SLAC-PUB-95-6951
JULY 1995

Lessons from an Enterprise-Wide Technical and Administrative Database Using CASE and GUI Front-ends

Andrea Chan, George Crane, Ian MacGregor, Steven Meyer

Stanford Linear Accelerator Center
Stanford University, Stanford, CA 94309

Introduction–The Vision

An enterprise-wide database built via Oracle*CASE is a hot topic. We describe the PEP-II/BABAR Project-Wide Database [1], and the lessons learned in delivering and developing this system with a small team averaging two and one half people. We also give some details of providing World Wide Web (WWW) access to the information, and using Oracle*CASE and Oracle Forms4. The B Factory at the Stanford Linear Accelerator Center (SLAC) is a project built to study the physics of matter and anti-matter. It consists of two accelerator storage rings (PEP-II) and a detector (BABAR)–a project of approximately $250 million with collaboration by many labs worldwide.

Foremost among these lessons is that the support and vision of management are key to the successful design and implementation of an enterprise-wide database. We faced the challenge of integrating both administrative and technical data into one CASE enterprise design. The goal, defined at the project's inception in late 1992, was to use a central database as a tool for the collaborating labs to:

1. track quality assurance during construction of the accelerator storage rings and detector
2. track down problems faster when they develop
3. facilitate the construction process.

The focus of the project database, therefore, is on technical data (see Fig. 1) which is less well-defined than administrative data. The accelerator and detector components are not fabricated through mass production, but instead consist of tens of thousands of unique tightly quality-controlled parts. This has made our database design very challenging and interesting!

Our Environment

The High Energy Physics community uses highly heterogeneous computer systems. There are more than 700 PEP-II and BABAR collaborators at more than 70 sites worldwide. They use a mixture of Macintosh OS, MS Windows, UNIX,

Presented at the *International Oracle User Week Conference, Philadelphia, Pennsylvania,* September 18-22, 1995

