DABATASE FOR NSLS-II ACCELERATOR OPERATION[*]

J. Choi[†], G. Weiner, T. Shaftan, R. Farnsworth, X. Yang BNL, Upton, NY 11973, USA

Abstract

NSLS-II is employing a database and corresponding user interfaces which are used for the accelerator data sharing and management [1]. The database include operation related information such as beam optics parameters, magnet measurement data, survey data and operation summary. To improve the usability, other functionalities are also being added. However, due to the limited scope, the general expectation of the overall facility cannot not be met and, in order to solve the issue, we are in the process of adopting Component Database (ComponentDB) [2] developed at Advanced Photon Source (APS). This paper shows the current status of the process.

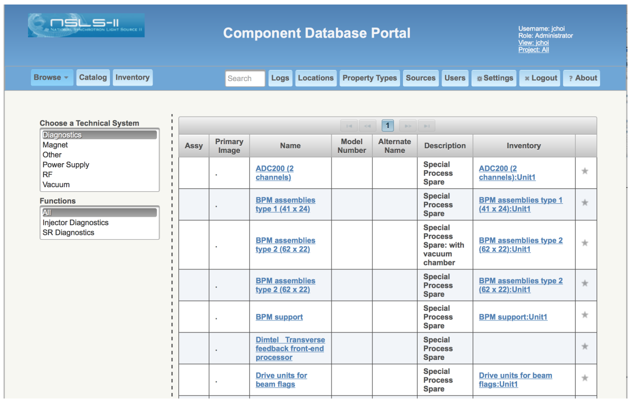
INTRODUCTION

During the construction period of NSLS-II project, a database was developed to help commissioning as a part of EPICS database (IRMIS: Integrated Relational Model of Installed Systems) [3]. The main motivation was fast and easy access to the information which may be needed during the commissioning, e.g. aperture or survey data at a specific location. It included the magnet measurement data and the commissioning lattices, and also provided functions to view the archived documents and share the commissioning procedures and results.

As the user operation starts, the database is renamed as operational database and the graphical interfaces are added to review the previous user operations. Meanwhile, NSLS-II IRMIS development wend out of the project scope and is not expected to be resumed in the near future. On the other hand, due to the limited scope, the operational database could not replace the IRMIS and the general expectation of the overall facility cannot not be met and requests are made that the operational database need to be extended to manage systems like inventory and maintenance procedures. However, considering its urgency as already operating machine, developing such a full system in a short period is not feasible with the given limited resources,

Therefore, instead of developing the new database system, we decided to adopt Component Database (ComponentDB) developed by Advanced Photon Source (APS) control group for the APS upgrade project (APSU) [2]. Because CDB is utilizing highly flexible mechanism which provides object dependent properties at any layer, it is thought to be easily applied to any facility such as NSLS-II with minor modification. At this moment, we are applying to very limited sample data to understand the full mechanism, and a CDB interface page for NSLS-II is shown in Fig. 1.

* Work supported by DOE Contract No: DE-SC0012704
† jchoi@bnl.gov



Assy	Primary Image	Name	Model Number	Alternate Name	Description	Inventory	
	.	ADC200 (2 channels)			Special Process Spare	ADC200 (2 channels):Unit1	★
	.	BPM assemblies type 1 (41 x 24)			Special Process Spare	BPM assemblies type 1 (41 x 24):Unit1	★
	.	BPM assemblies type 2 (62 x 22)			Special Process Spare: with vacuum chamber	BPM assemblies type 2 (62 x 22):Unit1	★
	.	BPM assemblies type 2 (62 x 22)			Special Process Spare	BPM assemblies type 2 (62 x 22):Unit1	★
	.	BPM support			Special Process Spare	BPM support:Unit1	★
	.	Dimtel Transverse feedback front-end processor			Special Process Spare		★
	.	Drive units for beam flags			Special Process Spare	Drive units for beam flags:Unit1	★

Figure 1: A CDB interface page for NSLS-II

COMPONENT DATABASE (CDB) [2] [4]

The Component Database (CDB) is a tool developed by APS control group to assist in documenting, organizing, and tracking the components planned for use in the MBA accelerator. It helps capturing component documentation, provides a repository for inspection and measurement data (e.g., travellers), and supports logging of component history through the component's life cycle. CDB also serves as a user portal for finding all known information about a particular component or a design. To that end, it provides links and interfaces to external systems commonly used at APS, such as various drawing and document management systems, procurement applications, etc. As the successful substitute for the IRMIS, CDB is expected to capture a complete "Bill of Materials" for the MBA.

