APPLICATION OF ORACLE DATABASE FOR THE VUV-FEL DAQ SYSTEM

G. Dimitrov /Oracle 9i Certified Associate/ (PITZ) DESY, Germany, V.Rybnikov, DESY, Hamburg Germany

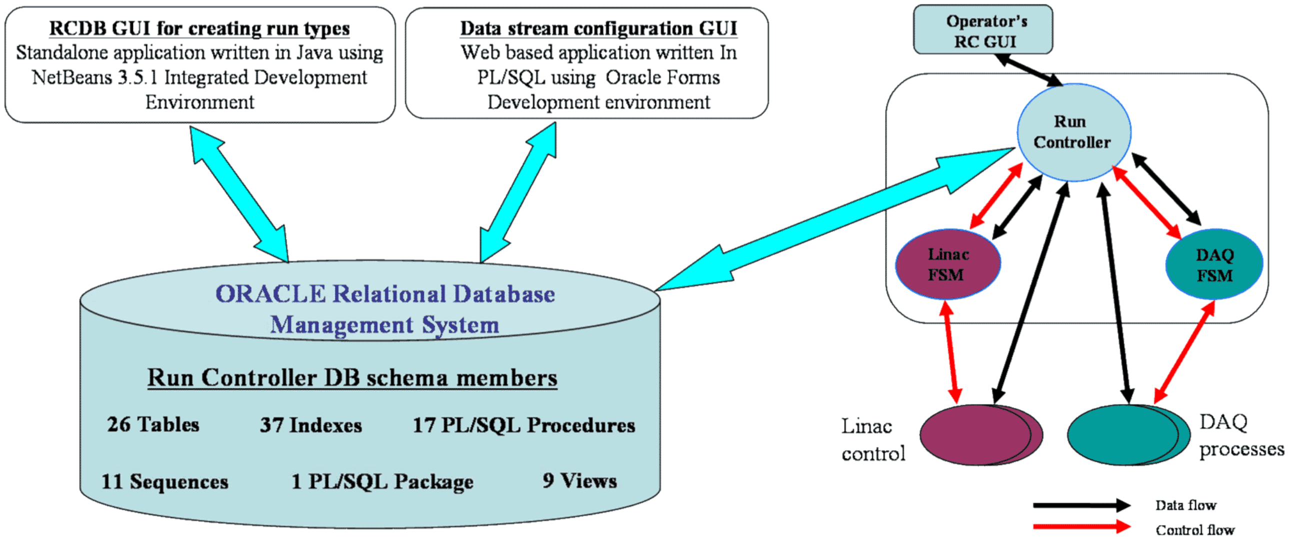

RCDB GUI for creating run types
Standalone application written in Java using NetBeans 3.5.1 Integrated Development Environment
Data stream configuration GUI
Web based application written In PL/SQL using Oracle Forms Development environment
Operator's RC GUI
Run Controller
Linac FSM
DAQ FSM
ORACLE Relational Database Management System
Run Controller DB schema members
26 Tables
37 Indexes
17 PL/SQL Procedures
11 Sequences
1 PL/SQL Package
9 Views
Linac control
DAQ processes
Data flow
Control flow

Figure 1: Run Controller DB, Run Controller server and client GUIs.

Abstract

One of the important issues in Data Acquisition System (DAQ) design is the type of storage for all kind of information used by the system. For the VUV-FEL (Vacuum-Ultraviolet Free Electron Laser) DAQ we have chosen ORACLE Relational Database Management System (RDMBS). The database stores not only the DAQ relevant information, but also control parameters for the LINAC. Data flow overview is shown in Fig.1

The paper presents the database structure and gives application of the stored information by the DAQ in detail.

INTRODUCTION

The RC database is the primary place for storing LINAC and DAQ configuration information. All configuration data follow the LINAC run modes [1]. The RC DB design foresees an easy way to extract all configurations for a certain run mode. This is achieved via the data base layout, which on one hand allows storing run mode specific information and on the other hand doesn't duplicate the common information used by all run modes.

Some LINAC and DAQ parameters can be associated with so called run parameters. The modification of a run parameter leads to the change in corresponding LINAC and DAQ parameters.