Work supported by Department of Energy contract DE-AC03-76SF00515

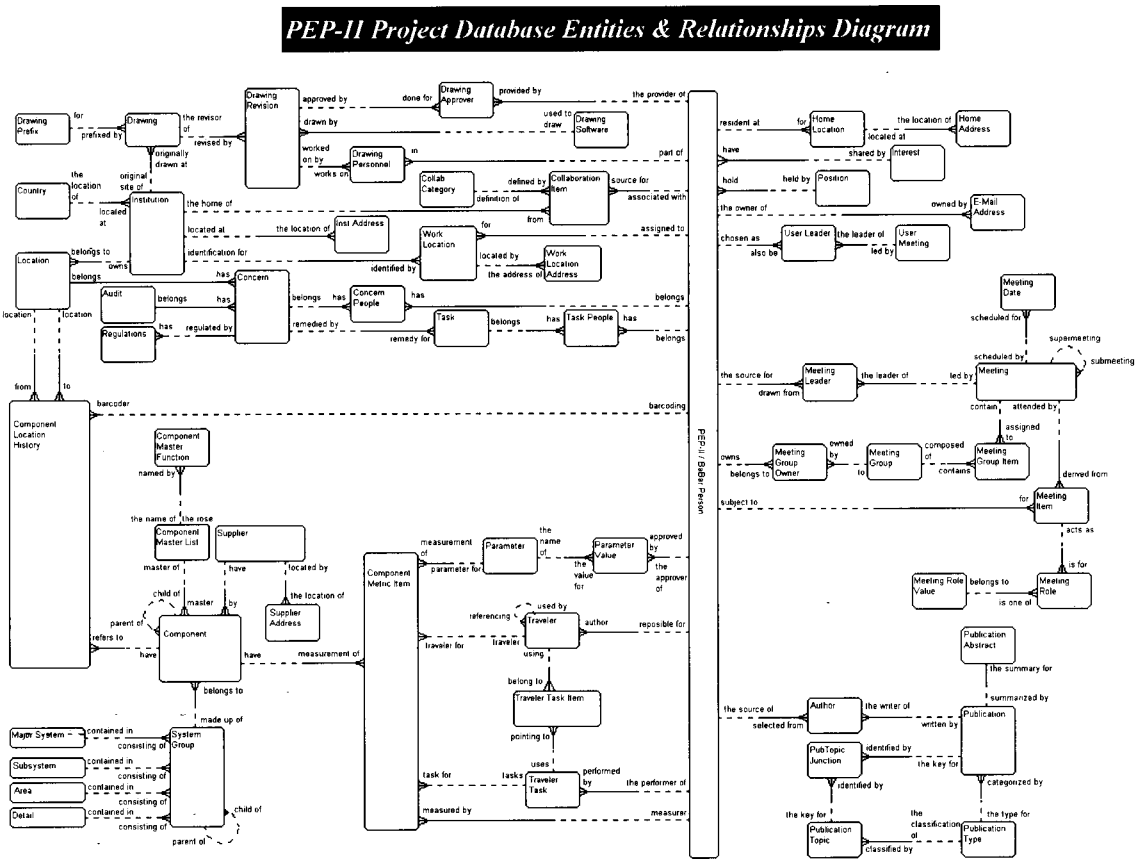

Fig. 1 PEP-II Project Database Entities and Relationships Diagram

VMS and VM operating systems. This open environment led us to rely on the WWW for broad user access to the data.

GUI interfaces are a necessity for our users. Most of them are physicists and engineers who are only casual users of the database. They are quite capable of building their own desktop computer applications. Only GUI tools (such as Forms4 and WWW) can entice them from their existing disparate desktop applications onto a central database. (In particular, we did not dare show them SQL*Forms!)

Scope of the Project-wide Database

The central database we have created will ultimately hold the accelerator design specifications, fabrication data and installation data in one integrated system. It can be extended to contain information required for the operation and maintenance of the accelerator and detector. The following are the main sub-systems, categorized by stages during construction of the machine. All but two of these are delivered and in use by the project.

Design Stage

1. The Personnel module.

2. The Parameter List module, which holds the physics and engineering parameters of the accelerator. Change control is applied to parameter values, and all changes are journaled in the database.

3. A Drawings module capable of producing drawing trees and Bills of Materials. Work is in progress to view CAD drawings on screen through WWW.

4. A Documentation tracking module

Construction Stage

5. A Purchase Requisition Tracking system, based primarily on legacy databases.

6. A Fabrication module based on "travelers" consisting of measurements taken of the component instances. A traveler is a set of fabrication instructions and measurements to manufacture that component instance.

Installation Stage

7. A Survey and Alignment module, which contains ideal and actual coordinates for installed components.

8. The Inventory/Property Control system, which is based on barcodes.

9. The Environment, Safety and Health (ES&H) Corrective Action module, which tracks safety problems.

10. The Personnel ES&H Training Records module -- produced by linking our database to the ES&H databases at different lab sites.

Implementing The Key Pieces

To deal with the contradiction between the large scope of an enterprise-wide database and the pressing needs of a construction project already under way, we focused on getting key pieces of the skeleton database running right away. Other pieces were created as management and production needs arose.

The first three modules implemented were (see Fig. 2):

1. Personnel (most tables in the database have relationships with this module); this includes a platform-independent e-mail distribution system.

