EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH
CERN - SL DIVISION

CERN-SL-95-111 OP

The Evolution of the LEP Logging Database

R. Billen

ABSTRACT

In January 1992, a project was started to create a system, using an on-line ORACLE database, to allow logging of a multitude of data on the Large Electron Positron Collider (LEP). The aim of this project was to log particle beam characteristics, physics parameters, hardware settings and environmental conditions. Storing and keeping track of this heterogeneous data for a period of at least one year would permit a better understanding of the behavior of the fairly new LEP Collider.

After using the logging system for almost four years, nearly three years of which in full operation, the reliability and performance has been proved, endorsing the design of the database and surrounding software. Moreover, the large number of users of the logging database and the huge amount of new requests for data logging shows the high activity and usefulness of this system. Furthermore, in the context of the 1993 and 1995 energy scans, the logged data turns out to be indispensable for thorough monitoring of the LEP beam energy, which is affected by many parameters.

Since the commissioning in 1992, the logging database has been subject to an ORACLE migration from version 6 to 7 and a hardware upgrade of the host platform, in order to keep in step with latest technology and future user requirements.

This paper describes the evolution and present state of the LEP logging database.

1 INTRODUCTION

The Large Electron Positron Collider (LEP) is presently used to investigate the mass of the Z^0 boson, determined by the combined data from the four LEP experiments. For a better understanding of the machine performance, a large number of heterogeneous parameters need to be logged and correlated. Moreover, in 1993 and 1995 energy scans were performed around the Z^0 resonance, with regular beam energy calibrations using resonant depolarization [1]. A continuous logging of parameters that could affect the energy of the beams, allows more precise determination of the Z^0 mass and width.

In January 1992, a project was started to create a logging system with an ORACLE Relational Database Management System (RDBMS) as kernel. One year's worth of data was to be kept on-line, which was estimated to be 8 Gbyte. The logging system as such has been described elsewhere [2]. In order to handle and manage a very large database, special techniques needed to be adopted, which have shown their effectiveness.

2 THE DATA

The nature of the data is very heterogeneous. One can distinguish beam parameters (e.g. lepton bunch intensity, longitudinal and transverse profile, closed orbit position, synchrotron tune,...) as well as equipment readings (e.g. power converter current, accelerating RF-unit voltage, relative vertical position,...) and environmental parameters (magnet temperature, air pressure, humidity,...).

Most of the data have the number data type, requiring a precision of 3 to 8 digits.

The data taking rate varies between seconds and hours, coming from repetitive measurements as well as from asynchronous events. Moreover, these data rates are a function of the operational mode of LEP. Maximum data rates occur during LEP modes "*physics*" (i.e. stable beam conditions at 45 GeV for experimental data taking) and "*calibration*" (i.e. single beam energy calibration with favorable spin polarization conditions).

3 DATABASE DESIGN

Design method

For data modeling, the Natural Information Analysis Method (NIAM) was used [3]. In a NIAM schema one defines objects, facts between objects, constraints on facts and between objects. This data modeling method was preferred above the Entity Relation Diagram method (ERD) because of the binary relationship approach, completeness and clarity of the method.

The RIDL* tool [4], based on the NIAM method, allows rapid creation of Oracle6 tables. Unfortunately, since the company ceased to trade, the RIDL* tool has not been supported for Oracle7 compatibility, which implies editing of the files generated in order to make them compliant with the new syntax concerning constraints and storage parameters.

Design principles

At the design level a strategy with the following principles has been carefully followed and applied to all data systems:

- Each logging object is identified by two attributes, namely a *timeslot* (primary key, number type) and a *timestamp* (unique key, date type). The introduction of the timeslot allows splitting into several tables where a huge number of data items is concerned. This minimizes data volumes and allows rapid searching for nearest timestamps to a given time.
- The number of columns in a table must not exceed 30 in order to use the block space efficiently. Hence, *row packing* can be applied to store synchronous data items over several rows making use of an extra identifier as part of the primary key. Therefore, the column names will be general (e.g. "value_1") and a correspondence table makes the link between data item identifier and it's location in the table.
- Column definitions must be as small as the data precision (e.g. number(6,4)) or size limitation must be enforced by software (e.g. round()). The function vsize() allows to verify the internal representation in bytes.