Every run mode usually has several setting versions that allow having the machine or the DAQ with different parameters within the same run mode. The new setting versions are stored by the operators after the machine optimization. Those settings can be reused in next runs.

The Run Controller makes use of the run mode setting versions to bring the LINAC and the DAQ in the corresponding mode.

Run Controller

The Run Controller (RC) [1] configures the systems participated in a run through the Distributed Object Oriented System (DOOCS) API [2] calls. It is responsible for configuration of both DAQ and LINAC parameters according to the LINAC run mode. The procedure of setting up a LINAC run consists of the following steps:

1. RC gets list of all run modes configured in the DB
2. the operator selects a run mode and a setting version of the machine from the DB
3. the operator can also select a set of subsystems within the selected run mode (e.g. some systems can have problems or their data is not included in the DAQ data flow).
4. some run parameters can be modified and sent to the RC
5. RC distributes the configuration information to the servers, brings Finite State Machines to the states according the definitions made in the DB. DAQ is being configured as well.

6. a new run number is generated. The machine is ready to produce the beam
7. the timing system swatches on the beam production and event collection

IMPLEMENTATION

The RCDB layout is shown in Fig.2. The data is well structured logically in tables. Basic definitions are kept in set of tables making up the catalog of definitions (the data is more or less static, supposed to be changed rarely). The actual values for the run mode setting version and run parameters are stored in 2 separate tables foreseen to have very large number of records /tens of million/. Therefore partitioning is applied for them. Partitioned tables allow data to be broken down into smaller, more manageable pieces called partitions, thus providing a structure that can be better tuned for availability and performance. [6]

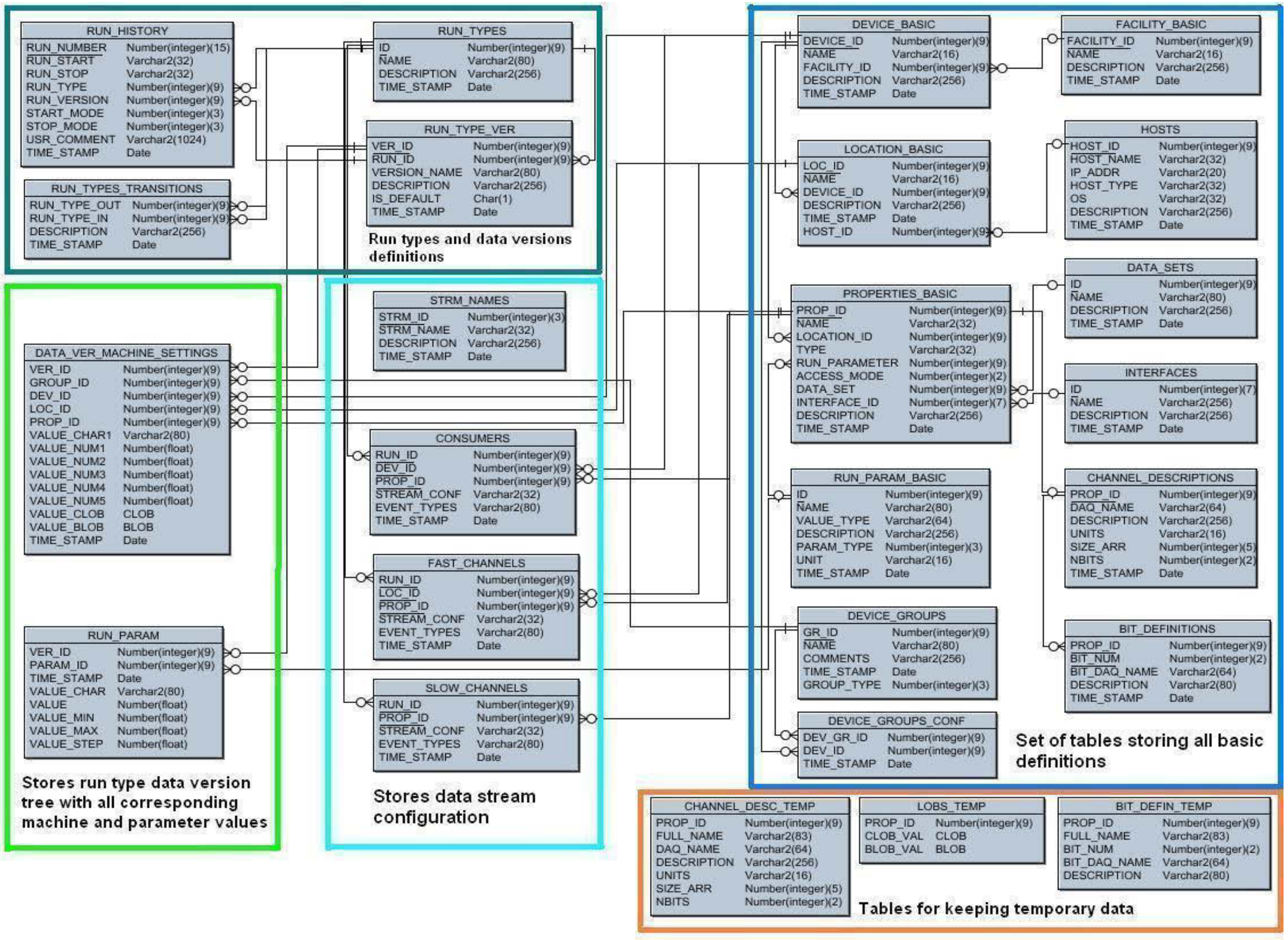
Integrity is provided through the use of such constraints as primary keys (unique identifiers that distinguish each row in a table), foreign keys (define special relationship between two tables), UNIQUE, NOT NULL and CHECK [3]

