A Prototype of the SNS Optics Database

BNL/SNS TECHNICAL NOTE

NO. 085

Nikolay Malitsky

November 9, 2000

COLLIDER-ACCELERATOR DEPARTMENT
BROOKHAVEN NATIONAL LABORATORY
UPTON, NEW YORK 11973

A Prototype of the SNS Optics Database.

Nikolay Malitsky

1. Introduction

Database is an important part of the modern software environment. It aims to consolidate diverse data sets and facilitate their analysis and management. Accelerator data can be distributed into several domain databases, such as Naming, Survey, Magnet, Optics, and others. Each domain is maintained by the corresponding group of specialists and is associated with their particular applications. The Optics database is a persistent representation of the accelerator model and provides accelerator physicists with data of " all elements that can influence single particle motion (in their idealized operation) and only those elements … "[1]. Usually, the schema of the Optics database reflects the structure of existing accelerator codes. In 1992, the LAMBDA collaboration [2] had introduced the most complete implementation of the Optics database employing the MAD model. Since that time, a new generation of object-oriented accelerator packages has been introduced and developed. For the SNS ring applications, we are using the Unified Accelerator Libraries (UAL) environment [3] that currently joins together six object-oriented libraries, such as PAC, TEAPOT, ZLIB, ACCSIM, ORBIT, and ALE. The paper presents a version of the Optics database representing the UAL accelerator model.

2. Schema of the Optics Database

The schema of the Optics database has been implemented after the UAL accelerator object model using the object-relational mapping approach [4]. Figure 1 illustrates the structure of the UAL accelerator model:

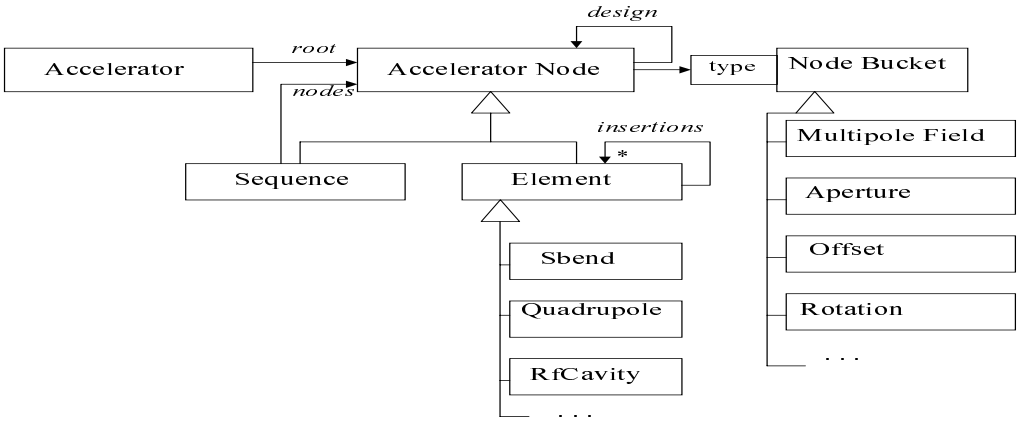

Figure 1: UAL accelerator object model.

The model is based on three entities:

- *Accelerator* is a hierarchical tree of accelerator nodes, elements and sequences of elements.
- *Accelerator Node* is a basis class of an accelerator element and sequence of elements. There are many different types of accelerator nodes (e.g. sbend, quadrupole, *etc*.) But all of them have the same structure: an open collection of accelerator node buckets.
- *Accelerator Node Bucket* is a set of attributes relevant to the single physical effect or feature (e.g. magnetic field, aperture, offset, rotation, *etc.*)

The following sections describe the representation of these objects in the Optics database.

2.1. Accelerator

In the UAL model, an accelerator is described as a hierarchical tree of accelerator nodes. This definition is very general and can be applied to different accelerator machines, their components, or transfer lines. Usually, the modern accelerator complexes contain several accelerators and each of them is developed by a separate group of scientists. To avoid naming collisions among different teams, each accelerator identifier determines a separate naming space for its elements and is used as the first component of combine element names in the Optics Database. All accelerator identifiers are located in the following table:

```
create table accelerators (
  accelerator_id              varchar2(30),
  primary key (accelerator_id)
);
```