BASIC CONCEPTS IN CDB

In CDB, every entry is called **item** whether it is a tangible or not. The item can be component, inventory or location and parent-child relationship is assigned through the concept of **element**.

Domain

An items should belong to one of the 3 domains, which are location, component and inventory. The items will be managed properly according to their domains

Component item

A component item refers a distinctive object, conceptual or physical, which has a its own meaningful role in accelerator operation.

The examples are model, prototype, and design. They should have name, item type, owner, owner group. The additional properties like project, subtype, model number, alias, or source (manufacturer or vendor) can be assigned when the item is created or later. If the item is a physical item, location and code number can also be assigned.

Inventory item

An inventory item is a tangible instance of a component item. They should have name, corresponding component item, location, owner, owner group. The code number and image can be also assigned.

Element

An inventory or component item as a constituent of the parent item. Even for the element, the corresponding component should be already registered in the database. Then, after the container item is created, the elements can be included in the container. Even a item without any constituent child component is considered an element without a contained item. That is every component and inventory item is an element itself.

PROPERTY

The most prominent feature of CDB is the flexible property mechanism. Different from the traditional concept of object properties, where they have fixed columns in the object table, CDB has separate property tables where properties are defined with the corresponding specifications and handlers. By having separate tables for property type, category, value, handler and meta data, the flexibility can be maximized. If new property type and category is needed, they can be created anytime probably together with corresponding handler and meta data. By the optional handler, different treatment is enabled depending on the property type when a service is requested from the client. The meta data keep the specific information for some special properties. For example, for the spare properties, the minimum required number is given by the property meta data.

CDB TABLES

In this section, we will show some tables in the database to help the understanding how CDB is working.

When an item is created, it will recorded in the table **item**. At the same time, it has entries in **item_element** and **entity_info** where additional information is saved which is not included in **item** or **item_element**. When a child element is added to a container element, it is recorded in **item_element** and **entity_info**. The items are characterized by categories and types in **item_category** and **item_type** tables and the relations between categories and types are defined in **item_category_item_type**

If there is a relation between two **item_element** entries the relation is written at **item_element_relationship** where the relations are defined at **relationship_type**. The relationship is also written at **item_element_relationship_history** and the changes in relationships are logged in the history table. The typical relationship is the location and we can find the location history of an item from the history table. Relationship also has property and it can be found or written at **item_element_relationship_property** table.

The property related tables implementing the flexibility as described in section **PROPERTY** are **item_element_property**, **property_type**, **property_value** **property_type_category**, **property_type_handler** and **property_ metadata**.

Also, several linked tables are servicing the log system. They are **log**, **log_topic**, **item_element_log** and **system_log** and tables **user user_info**, **user_group** and **user_user_group** are used for user entries. And table **source** is prepared for the information of manufacturers and vendors.

TECHNOLOGY

Simply speaking, what we want is implement is saving and retrieving user's data through convenient interfaces. Therefore, what we need are a data source and user interfaces which can query the data. Many tools are available for these jobs and CDB chose the object-oriented tools which are proven to be efficient and work together without friction.

- Database: MySQL [5]
  MySQL is an open-source relational database management system (RDBMS).
- Web Portal: GlassFish [6]
  GlassFish is an open-source application server project implementing Java Enterprise Edition (EE) developed by Oracle Corporation. GlassFish provides Integrated development environment (IDE) support and can be easily integrated with MySQL.
- REST [7] Web Service: CherryPy [8]
  REpresentational State Transfer (REST) is a style of software architecture implementing web services. Because of its simple style it is relatively easy to use and CherryPy is object-oriented web framework python library which can implement REST.
- Java EE: JSF (PrimeFaces) [9], JPA
  PrimeFaces is a lightweight Java library which enables to create the hierarchical web interface conveniently. It is also equipped with data model which can query the data source. The Java Persistence API (JPA) is used to control the results from the queries.
- Python ORM: SQLAlchemy [10]
  Object Relational Mapping (ORM) is a technique of querying which converts the data between different languages. SQLAlchemy is the Python library for SQL toolkit and ORM which was developed to provide the efficient SQL query.
- Python-Exel: XLRD, XWT [11]
  Python XLRD and XWT are libraries for reading and writing the MS-EXEL spreadsheets. The library is not used in CDB itself but used for moving the existing data to the database.