Run Control Data Base schema

The RCDB consists of the following entries:

- Regular tables of type HEAP /unordered collection/ for storing data.
- Indexes – objects, used to improve performance on SELECT statements
- Sequences – named sequential number generators, used to generate unique primary keys
- Stored PL/SQL procedures – compiled once and stored in executable form, so procedure calls are quick and efficient.
- Views – stored SELECT statement
- Packages - encapsulated collection of related program objects such as procedures, functions, user types

PL/SQL stands for “Procedural Language extensions to SQL” and provides procedural constructs, such as loops and conditional statements, that are not available in standard SQL [4]



RUN_HISTORY	
RUN_NUMBER	Number(integer)(15)
RUN_START	Varchar2(32)
RUN_STOP	Varchar2(32)
RUN_TYPE	Number(integer)(9)
RUN_VERSION	Number(integer)(9)
START_MODE	Number(integer)(3)
STOP_MODE	Number(integer)(3)
USR_COMMENT	Varchar2(1024)
TIME_STAMP	Date

RUN_TYPES	
ID	Number(integer)(9)
NAME	Varchar2(80)
DESCRIPTION	Varchar2(256)
TIME_STAMP	Date

RUN_TYPE_VER	
VER_ID	Number(integer)(9)
RUN_ID	Number(integer)(9)
VERSION_NAME	Varchar2(80)
DESCRIPTION	Varchar2(256)
IS_DEFAULT	Char(1)
TIME_STAMP	Date

RUN_TYPES_TRANSITIONS	
RUN_TYPE_OUT	Number(integer)(9)
RUN_TYPE_IN	Number(integer)(9)
DESCRIPTION	Varchar2(256)
TIME_STAMP	Date

Run types and data versions definitions

DATA_VER_MACHINE_SETTINGS	
VER_ID	Number(integer)(9)
GROUP_ID	Number(integer)(9)
DEV_ID	Number(integer)(9)
LOC_ID	Number(integer)(9)
PROP_ID	Number(integer)(9)
VALUE_CHAR1	Varchar2(80)
VALUE_NUM1	Number(float)
VALUE_NUM2	Number(float)
VALUE_NUM3	Number(float)
VALUE_NUM4	Number(float)
VALUE_NUM5	Number(float)
VALUE_CLOB	CLOB
VALUE_BLOB	BLOB
TIME_STAMP	Date

