DATABASE FOR ACCELERATOR MODELING*

P. Chu#, Y. Zhang, FRIB, East Lansing, MI 48824, USA
D. Dohan, G. Shen, BNL, Long Island, NY 11973, USA
J. Wu, SLAC, Menlo Park, CA 94025, USA
H. Lv IHEP, Beijing, China

Abstract

A database for model data is design for the Facility for Rare Isotope Beams (FRIB) Project. The database schema design takes most general approach which is not limited to FRIB models. Programmatically access to the database can be done through a set of Application Programming Interfaces (APIs). Initial data population demonstrates that the database is suitable for XAL application framework [1]. The model database is also part of collaboration for complete database needs among various data domains across an accelerator.

INTRODUCTION

High-level applications for accelerator control are often based on physics models which require various data for the calculation. FRIB physics application architecture shown in Fig.1 as an example, starting from left-hand side an offline modelling code with a given lattice computes a model; both the computed model and lattice settings are then saved in a lattice/model database which is part of a global database; a model service then servers up model data by either extracting offline model data from the database or computing online machine model in real-time. Physics applications, as clients of the model service, can then carry out machine tuning based on the model data provided by the service. With this software architecture design, database and its data service is vital for physics applications. This paper will report the lattice and model database design and the progress for the associated data services.

LATTICE AND MODEL DATABASE DESIGN

A global database which covers many domains of an accelerator including lattice, model, magnet measurement, survey and alignment, and machine setting save set, is under development. The global database development is a collaborative effort among different institutions. In order to distribute development effort among several collaborative institutes, it is necessary to divide the entire global database into domains. The lattice and model database domain provides coverage for any lattice and model data storage needs. Because the lattice data and model data are tightly coupled, they are developed as one single domain module. The lattice and model database design is mainly according to the data storage needs, usefulness for physics applications, flexibility of various modelling tools and standardized data access API. The global database design is based on the Integrated Relational Model of Installed Systems (IRMIS) [2].

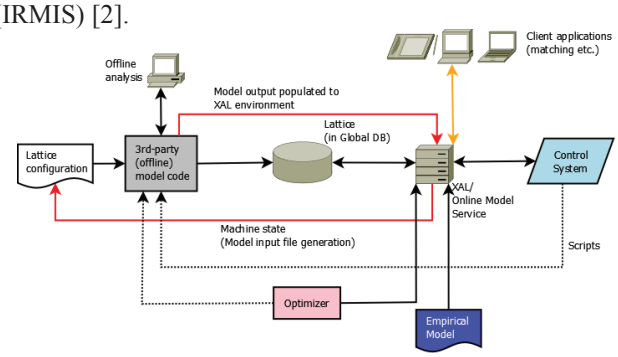

Figure 1: FRIB physics application architecture.

Data Coverage for Lattice and Model Database

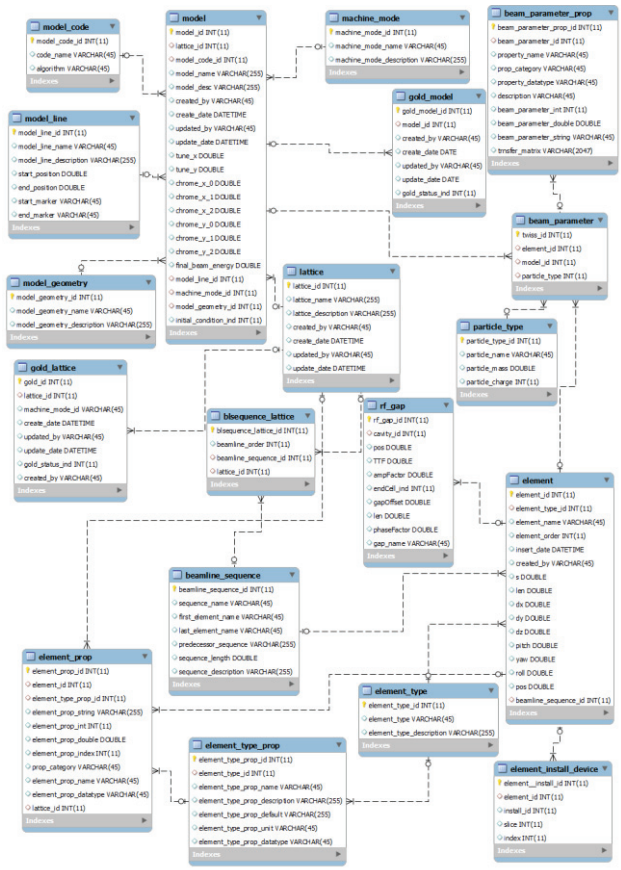

Figure 2: Lattice and model database schema.