The UAL accelerator description is highly modularized. An accelerator structure is composed of independent entities, accelerator nodes. Each accelerator node, in turn, is built from the collection of separate element buckets, orthogonal sets of element attributes. Such organization facilitates a fine-grain approach for the versioning of accelerator data. In our scheme, the accelerator description has three independent version attributes: *lattice_id*, *state_id*, and *source_id*. The first attribute is associated with an accelerator structure. The *state_id* attribute represents a set of element buckets used in the particular application; *source_id* defines the origin of element data (e.g. expectation, measurement, *etc*.). In the Optics database, each of these attributes has its own table:

```
create table accelerator_lattices (
  accelerator_id              varchar2(30),
  lattice_id                  varchar2(30),
  primary key (accelerator_id, lattice_id),
  foreign key (accelerator_id) references accelerators (accelerator_id)
);
```

```
create table accelerator_states (
  accelerator_id        varchar2(30),
  state_id              varchar2(30),
  primary key (accelerator_id, state_id),
  foreign key (accelerator_id) references accelerators (accelerator_id)
);

create table accelerator_sources (
  accelerator_id        varchar2(30),
  source_id             varchar2(30),
  primary key (accelerator_id, source_id),
  foreign key (accelerator_id) references accelerators (accelerator_id)
);
```

A set of accelerators together with their version attributes is associated with *system_id* . All these associations are stored in the *accelerator_systems* table:

```
create table accelerator_systems (
  system_id             varchar2(30),
  accelerator_id        varchar2(30),
  lattice_id            varchar2(30),
  state_id              varchar2(30),
  source_id             varchar2(30),
  primary key (system_id, accelerator_id),
  foreign key (accelerator_id, lattice_id) references accelerator_lattices (accelerator_id, lattice_id),
  foreign key (accelerator_id, state_id) references accelerator_states (accelerator_id, state_id),
  foreign key (accelerator_id, source_id) references accelerator_sources (accelerator_id, source_id)
);
```

2.2. Accelerator Node

Accelerator node is a basis class of various types of accelerator elements and sequences of elements. In the Optics database, this class is represented by the corresponding table:

```
create table accelerator_nodes (
  accelerator_id        varchar2(30),
  node_id               varchar2(30),
  length                float(126) not null,
  design_id             varchar2(30),
  node_type             varchar2(30),
  primary key (accelerator_id, node_id),
  foreign key (accelerator_id) references accelerators (accelerator_id),
  foreign key (node_type) references accelerator_node_types (node_type),
);
```

The first two fields, *node_id* and *accelerator_id*, form a composite element identifier that provides the referential integrity of the whole Optics database. The third field is an effective *length* of the accelerator node. The optional attribute *design_id* is a reference to the design element. This reference allows one to support the relationship between

operational and design accelerator descriptions. The discriminator *node_type* specifies the type of the accelerator node.
The UAL model preserves all MAD element types (such as sbend, quadrupole, rfcavity, etc.) and defines the mechanism for introducing new ones. A predefined set of legal UAL node types is located in the *accelerator_node_types* table:

```
create table accelerator_node_types (
  node_type                varchar2(30),
  primary key (node_type)
);
```

Elements of different types have the same structure and do not require to be represented by specialized tables. Contrary, the Sequence, a composite node in an accelerator hierarchical tree, introduces an one-to-many relationship between composite and contained nodes. According to the object-relational mapping approach, this association is implemented in the following table:

```
create table accelerator_sequences (
  accelerator_id           varchar2(30),
  lattice_id               varchar2(30),
  sequence_id              varchar2(30),
  node_id                  varchar2(30),
  position                 float(126) not null,
  primary key (accelerator_id,  lattice_id, node_id),
  foreign key (accelerator_id, lattice_id) references accelerator_lattices (accelerator_id, lattice_id),
  foreign key (accelerator_id, sequence_id) references accelerator_nodes (accelerator_id, node_id),
  foreign key (accelerator_id, node_id) references  accelerator_nodes (accelerator_id, node_id)
);
```

The *accelerator_id* and *lattice_id* fields specify the accelerator name and the version of the accelerator structure, respectively. The *node_id* column keeps identifiers of the accelerator nodes contained into the *sequence_id* structure. The *position* attribute is a longitudinal position of the node with respect to the beginning of the parent sequence.