RUN_PARAM	
VER_ID	Number(integer)(9)
PARAM_ID	Number(integer)(9)
TIME_STAMP	Date
VALUE_CHAR	Varchar2(80)
VALUE	Number(float)
VALUE_MIN	Number(float)
VALUE_MAX	Number(float)
VALUE_STEP	Number(float)

Stores run type data version tree with all corresponding machine and parameter values

STRM_NAMES	
STRM_ID	Number(integer)(3)
STRM_NAME	Varchar2(32)
DESCRIPTION	Varchar2(256)
TIME_STAMP	Date

CONSUMERS	
RUN_ID	Number(integer)(9)
DEV_ID	Number(integer)(9)
PROP_ID	Number(integer)(9)
STREAM_CONF	Varchar2(32)
EVENT_TYPES	Varchar2(80)
TIME_STAMP	Date

FAST_CHANNELS	
RUN_ID	Number(integer)(9)
LOC_ID	Number(integer)(9)
PROP_ID	Number(integer)(9)
STREAM_CONF	Varchar2(32)
EVENT_TYPES	Varchar2(80)
TIME_STAMP	Date

SLOW_CHANNELS	
RUN_ID	Number(integer)(9)
PROP_ID	Number(integer)(9)
STREAM_CONF	Varchar2(32)
EVENT_TYPES	Varchar2(80)
TIME_STAMP	Date

Stores data stream configuration

DEVICE_BASIC	
DEVICE_ID	Number(integer)(9)
NAME	Varchar2(16)
FACILITY_ID	Number(integer)(9)
DESCRIPTION	Varchar2(256)
TIME_STAMP	Date

FACILITY_BASIC	
FACILITY_ID	Number(integer)(9)
NAME	Varchar2(16)
DESCRIPTION	Varchar2(256)
TIME_STAMP	Date

LOCATION_BASIC	
LOC_ID	Number(integer)(9)
NAME	Varchar2(16)
DEVICE_ID	Number(integer)(9)
DESCRIPTION	Varchar2(256)
TIME_STAMP	Date
HOST_ID	Number(integer)(9)

HOSTS	
HOST_ID	Number(integer)(9)
HOST_NAME	Varchar2(32)
IP_ADDR	Varchar2(20)
HOST_TYPE	Varchar2(32)
OS	Varchar2(32)
DESCRIPTION	Varchar2(256)
TIME_STAMP	Date

PROPERTIES_BASIC	
PROP_ID	Number(integer)(9)
NAME	Varchar2(32)
LOCATION_ID	Number(integer)(9)
TYPE	Varchar2(32)
RUN_PARAMETER	Number(integer)(9)
ACCESS_MODE	Number(integer)(2)
DATA_SET	Number(integer)(9)
INTERFACE_ID	Number(integer)(7)
DESCRIPTION	Varchar2(256)
TIME_STAMP	Date

DATA_SETS	
ID	Number(integer)(9)
NAME	Varchar2(80)
DESCRIPTION	Varchar2(256)
TIME_STAMP	Date

INTERFACES	
ID	Number(integer)(7)
NAME	Varchar2(256)
DESCRIPTION	Varchar2(256)
TIME_STAMP	Date

RUN_PARAM_BASIC	
ID	Number(integer)(9)
NAME	Varchar2(80)
VALUE_TYPE	Varchar2(64)
DESCRIPTION	Varchar2(256)
PARAM_TYPE	Number(integer)(3)
UNIT	Varchar2(16)
TIME_STAMP	Date

CHANNEL_DESCRIPTIONS	
PROP_ID	Number(integer)(9)
DAQ_NAME	Varchar2(64)
DESCRIPTION	Varchar2(256)
UNITS	Varchar2(16)
SIZE_ARR	Number(integer)(5)
NBITS	Number(integer)(2)
TIME_STAMP	Date

DEVICE_GROUPS	
GR_ID	Number(integer)(9)
NAME	Varchar2(80)
COMMENTS	Varchar2(256)
TIME_STAMP	Date
GROUP_TYPE	Number(integer)(3)