4 DATABASE IMPLEMENTATION

All data logged is to be kept on-line for at least one year, in some cases it is even kept several years to allow comparison with data of previous years. To avoid storage allocation problems, which unavoidably result in performance degradation, all data for a given table should be held in one extent. The size of the table can be calculated with the maximum frequency of the measurements and desired minimum period length before overwriting the data.

Database objects

Since the original implementation was done in Oracle6, declarative data integrity is not enforced according to the Oracle7 syntax for all objects. However, all new objects use named entity and referential constraints
(e.g. primary key, unique key, default, not null). The use of triggers and stored procedures has not been implemented yet due to the amount of surrounding operational software which already exists.

A separate data tablespace and index tablespace has been assigned for each individual data system (e.g. beam intensities, power converters,...). The data and index tablespace files reside on physically separate disks to

avoid an I/O bottleneck. By means of select grants to a user account without resources, any client can query all objects.

For each data system a single-row *time manager* table holds the last timeslot where data has been written as well as the maximum timeslot in order to permit an automatic roll-around. The clients writing into the database tables are Pro*C processes that only *update* timestamp and data columns at the appropriate timeslot, avoiding to update indexed columns as much as possible.

Storage parameters

All storage parameters are of major importance and need to be defined correctly at table creation time. Immediately after table creation, the tables are filled with a sequence of timeslots and dummy (but real size) data. The following storage parameters are defined for tables and indexes at creation time and must be changed with respect to their default value:

- *initial extent* : can be calculated based on the number of rows, number of columns, column sizes and taking into account block header size and *pctfree;*
- *next extent* : If the initial extent is correct, no next extent is ever used. However, a constant value for all tables should be chosen (e.g. 5 Mbytes);
- *pctfree* = 1 : Since tables are filled with correct row sizes, no space needs to be reserved for future updates;
- *pctused* = 99 : The only moment where row insertion takes place is when tables are filled at creation time, row deletion never takes place;
- *pctincrease* = 0 : In order to keep the next extent of constant size.

5 DATABASE ENVIRONMENT

Client Software

The logging system is a complex of UNIX processes running on a dedicated HP-UX workstation. The processes that write to the database are compiled Pro*C programs called "*logging black boxes*". These logging black boxes read the data from a measurement source and write the data to the appropriate table at the correct timeslot. The measurement sources can be very different: an ORACLE (measurement) database, ASCII files or C-structures.

As this software needs to run continuously, one process "baby sits" all logging black boxes and another process polls the database and sends out alarms in case of logging malfunctioning.

Hardware

In June 1992 a dedicated SUN workstation 630 MP with double CPU (20 MIPS, 64 Mbyte memory) was purchased with 14.3 Gbyte of disk, running SunOS 4.1.2. The RDBMS ORACLE 6.0.33.2.2 was originally installed. Nowadays, a SUN SPARCserver 1000 with two CPU modules (135.5 MIPS, 128 Mbyte memory) hosts two 1 Gbyte system disks and one SPARCstorage Array with thirty 1 Gbyte disks. The operating system is Solaris 2.4. Two RDBMS ORACLE 7.1.6.2.0 instances have been installed to accommodate the LHC String Test Facility as well [5].

The FDDI interface ensures reliable and fast communication over the network, supporting mainly SQL*Net V2 connections. An Exabyte 10i jukebox driven by Legato Networker software permits consistent backing up.

Backup strategy

Taking into account the importance of the logged data, the archive log mode is set to allow a complete database recovery in the eventuality of a database crash. Under the present conditions, 120 Mbytes of archive files are created per day. These files are regularly copied to off-line media. A full off-line database backup is performed approximately every week when LEP operating conditions permit. This disk-to-disk backup takes 30 minutes, an on-line backup has not been implemented yet.

The need for some archiving starts appearing, but no definite solution has been outlined yet. For now, some tables with old data that might have a future interest are imported into a dedicated tablespace.

6 DATA RETRIEVAL

A wide variety of people are interested in the logging data on a regular basis. The main users are accelerator physicists, control room operators, experimental physicists and equipment group people. For all these end-users, the access to the logged data has to be easy, self-explanatory and fast. At first, a menu driven program running on all main platforms was developed and installed. Nowadays this program bas been completely replaced by a graphical interface called GUILS [6].

Graphical Interface

Today, GUILS has evolved to a C-program where data retrieval is performed by Pro*C routines and where the graphics part is based on the CERN X-Window User Interface Management System [7]. GUILS allows the user to select a number of data systems (e.g. beam current, luminosity,..) followed by specific data items (e.g. positron current, instantaneous experimental luminosity,...) for a certain time period. The time period is introduced manually or can be obtained by selecting LEP modes for a specific LEP fill number. An external X-window is spawned, displaying the time-based result.

