DATABASE APPLICATIONS FOR CONTROL SYSTEM HARDWARE MANAGEMENT

T. Pal, R. Broomfield, M. Heiniger, T. Korhonen

The control system hardware for the SLS, high intensity proton cyclotron and the Proscan cyclotron facilities is managed together, by a single group. To be able to keep track of the hardware, which will consist of more than 300 VME and CAMAC systems, database applications are being developed. The Controls Inventory Database (CIDB), provides features to list the deployed equipment, track the life cycle of components and the stock status as well as optimize the spare parts stock, thereby improving the overall quality of service. The conceptual, object-relational design and implementation of the core functions and interfaces is described, along with a statement of work in progress for future requirements.

INTRODUCTION

With the large number of hardware deployed in the control systems of PSI accelerators, the logistics of handling all the hardware is cumbersome. Without an overview of the status of hardware stock, it is difficult to know when new modules should be ordered, and where the existing ones are installed. To enable a better control of the hardware status, database applications are being developed. These applications will keep track of what hardware modules are installed, what is in stock, the components in repair, on loan or otherwise not deployed. By recording the history of module movements, their fault history and repair data, one will be able to accumulate a wealth of empirical reliability data, which will aid in estimating needed stock levels, finding out weak modules, and performing preventive maintenance. The database includes the notion of installed subsystems, such that it is possible to find out which hardware modules are used in a subsystem. Additional features will be added to the core structure, which would enable to handle and track requests for additional hardware (new subsystems), include financial information to readily find the costs of systems and to store details such as firmware versions. At a later stage it is envisaged to couple this database with the controls system, such that the control system hardware configuration for the low-level controllers could also be generated.

The SLS controls system uses predominantly VME, and in particular VME64x. The rest of the control system is also moving towards VME. Thus, the design was started using the normal hierarchy of a VME system as a guide implementing the table structures, i.e. racks contain VME crates, VME crates can contain modules and VME modules can contain mezzanine cards. Initially, the bulk of the equipment would be VME or related products, however, provisions would be made to incorporate all types of hardware.

DESIGN AND IMPLEMENTATION

The relational model, in terms of the entities and their relationships has been built using Oracle tools [1]. The implementation is on version 8.1.7 of the server engine. The entities are a set of modular repetitive pair structures, for the different hardware (crates, power supplies, VME cards, mezzanine cards), comprising of a part which describes the type of hardware (description tables) and the other documenting the actual hardware component details (master tables). The primary keys for the hardware components are the site-specific, unique identity tags attached to a particular piece of hardware, and a description identifier for the description tables. The cardinality is one-to-many, and the optionality, depending on the case, is imposed via not null constraints. Additional restrictions on the values are enforced by check constraints. Downward denormalization has been used on some tables for convenience, bearing in mind the requirement to maintain the appropriate attributes after inserts and updates. Fig. 1 shows the server model.

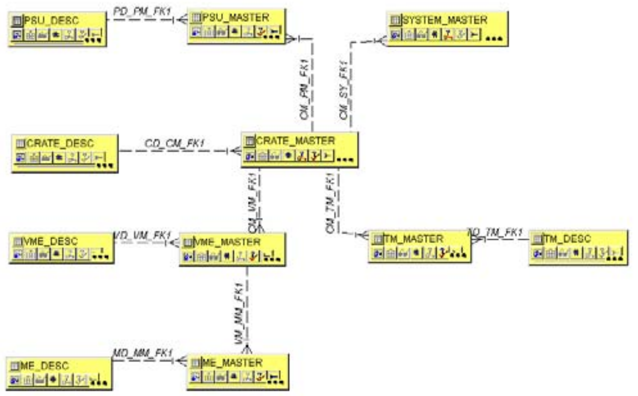

Fig. 1: Core server model tables (collapsed) and relations.

The crate master is the driving table. Each crate and its contents has an association to one or more system names, according to a naming convention [2], which represents the accelerator or beamline devices controlled by the Input Output Controller(s) (IOC) in the crate. For the initial data capture, the sql*loader utility was used with a set of comma separated variable files, mapping to the individual tables, after a process of consolidation from legacy data. The table loading sequence was according to the foreign key constraints. The total number of rows, in all tables, for the initial data was approximately 4000. The time stamp and identity of the user performing inserts and updates is recorded, for each table, as is common practice. In addition, however, a complete set of history tables (not shown in Fig. 1) were created, which mirror the actual tables themselves, and record the Data Manipulation Language (DML) action, as well as the old/new values of the attributes, via triggers.