BIT_DEFINITIONS	
PROP_ID	Number(integer)(9)
BIT_NUM	Number(integer)(2)
BIT_DAQ_NAME	Varchar2(64)
DESCRIPTION	Varchar2(80)
TIME_STAMP	Date

DEVICE_GROUPS_CONF	
DEV_GR_ID	Number(integer)(9)
DEV_ID	Number(integer)(9)
TIME_STAMP	Date

Set of tables storing all basic definitions

CHANNEL_DESC_TEMP	
PROP_ID	Number(integer)(9)
FULL_NAME	Varchar2(83)
DAQ_NAME	Varchar2(64)
DESCRIPTION	Varchar2(256)
UNITS	Varchar2(16)
SIZE_ARR	Number(integer)(5)
NBITS	Number(integer)(2)

LOBS_TEMP	
PROP_ID	Number(integer)(9)
CLOB_VAL	CLOB
BLOB_VAL	BLOB

BIT_DEFIN_TEMP	
PROP_ID	Number(integer)(9)
FULL_NAME	Varchar2(83)
BIT_NUM	Number(integer)(2)
BIT_DAQ_NAME	Varchar2(64)
DESCRIPTION	Varchar2(80)

Tables for keeping temporary data

Figure 2: Run Controller DB layout.

Oracle Data Base to Run Controller interface

Interface between Run Controller and the Oracle DB is shown in Fig.3.

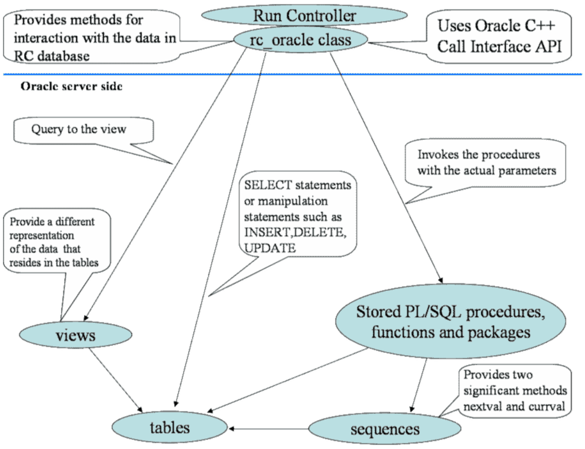

Run Controller
rc_oracle class
Provides methods for interaction with the data in RC database
Uses Oracle C++ Call Interface API
Oracle server side
Query to the view
Invokes the procedures with the actual parameters
SELECT statements or manipulation statements such as INSERT,DELETE, UPDATE
Provide a different representation of the data that resides in the tables
views
Stored PL/SQL procedures, functions and packages
Provides two significant methods nextval and currval
tables
sequences

Figure 3: Interface between Run Controller and Oracle.

An RC_oracle C++ class using Oracle C++ Call Interface (OCCI) API and provides several methods for interaction with the database for:

- Retrieving data - run modes list, basic definitions related to a chosen run, run parameters and machine setting defined for a certain run mode version and so on.
- Storing data:
    - machine values as new setting version of the current run mode;
    - run history information;
- creation new parameters;
- changing stream configurations