The interface makes it possible for any user to explore the Logging Database and to extract any data useful to him.

Data correlation

One of the original goals of the Logging Project was to be able to correlate the heterogeneous, asynchronous data, since many of these parameters may interact in unexpected ways and affect the LEP performance. In GUILS, the possibility is present to plot one parameter against another. For more complex correlations however, the users are advised to extract the flat data into an ASCII file and pipe it into their favorite mathematical application tool.

In fact, the data correlation function in GUILS does not perform the actual correlation inside the database. Searching for overlapping time periods while making complex joins of very large tables, results in unacceptable response times. The trick that has been applied is to get the data of the individual data systems and using the *union* operator. Afterwards, a small C-program can easily filter out the relevant data, making the correlation outside the database.

7 DATABASE PERFORMANCE

Today, the total amount of data in the LEP Logging Database is 4.3 Gbyte. Performance problems in terms of access speed while writing to or reading from the database are not observed under the present circumstances.
This is due to the fact that most pitfalls and bottlenecks have been carefully anticipated or avoided.

Data retrieval of time based data on a million-row table is reasonably fast. It is clear that a query on an non-indexed column results in an unacceptable wait time. Also time correlations between multiple asynchronous data systems should be handled outside the RDBMS.

During peak activities, up to 100 ORACLE sessions are open. Potentially dangerous actions by inexperienced clients are limited by the fact that they are obliged to retrieve data through a user account without resources.

The “cohabitation” of the two ORACLE instances on a single platform runs smoothly and unnoticed. The applications benefit from the improved processing power, without being penalized by the other instance (different priorities at the operating system level).

Problem Areas

One of the most important phases in the development of a database is the data modeling. The NIAM methodology has proven its thoroughness but today there is no up-to-date tool available, based on NIAM and mapping to Oracle7. Nevertheless, no matter which method or tool is used, one must not skip the design phase

since this leads inevitably to an expensive trap. Each data system must be carefully analyzed to avoid logging redundant and unnecessary data.

A major problem area is the continuous sequence of ORACLE releases. In order to follow the technology trend and to exploit new possibilities, the RDBMS is regularly upgraded. During this process however, new unexpected bugs are introduced which can take quite a while to surface. The increasing complexity makes it harder to diagnose and analyze a specific problem. The use of the Multi-Threaded Server with its shared servers and dispatchers in stead of dedicated connections, makes it difficult to trace the data flow of a specific transaction.

Future possibilities

New logging requests are still expressed and need to be satisfied. The trend set by the new technical features of Oracle7 will be surely pursued. Enforcing database security and data integrity and making full use of stored procedures will reduce software and improve overall performance.

8 CONCLUSION

After almost four years of working with the logging system, the LEP Logging Database has proven to be fully operational without performance problems despite the increasing size and number of users. There is a continuous flow of new logging requests, indicating the need for a central, long-term, reliable and accessible storage medium.

At CERN, the LEP Logging Database is still an important tool for LEP optimization and development, and is crucial for the energy scanning.

Throughout the years, the database has followed and survived several hardware and software evolution steps. Nevertheless, these upgrades must not be taken lightly in an environment of increasing complexity.

ACKNOWLEDGMENTS

I would like to thank Frédérick Bordry, the original Project Leader of the *LEP Logging* for his guidance and insight in database design and development.

REFERENCES

[1] R. Assmann et al., “The Energy Calibration of LEP in the 1993 Scan”, CERN-SL/95-02, 1995
[2] R. Billen et al., “LEP accelerator logging system using on-line database”, Nuclear Instruments and Methods in Physics Research, Section A 352, pp. 296-299 , 1994.
[3] J. Wintraeken, “The NIAM Information Analysis Method - Theory and Practice”, Kluwer, 1985
[4] RIDL* (V1.2) manual set, IntelliBase nv/sa, Antwerp, Belgium, 1991
[5] R.Saban et al., “The Control and Data Acquisition of the LHC Test String”, paper presented at ICALEPCS, 1995
[6] F. Bordry, J. Fritze, P. Ninin, “Graphical User Interface for Logging Systems”, CERN Note, 1993
[7] M. Vanden Eynden, “The X-Window User Interface Management System at CERN”, paper presented at ICALEPCS, 1995