DATA TO BE SERVICED

Because CDB is providing very convenient environment, all kinds of data can be serviced with their own properties and interfaces. They include hardware and software element, physics parameters, drawings, and documents. The preliminary list of the data is as follows.

Accelerator Physics analysis/calculations data

- Global Accelerator Parameter List
- Lattices in Elegant/Tracy formats
- Linear optics
- Beam Intensity
  Collective effects
  Losses
  Synchrotron and undulator radiation
- References and electronic texts of textbooks
- Project software libraries

Element/Device Descriptions

- Magnets
- Supports
- Power Supplies
- Vacuum devices
- Diagnostics and Instrumentation devices
- RF elements
- Control system devices
- Insertion Devices

Auxiliary systems

- Electrical power distribution
- Cable distribution and cable trays
- Water system
- Compressed air system
- Liquid gas system
- Primary and secondary alignment systems

Others

- Project Drawings
- Subsystem block-diagrams and functional schematics
- Publications

DISCUSSION

For a machine which already has been operating for several years, the data of NSLS-II cannot be manually input through the web interfaces. Therefore, for each format of the data, we are developing the python scripts moving existing data to a spreadsheet file with multiple sheets where each sheet has one-to-one correspondence to the database and dump each spreadsheet to a database table. Developing scripts is straightforward once the given data structure is identified. On the other hand, finding out the data in various archives and verifying the validity is very difficult and time-consuming. Therefore, we will gather any data seemingly useful and publish internally so that the corresponding groups can verify or replace them. At the same time, each group will be requested to collect and submit the missing data. Meanwhile, the basic structures, such as properties, types, categories, log types will be optimized.

For the maintenance management, which seems not in the primary concerns of CDB yet because it was developed for APS upgrade, we have a good example named CATER (Computer Aided Trouble Entry and Reporting) [12] [13] which has been used for a long time at SLAC. We hope to understand CATER also and add the functions to the NSLS-II CDB.

ACKNOWLEDGMENT

The authors would like to stress again that the ComponentDB is fully developed by APS Controls Group and this work is about understanding and applying the CDB. The authors thanks to APS Controls Group, especially to Dariusz Jarosz and Ned Arnoldfor for providing CDB.

REFERENCES

[1] J. Choi, T. Shaftan, “Database Development for NSLS-II Accelerator Data Management,” Proc. of IPAC2016, Busan, Korea, p.4137, 2016.

[2] S. Veseli, N.D. Arnold, D.P. Jarosz, J. Carwardine, G. Decker, N. Schwarz, “Component Database for the APS Upgrade,” Proc. of ICALEPCS2015, Melbourne, Australia, 2015.

[3] D. Dohan, “IRMIS Universal Component-Type Model,” Proc. of ICALEPCS07, Knoxville, Tennessee, p.82, 2007.

[4] CDB/APSU, `https://confluence.aps.anl.gov/display/APSUCMS/Home`

[5] MySQL, `https://www.mysql.com`

[6] GlassFish, `https://glassfish.java.net`

[7] Representational State Transfer ,`http://www.service-architecture.com/articles/web-services/representational_state_transfer_rest.html`

[8] CherryPy, `http://cherrypy.org`

[9] PrimeFace, `https://www.primefaces.org`

[10] SQLAlchemy, `https://www.sqlalchemy.org`

[11] Python-Exel, `http://www.python-excel.org`

[12] C. W. Allen, S. Anderson, R. Erickson, W. Linebarger, J. C. Sheppard, and M. Stanek, “Opportunistic or Event-Driven Maintenance at the Stanford Linear Accelerator Center,” SLAC-PUB-7424, 1979.

[13] R. Sass, H. Shoaee, “CATER: An Online Problem Trcking Facility for SLC,” Proc. of PAC1993, Washington, DC, p.1946, 1993.