Lattice and model database can hold 1) lattice data which is a layout along the beamline plus a set of active

*Work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661 and DE-AC02-98CH10886 and Michigan State University, NSLS-II Project and CSNS Project.
#chu@frib.msu.edu

ISBN 978-3-95450-122-9

device settings with optional diagnostic devices; and 2) model data which is the result of physics simulation for a particular beam running through a lattice. There are a total of 19 tables for the lattice and model database which can hold any number of lattices and model data sets, and link to the installation database domain which may contain controls information for devices.

The schema has a "gold lattice" table and a "gold model" table to keep track of default lattices and models for each beam-line and machine mode. The API signature for accessing default lattice is based on physicists' point of view, e.g. *getDefaultLatticeForbeamline("a_line")* is the API for obtaining the default lattice for a beamline. For non-default lattices, a unique database set identification (ID) number is required in the API.

Brief description for each database table is below:

- Beam Parameter: A link table between an element and the beam properties such as Twiss Parameters at the element.
- Beam Parameter Property: Individual beam parameter for each element. This table is in property name-and-value pair format so it can accommodate any modelling tool's outputs.
- Beamline Sequence: Typically an accelerator is divided into multiple beamline segments for physics tuning and modelling convenience reasons. This table records the information for each individual beamline segment.
- Beamline Sequence Lattice: A link table between the Beamline Sequence table and the Lattice table.
- Element: Information for each modelling element including misalignment.
- Element Install Device: A link table between the lattice and model database domain, and the installation/configuration database domain.
- Element Property: Element properties for each element.
- Element Type: Definition for modelling element types.
- Element Type Property: Common properties for a given element type.
- Gold Lattice: Track records for current gold (or default) lattices and previously tagged as gold lattices.
- Gold Model: Track records for current gold (or default) models and previously tagged as gold lattices.
- Lattice: General information for a lattice.
- Machine Mode: Examples for machine mode are "design", "extant", and "user-defined".
- Model: General information for a model.
- Model Code: Modelling code and its algorithm information.
- Model Geometry: Geometry information for a model.
- Model Line: Information about the start and end elements for a given model. A model line can contain multiple beamline sequences.
- Particle Type: Particle species used in model calculation.
- RF Gap: Specific information about RF accelerating gaps. RF cavities are the modelling elements and RF gaps are structures within an RF cavity.

So far, the lattice and model database schema can accommodate all data needs. However, further performance and storage efficiency optimization is needed.

Properties Stored as Name/Value Pairs

The lattice and model database stores property name-and-value pair in the database to accommodate various modelling tools' needs. The name/value pair approach is in contrast to the conventional way of using the property's name as the database table column label. If a new property is introduced or a property name is changed, one has to modify the conventional database and consequently the affecting data access API(s). Such tedious maintenance can be avoided with the property name-and-value pair approach. Also, property name/value can accommodate various properties from different modelling tools.

Data Link among Domains

Some of the data saved in other domains of the global database are needed for physics model computation. There are a couple of ways to link the data across domains. First, using primary/foreign keys between database tables can avoid data duplication. However, this approach may result domains tightly bounded with data dependency and cannot be deployed independently. Alternatively, each domain maintains self-contained data sets with software services as the interface for accessing multiple domains to provide a complete set of joint query. The latter is, however, at the price of potential data duplication and inconsistency. It depends on the situation whether using database table linkage or service interface.

DATA PREPARATION

As shown in Fig.1, typically lattice is designed with offline simulation modelling tools. Each offline modelling code has its own format and, therefore, very difficult to write a universal data upload program for various modelling codes.

Standard Spread Sheet for Lattice Data Upload

A complete set of design lattice can be uploaded with a batch data upload program.