2. Drawings and specifications (mainly, but not exclusively, CAD)

3. Components

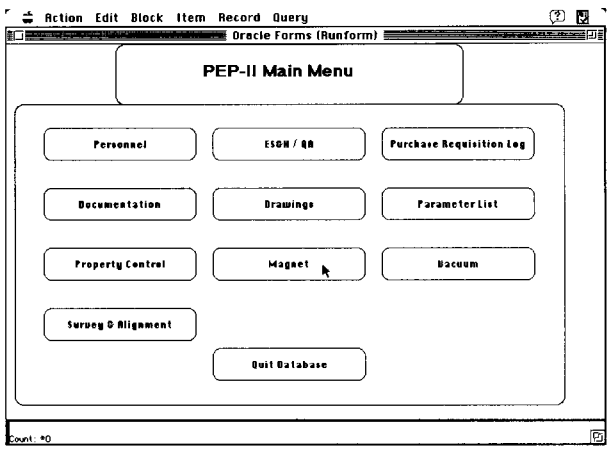

Fig. 2. Main Menu from Oracle Forms4.

The Components system was the heart of the technical part of the database, and we revised its design many times. In Fig. 1, the Entities and Relationships Diagram shows that, for a component to exist in the database, it must be entered in the Component Master List entity. The component can have many parameters and their corresponding design values. It can also have a drawing number and a revision number. Each instance of the component is entered into the Component entity. The component instance is produced according to the order of procedures from a traveler which has a traveler number and revision number. In the traveler, the fabrication instructions and measurements are identified by task numbers. Measurement values are stored in the Component Metric Item entity, see Fig. 3. The Component Location History entity records the history of physical locations of the component instance, with the final designation being its destination in the PEP-II tunnel.

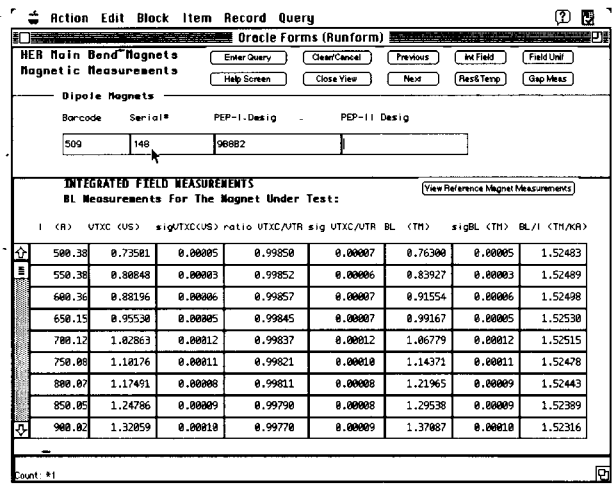

Fig. 3. Bdl measurements for HER Dipole Magnet serial number 148 from Oracle Forms4

We have tested this Components design (see Fig. 1) with PEP-II magnet and vacuum systems, and with the BABAR calorimeter crystals system. Through these tables, we are able to retrieve fabrication and measurement data by many criteria. Users can query against views that we have built which join these normalized tables. Data can also be dumped from the database by the Clear Access(TM) into many other software packages familiar to and preferred by the users, such as Microsoft Excel (see Fig. 4).

One of our most successful applications is the Purchase Requisition tracking module which integrated together Oracle tables and six legacy databases. Before this, users had to search for the data using these disparate databases. Through WWW, as well as Forms4, we provided the interface into these hybrid systems as part of our project 'enterprise database'. In the beginning, we were reluctant to venture into this legacy swamp. However, this turned out to be one of the most utilized modules and encouraged us to look at the enterprise-wide database more from the user viewpoint. That is, to the users, the interface *is* the database.

Often our users capture data in a myriad of ways not involving Oracle--in spreadsheets, flat files, non-relational databases--and they will continue to do so. We work with the users to record the data in these formats until our analysis and development are completed.

Creating the Seamless Whole with World Wide Web

As many institutions have found, access to and use of WWW has been an enormously successful tool. Providing WWW access to the many components of the PEP-II/BABAR Project Database has had the effect of "turning on the light" for literally hundreds of our users. Through WWW, we can provide easy searching, retrieval and reporting directly from the Oracle database. Complex search criteria, table joins and linking to legacy databases are all transparent to the users. Response time for Oracle access via WWW has been excellent.

