given in: J. Poole, P. M. Strubin, *A Survey of the Use of Database Management Systems in Accelerator Projects*, Proceedings of the ICALEPCS'95, Chicago 1995, USA

[3] J. Bergl, B. Kuner, R. Lange, I. Müller, R. Müller, G. Pfeiffer, J. Rahn, H. Rüdiger, *Embedded Controllers, Field Bus and a Modular IO Concept: Central elements of BESSY II Controls*, PAC97 , Vancouver 1997, Canada

[4] http://www.bessy.de/control/API/nam_conv.html.

[5] *PowerDesigner DataArchitect*, PowerDesigner 6.0 Family of Products, PowerSoft Design Tools, Sybase Inc.

[6] *oratcl* was written by T. Poindexter <tpoindex@nyx.cs.du.edu> and is available on all major Tcl/Tk ftp servers

[7] *oddis* has been developed at the University of Hanover, Germany and is available under the GNU General Public License

[8] http://www.bessy.de/oracle/.

[9] J. Chen, W. Akers, G. Heyes, D. Wu, C. Watson, *An Object-Oriented Class Library for Developing Device Control Application*, Proceedings of the ICALEPCS'95, Chicago, 1995, USA

[10] *adl_gen* is a C-library for generating adl-files (Ascii-Display-List), written by Jeff Karn (TJNAF)