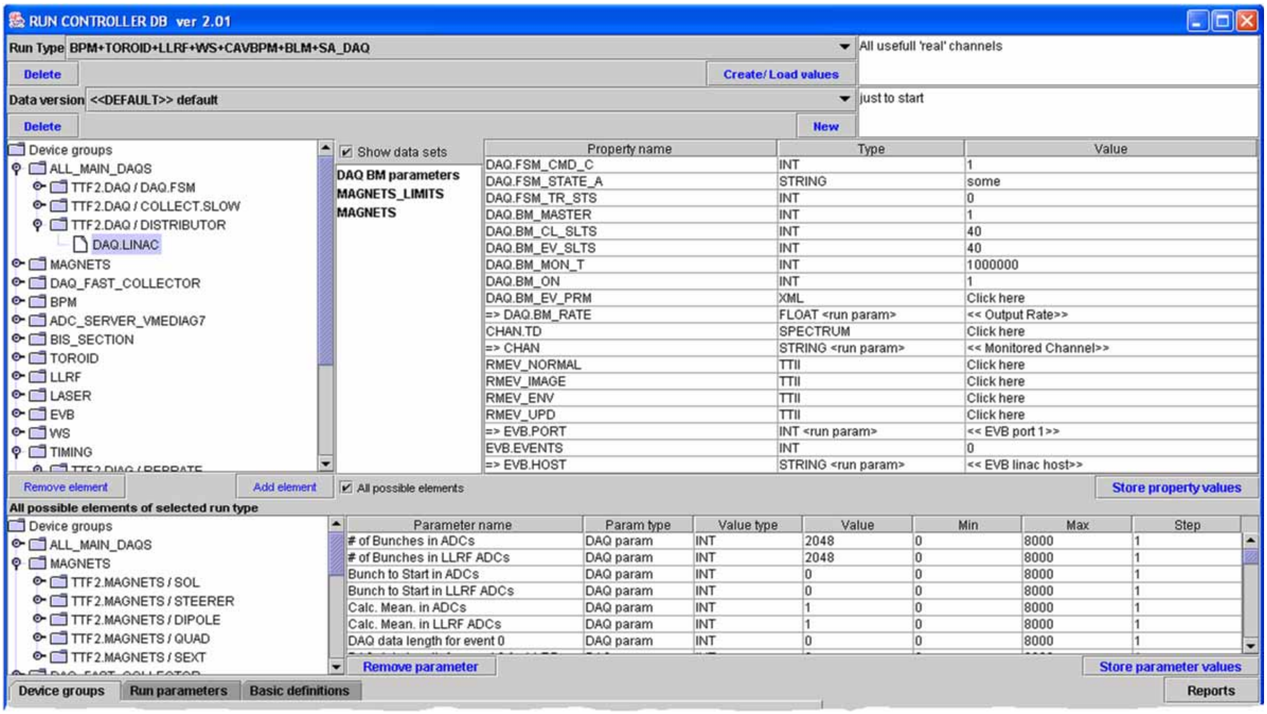
Run Controller Database GUI

The Run Control Database GUI is a standalone application written in Java using NetBeans 3.5.1 IDE [5]

DB client GUI gives a lot of possibilities for creating and configuring run modes and all attributes related to them. A part of the main dialog is shown in Fig. 4.

User specifies run mode name, corresponding data version name and some comments to them. Each run mode must have one 'default' setting version. Next step is to add to the run tree 'device groups' and 'run parameters'. The device groups and run parameters have to be predefined by the user in the DB via some additional dialogs. Navigating through the run mode tree one can easy see all properties for a certain device location and the values specified for them. The GUI supports most of the DOOCS data types and structures for data manipulation of the properties (e.g. INT, FLOAT, STRING, BOOL, IIII (4 INT), etc.) Access mode can be Read, Write, Read Write, Configuration and Check. They require different handling from the RC.

User can easily create a new run mode through the copying the tree structure and run parameters of an existing run mode into the new one, and after that modify its content to the desired structure. That can be done via adding new elements from the pattern Facility/Device/ Location/Property tree to the run mode tree. Additional feature is creating a new run setting version by copying the values from an existing one and modifying the values to the desired values.



Run Type BPM+TOROID+LLRF+WS+CAVBPM+BLM+SA_DAQ

All usefull 'real' channels

Data version <<DEFAULT>> default

just to start

Property name	Type	Value
DAQ.FSM_CMD_C	INT	1
DAQ.FSM_STATE_A	STRING	some
DAQ.FSM_TR_STS	INT	0
DAQ.BM_MASTER	INT	1
DAQ.BM_CL_SLTS	INT	40
DAQ.BM_EV_SLTS	INT	40
DAQ.BM_MON_T	INT	1000000
DAQ.BM_ON	INT	1
DAQ.BM_EV_PRM	XML	Click here
=> DAQ.BM_RATE	FLOAT <run param>	<< Output Rate>>
CHAN.TD	SPECTRUM	Click here
=> CHAN	STRING <run param>	<< Monitored Channel>>
RMEV_NORMAL	TTII	Click here
RMEV_IMAGE	TTII	Click here
RMEV_ENV	TTII	Click here
RMEV_UPD	TTII	Click here
=> EVB.PORT	INT <run param>	<< EVB port 1>>
EVB.EVENTS	INT	0
=> EVB.HOST	STRING <run param>	<< EVB linac host>>