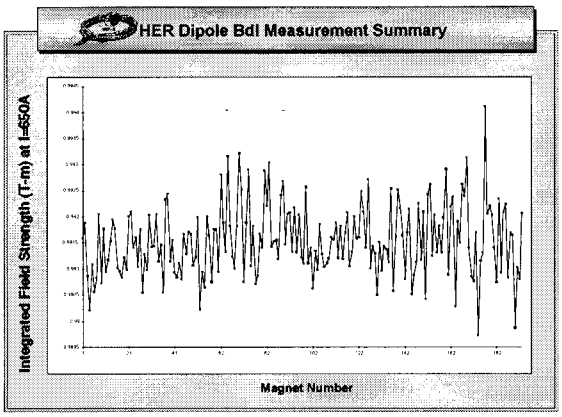

Fig. 4. Graph of Bdl measurements at 650 A for all HER Dipole Magnets from data retrieved by Clear Access and charted in Microsoft Excel.

Our goal is to provide access to all public components of the Project Database via WWW. In general, direct access to the database via Forms should only be necessary for:

1. users needing to make modifications to the database

2. tables where security issues cannot be fully resolved on WWW.

We prefer the WWW interface because we find Oracle Forms4 and Reports2 cumbersome in requiring that users run two distinct programs, even when Reports2 is called by Forms4. The two programs require a lot of computer disk space and memory, and it is a chore updating the client computers when new versions of the programs are released. Although Forms4 is nearly source code portable across platforms, we had to do significant work to manipulate buttons and fonts.

Implementation

At SLAC we are currently running the CERN WWW server code on UNIX and VM computers. The VM mainframe contains our legacy databases. The UNIX WWW server is on a SUN with SQL*Net access to the Oracle database instance, which resides on an RS/6000.

The heart of the database/web interface is in one Compatible Gateway Interface (CGI) script. A major drawback to using CGI scripts is that, at this level, they do not have any security and could potentially execute undesirable commands or have unexpected results.

We are fortunate that much of our data is not of a sensitive nature. With hundreds of collaborators around the world, being able to provide information with few security constraints is a big plus. However, some data are of a semi-sensitive nature so we have been faced with the security issue. Therefore, access to some information from off-site machines is controlled.

In order to provide some minimal level of security for CGI scripts on the SLAC WWW server, a CGI security Wrapper is used. The server invokes the PEP-II CGI script through the Wrapper, which is itself a CGI script. The Wrapper provides some simple checking on input to the PEP-II CGI script. This also makes it trivial to permit the execution of "authorized" UNIX commands.

The PEP-II CGI script may be called in several ways:

1. Directly from a WWW form requesting a database search with specified criteria.

2. From an http URL requesting that a customized WWW form be created based on a requesting user's IP address. In this case, the form produced may be dependent on the requesting IP address (on-site versus off-site).

3. From a hypertext reference (hot spot) requesting information from the Project Database or perhaps from a legacy database system.

The following example illustrates the process of searching for a drawing using WWW. The user first navigates to a WWW form that has several choices for search criteria, such as drawing number, title words, date ranges. The user fills in one or more search criterion fields (using a wild card character if desired) and presses the "search" button. Control then is passed to the PEP-II CGI script (via the Wrapper). Arguments passed to the CGI script include all of the filled-in search criteria plus a function parameter that tells the script which operation to perform.

The arguments are examined by the script resulting in a single SQL WHERE clause. The predicate is then passed to a Pro*C program that formats and outputs data based on a declared cursor. In addition to outputting the requested data, the C program may insert http references that in turn call upon the same CGI script with other functions. For instance, engineers and approvers of the drawings are linked to our personnel tables so hot spots are included for quick reference. In addition, hot spots to produce higher and lower assembly drawing trees may be automatically created. This is one example of the many relationships that are included in the WWW pages, not only from within the Project Database in Oracle but also from hot spots referencing legacy databases. (Fig. 5)