WEB APPLICATIONS

One of the objectives is to provide a versatile set of interactive web applications to access the data, and display query results. Oracle WebDB (Portal) is used, and provides thin client HTML interfaces, via the Oracle application server (9iAS), respectively. The application component wizards of the tools allow for rapid development, in the case of simple forms and reports. However, due to the limitations of the wizards, as well as the inherent limitations of the web, such as its stateless nature, it is difficult to store and maintain variables. To overcome these, concerning the more complex applications, with interactive feedback, it is necessary to use pl/sql stored procedures and JavaScript. The web applications have been structured in a menu, according to the functions, such as deploying equipment to a system, returning it to stock or to ascertain the availability. Access security and privileges to execute components and modify the data is granted via schemas and roles.

The applications require between five and eight interactive steps via forms and report components to insert or update equipment. In order to optimize performance, collection constructs are used to bulk-bind the output of queries against tables, in the pl/sql stored procedures associated to the application components. At the time of writing approximately 30 applications exist, built using WebDB components. These are in the process of being transformed to portal components and portlets, which offer added value to the end user, such as to reduce the number of navigation steps, to different pages, for a specific task. Fig. 2 shows an example of a page, comprising two portlets.

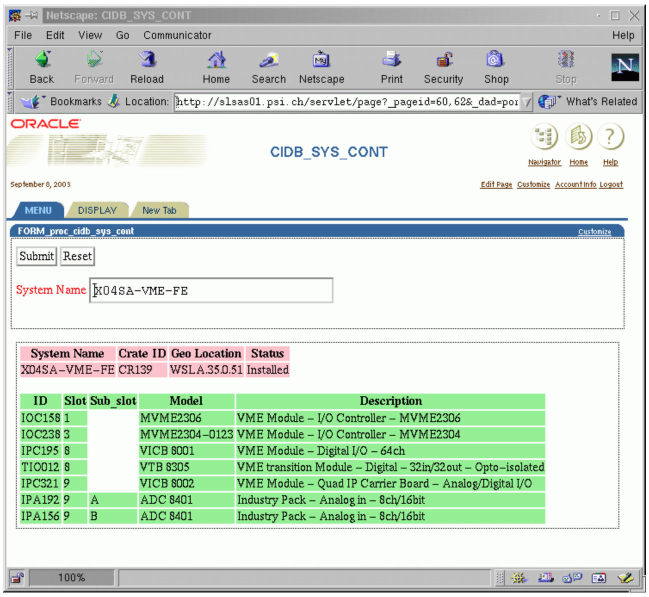

Fig. 2: Web Portal application components.

The upper portlet is a form component, and the lower a report component. On click of the form's submit button, an event handler passes the input parameter(s) to the report component, and after processing, the results are displayed, after a page refresh, on the same HTML page. Page tabs, also visible in Fig. 2, are used to group sets of applications, again with the goal to reduce the time needed to search for components. The development of the web applications has been the majority fraction (approximately two-thirds) of the total time resource. From our experience, it is unlikely that this fraction can be reduced significantly in the future, as it is dominated by the development time for the customized requirements, albeit taking into account the reuse of pl/sql packages, common to the applications.

SUMMARY AND OUTLOOK

A core set of functions for CIDB have been put into operation, and are in use for the control system hardware management. This has enabled us to overcome legacy implementation, and provide a common repository for the accelerator facilities. Oracle object types are being progressively introduced, with the existing tables, which enable to encapsulate the underlying structures of the hardware components, with a corresponding simplification in the sql and pl/sql constructs for the DML statements.

Extensions to the core functions, such as to keep track of the information concerning the reliability and life cycle of the components, although available in an indirect manner from the existing tables, needs to be fully integrated into the set of applications. It is also envisaged to implement functionality in order to store time-dependant calibration data for the appropriate hardware components.

REFERENCES

[1] Oracle Corporation, http://www.oracle.com

[2] A. Streun, SLS Functional Device Naming Convention,
http://slsbd.psi.ch/pub/slsnotes/naming