Parameter name	Param type	Value type	Value	Min	Max	Step
# of Bunches in ADCs	DAQ param	INT	2048	0	8000	1
# of Bunches in LLRF ADCs	DAQ param	INT	2048	0	8000	1
Bunch to Start in ADCs	DAQ param	INT	0	0	8000	1
Bunch to Start in LLRF ADCs	DAQ param	INT	0	0	8000	1
Calc. Mean. in ADCs	DAQ param	INT	1	0	8000	1
Calc. Mean. in LLRF ADCs	DAQ param	INT	1	0	8000	1
DAQ data length for event 0	DAQ param	INT	0	0	8000	1

Figure 4: part of the RCDB GUI main dialog.

All DOOCS names which consist of Facility, Device, Location and Property names may be included in the DB either if a corresponding process is running or even if such a name doesn't exists or the corresponding DOOCS server is down.

Facility, device and location part of the name specifies a server process. These names are resolved by **E**quipment **N**ame **S**erver (ENS)[7] which give the real address of the DOOCS servers. The DOOCS properties in the DB can have one of the following access modes: Read, Write, Read Write, Check and Configuration.

Data Stream Configurator

The task of the Stream configurator is to assign event consumer processes, slow channel properties and fast channels properties to certain data streams on the run mode level. Streams construct logically lists of channels. Every list presents a data flows handled by distributor and event builder processes [1].

Data Stream Configurator is a Web based application written in PL/SQL using Oracle Forms Development Environment. Typical three-tier architecture is in action – Web client (browser), an application Web Server and Oracle Database server. The browser on the client side uses HTTP to communicate with the Web server to make a connection request. Application on the web server processes the request and sends it to the DB server

CURRENT STATUS

The RCDB is being used for the DAQ system tests since autumn 2004. At the moment there is one run mode configured, that includes all required DAQ and LINAC processes to provide the beam from the injector up to the undulator. The number of properties for this mode made up ~ 4300 entries. To boot this run mode to the RC takes ~ 4 sec. To save a new setting version from RC to the DB is about 5 sec. We don't expect the increase of the boot and storing time even in case of several times increase of properties. The way we achieve that is by fine database tuning:

- All indexes are stored in a tablespace (logical database structure) on different I/O drive separately from the storage where the tables are located

- SELECT statements are written in a way to avoid full table scans for the large tables, so that the processing time doesn't depend on the number of records

- Performance gain in storing data is achieved by developing different PL/SQL procedures and packages. Avoid loops using intensively collections (ordered group of elements) with BULK SELECTS, INSERTS, UPDATES or DELETES statements.

CONCLUSIONS

- All DAQ configuration data is on one secure place.
- No duplication
- Data integrity is preserved
- The data is well structured and easy for storing and retrieving.
- Oracle RDBMS provides a lot of possibilities for easy object development and maintaining.

REFERENCES

[1] V. Rybnikov, "Data acquisition system for VUV-FEL LINAC", these proceedings

[2] G.Grygiel, O.Hensler, K.Rehlich, "DOOCS: a Distributed Object Oriented Control System on PC's and Workstations", PCaPAC96, DESY, Hamburg, October 1996.

[3] Jason S. Couchman, "OCP Oracle9i Database: Fundamentals I Exam guide, 2002 ISBN 0-07-219540-1

[4] Chip Dawes, "OCA/OCP Introduction to Oracle9i SQL: 2002 ISBN 0-7821-4062-9

[5] NetBeans IDE for developing Java applications http://www.netbeans.org

[6] Managing Partitioned Tables and Indexes http://oraclesvca2.oracle.com/docs/cd/A91202_01/901_doc/server.901/a90117/partiti.htm

[7] Equipment Name Server (ENS) http://tesla.desy.de/doocs/doocs.html