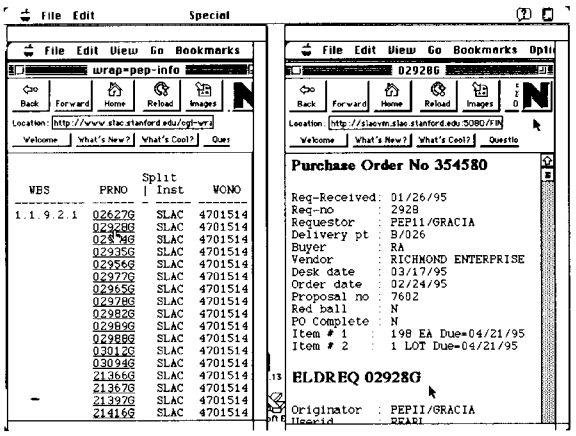

Fig. 5. World Wide Web interface to PEP-II Purchase Requisitions joining Oracle and legacy SPIRES data

The URL for the PEP-II/BABAR Project is:

http://www.slac.stanford.edu/accel/pepii/home.html

Using CASE for Design and Maintenance

The project is a large one with many database objects as well as many screens and reports. We needed something to help us manage these and, just as importantly, we wanted to gain experience with Oracle*CASE. CASE has helped greatly, but at times has hindered us from upgrading client software. CASE generators for Forms4 were not released at the same time as the CDE suite. We had no choice but to abandon CASE for a time.

The first lesson one learns about CASE is that training in the use of the product is a prerequisite. This training is crucial to understanding how facets of CASE integrate and complement each other. The course materials one obtains from completing CASE classes include recipes detailing how to accomplish the required operations to develop a project. Unfortunately, the courses do not include how to recover from missteps, and on the first project these will be made. After a short while, the workings of CASE become familiar and its benefits apparent.

We have a very small team doing the analysis. The physicists and engineers were extremely busy with design problems of their own to meet the requirements of the project. Time and again, the users told us that they did not have a clear idea of what they wanted. In this hectic environment, it was often difficult to perform a detailed analysis of the system. We combined lessons we learned during our visit to CERN in Geneva, Switzerland, our knowledge of databases used to track the manufacture of other products, and what our users liked and disliked. The result was a decision to show the users prototype systems and solicit comments.

This prototyping worked extremely well. Users were eager to state how the system should be improved to better meet their needs. CASE allowed us to build the original screens and reports rapidly, and to easily implement changes. Using CASE, we could often produce in a couple of hours what might have taken a week or more using only Forms and Reports. CASE did not solve all our forms problems. We had to code some procedures on our own, as well as some triggers. However, approximately 90% of the code was written by CASE. The code produced also served as a PL/SQL training aid. By examining the code that CASE produced, developers were able to write better triggers in other applications.

In the prototyping methodology where objects and the relationships between them will change, CASE provided some stability and documentation. This project was first built against a version 6 database when roles were unavailable; however the fact that all the grants were stored in CASE made it simple to restore permissions on dropped and recreated objects.

Why use Forms4?

During development of our database, it became clear that Forms4 would be the preferred interface for data input for several reasons:

1) While several other interfaces are available for data input (for example, MS Access), data type and range checking, inter-table integrity, key-column protections and locking schemes are not built into these applications.

2) Login security. Some possibilities for data input, for instance WWW, do not enforce user and password protection to limit who may input data.

3) Forms generation and re-generation using Case. While the database is still being developed, the forms used to input data may change many times. Using Case to generate input forms speeds the process and insures that the new forms

are correctly modified to match database changes.

4) GUI interface. Most of the users of the database were from a Macintosh or MS Windows background. They would not be satisfied with a character-mode application. Forms4 does much to accommodate these users.