- Beamline Sequence – Breaking the entire accelerator into segments suitable for physics modelling and beam tuning purposes.
- Elements – A spread sheet contains a flat list of all modelling elements with information such as names, locations, effective lengths, and nominal settings.
- RF Gaps – RF accelerating gap data special for XAL, e.g. gap electric-to-geometrical centre offset, and end-cell indicator.

- Beam Initial Conditions – Beam initial condition (particle type, rest mass, charge, beam intensity, phase space coordinates, and Twiss Parameters) for each beamline segment.
- Name Mapping – Name mapping between physics names and engineering names used for control purpose.
- Device Types – Defining device types according to site specific naming convention document.
- Device-model Types – Defining how to model a given device type, e.g. dipole correctors can be modeled as either thin lens or thick lens, device type "DH" can be modeled as Bend in XAL or sector bend in MAD.
- TTF Curves – Transit-time Factor (TTF) curves, a polynomial curve fit program fits the curves and the fitted polynomial coefficients are saved in database with corresponding RF cavities.

LATTICE AND MODEL API

Typically, accessing data in a relational database is done through Sequential Query Language (SQL). However, it is not intuitive for physicists to use SQL. Also, application software written in modern programming language often uses object-oriented approach. An object to relational mapping (ORM) API can facilitate the database query. Java Persistence API (JPA), which is included in the Java Enterprise Edition Version 2 (J2EE), is chosen as the ORM tool. A set of data API based on JPA is written for easy database access. In addition, database applications and services can take advantage of the data API. A proof-of-principle lattice service based on Enterprise Java Bean (EJB) is developed. The service is demonstrated with J2EE Glassfish Application Server. Typical data API follows the EJB conventions, for instance, the API for setting Twiss Parameters for a given element looks like *model.setTwissFor("an_element", Twiss_Parameters)* where *an_element* is the element name and *Twiss_Parameter* is a collection of Twiss Parameters. Because the API is written in Java, it is compatible with MATLAB and Python scripting languages with proper configuration and additional modules.

Model Data Upload

For each modelling code, an "adapter" program is needed to extract the model output and then to write the data into the database via the data access APIs. For non-Java modelling code, the easiest way is to write a run-control program in scripting language such as Python which is for starting a model run, monitoring the progress, extracting the result files and uploading the model data to the database.

Individual Lattice Data Upload and Extraction

In addition to the batch lattice data upload, individual data can also be uploaded to or extracted from the database via data access APIs. A web based user interface (UI) is under development for end users to update the database.

Modelling Tool Input File Generation

Modelling tool input files for lattice, element settings, and initial beam conditions can be generated directly from the database using the data access API set. Currently, a program for generating XAL configuration files is implemented. The XAL files generated by the program are tested with the XAL Online Model.

Lattice Data Validation

Data uploaded to the database should be validated to ensure data integrity. Typically the data check is done by programmatically extracting the data from the database, running physics simulation code against the extracted data, and comparing the results with the original data.

Lattice and Model Data Service

Service-oriented architecture for high-level applications can provide better performance and efficiency [4, 5]. Model data can be updated periodically as a running service.

As mentioned above, proof-of-principle service based on J2EE EJB convention has been developed. Glassfish Web Application Server can host such services. The services will be standard interface between lattice/model data and client physics applications.

CONCLUSION

Lattice and model database as part of a global database has been designed and implemented. Lattice data template was defined for a standard data upload program. As a first demonstration, XAL configuration files are generated programmatically from the database. Lattice and model data API set is written for easy data access. Based on the data API, a proof-of-principle service is developed. Near feature plan is to further develop services and UI.

ACKNOWLEDGMENT

The authors would like to thanks other team members involved in the global database development for their valuable discussions and technical recommendation.

REFERENCES

[1] J. Galambos, et al, "XAL Application Programming Structure", p. 79, Proceedings of 2005 Particle Accelerator Conference.
[2] http://irmis.sourceforge.net/
[3] http://poi.apache.org/
[4] P. Chu, et al, "FRIB High-Level Software Architecture", Proceedings of 2012 International Particle Accelerator Conference.
[5] P. Chu, et al, "Online Physics Model Platform", Proceedings of 2012 International Particle Accelerator Conference.

ISBN 978-3-95450-122-9