Making Spreadsheet Forms

Our users really prefer the cell-grid layout of commercial spreadsheet programs such as MS Excel. By presenting the users with forms that visually resemble a spreadsheet, we were able to increase the user acceptance. In this situation, Case 5 is quite helpful by creating Spreadtable blocks. This is done by forcing each row of the block to take up only one line on the screen. Some modifications to the standard forms template file (usually this is x:\orawin\cgen40\admin\ofg4guit.fmb), can allow a form's Spreadtable block to stretch when a user resizes the window.

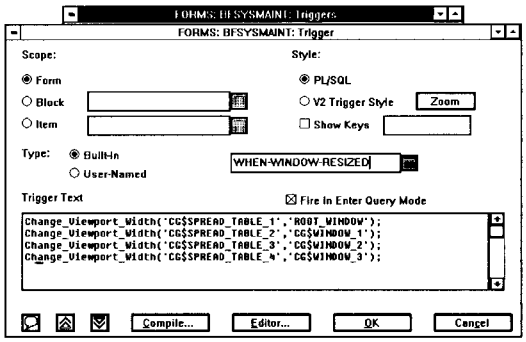

Fig. 6 Defining The When-Window-Resized Trigger

A trigger is created (see Fig. 6).

One call to the Change-Viewport-Width procedure is required for each window in the final form. The second and subsequent calls in the template form may be commented out to allow the generated forms to compile successfully under the generator. Before using the form, the Forms Designer should be used to edit the trigger to include only and all required calls.

```
PROCEDURE Change_Viewport_Width(
   P_VIEWPORT IN VARCHAR2,
   P_WINDOW IN VARCHAR2) IS
/*Change viewport width so that*/
/*spreadtables will fill the */
/*root window*/
root_window_width integer := get_window_property(P_WINDOW, WIDTH);
view_x_pos integer := get_view_property(P_VIEWPORT, DISPLAY_X_POS);
view_width integer := root_window_width - view_x_pos - 2;
Begin
set_view_property(P_VIEWPORT, WIDTH, view_width);
end;
```

Fig. 7 The Change-Viewport-Width procedure

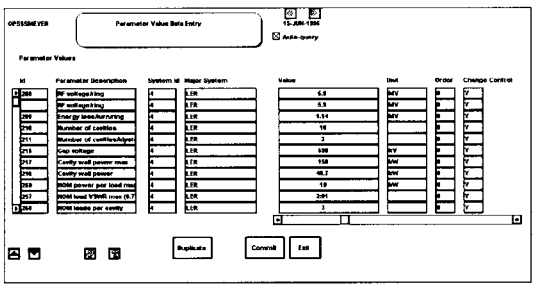

Fig. 8 An Example of a Spreadtable Form

Conclusions

-- The time is ripe for the enterprise-wide database, and the integration of administrative and technical data is an innovative use of this approach within the High Energy Physics community. [2]

-- Our users are excited by the capabilities of such a database,

especially when they use WWW to access the data and see it interlinked (no matter in what kind of database back-ends or servers the data resides).

--To make this a reality, delivery time on modules had to be fast. This is what we have done in the last three years with a small team averaging two and one half full time employees.

--Oracle*CASE enabled us to build prototypes quickly. We used it to generate most, although not all, of the codes.

References

[1] A. Chan, S. Calish, G. Crane, I. MacGregor, S. Meyer, A. Weinstein, J. Wong, 'The PEP-II Project-Wide Database', Proceedings of the 16th IEEE Particle Accelerator Conference (PAC95)

[2] J. Poole, 'Databases for Accelerator Control - An Operations Viewpoint', Proceedings of the 16th IEEE Particle Accelerator Conference (PAC95)

Acknowledgments

We are grateful to the PEP-II/BABAR management for their support since the beginning of this endeavor. We are also grateful to our colleagues at SLAC, PEP-II and BABAR, in particular Geoff Girvin, Sheryl Calish and Arnold Weinstein, for their contributions. We thank our colleagues at CERN from whom we learned much.

Work supported by the Department of Energy contracts DE-AC03-76SF00515.