2.3. Accelerator Node Bucket

Accelerator node bucket encapsulates the minimal set of attributes relevant to the single physical effect (e.g. magnetic field, offset), element feature (e.g. size, aperture), or special algorithm parameters. This structure addresses several tasks. First, it facilitates selection and classification of well-defined concrete data types. Second, it provides a consistent scalable approach for introducing and integrating new elements and element attributes. Finally, it offers a uniform algorithm-neutral element description that is open to diverse applications. Usually, each application requires an individual combination of element buckets. For example: a survey layout is based only on geometrical parameters; design programs use main element attributes; simulation codes includes field errors, misalignments, aperture, and other features that are critical for the particular scenario and accelerator. These applications can be configured in many different ways. In the Optics database, all these configurations are stored in the *accelerator_node_buckets* table:

```
create table accelerator_node_buckets (
  accelerator_id          varchar2(30),
  state_id                varchar2(30),
  node_id                 varchar2(30),
  bucket_type             varchar2(30),
  primary key (accelerator_id,  state_id, node_id, bucket_type),
  foreign key (accelerator_id, state_id) references accelerator_states (accelerator_id, state_id),
  foreign key (accelerator_id,  node_id) references accelerator_nodes (accelerator_id, node_id),
  foreign key (bucket_type) references accelerator_bucket_types (bucket_type),
);
```

The *accelerator_id* and *state_id* fields specify the accelerator name and the version of the accelerator node configuration, respectively. The *bucket_type* column keeps identifiers of buckets included in the *node_id* structure.
A predefined set of legal UAL bucket types is located in the *accelerator_bucket_types* table:

```
create table accelerator_bucket_types (
  bucket_type             varchar2(30),
  primary key (bucket_type)
);
```

Each bucket type is represented by the separate table. At this time, we consider buckets with bend attributes, magnet and accelerating fields, alignment errors, and aperture parameters:

```
create table bend_buckets (
  accelerator_id          varchar2(30),
  node_id                 varchar2(30),
  source_id               varchar2(30),
  hangle                  float(126),
  vangle                  float(126),
  primary key (accelerator_id, node_id, source_id),
  foreign key (accelerator_id,  node_id) references accelerator_nodes (accelerator_id, node_id),
  foreign key (accelerator_id, source_id) references accelerator_sources (accelerator_id, source_id)
);

create or replace knl_va_type as varray(10) of float(126);
/
create or replace ktl_va_type as varray(10) of float(126);
/

create table mfield_buckets (
  accelerator_id          varchar2(30),
  node_id                 varchar2(30),
  source_id               varchar2(30),
  knl                     knl_va_type,
  ktl                     ktl_va_type,
  primary key (accelerator_id, node_id, source_id),
  foreign key (accelerator_id,  node_id) references accelerator_nodes (accelerator_id, node_id),
  foreign key (accelerator_id, source_id) references accelerator_sources (accelerator_id, source_id)
```

```
);

create or replace volt_va_type as varray(10) of float(126);
/
create or replace lag_va_type as varray(10) of float(126);
/
create or replace harmon_va_type as varray(10) of float(126);
/

create table rffield_buckets (
  accelerator_id            varchar2(30),
  node_id                   varchar2(30),
  source_id                 varchar2(30),
  volt                      volt_va_type,
  lag                       lag_va_type,
  harmon                    harmon_va_type,
  primary key (accelerator_id, node_id, source_id),
  foreign key (accelerator_id,  node_id) references accelerator_nodes (accelerator_id, node_id),
  foreign key (accelerator_id, source_id) references accelerator_sources (accelerator_id, source_id)
);

create table alignment_buckets (
  accelerator_id            varchar2(30),
  node_id                   varchar2(30),
  source_id                 varchar2(30),
  x                         float(126),
  y                         float(126),
  z                         float(126),
  phi                       float(126),
  theta                     float(126),
  psi                       float(126),
  primary key (accelerator_id, node_id, source_id),
  foreign key (accelerator_id,  node_id) references accelerator_nodes (accelerator_id, node_id),
  foreign key (accelerator_id, source_id) references accelerator_sources (accelerator_id, source_id)
);

create table alignment_buckets (
  accelerator_id            varchar2(30),
  node_id                   varchar2(30),
  source_id                 varchar2(30),
  shape                     float(126),
  x                         float(126),
  y                         float(126),
  primary key (accelerator_id, node_id, source_id),
  foreign key (accelerator_id,  node_id) references accelerator_nodes (accelerator_id, node_id),
  foreign key (accelerator_id, source_id) references accelerator_sources (accelerator_id, source_id)
);
```

All these tables have a similar structure. The first two fields, *accelerator_id* and *node_id*, identify the accelerator node. The *source_id* field is associated with the origin of bucket attributes. Attribute values can be initialized from many different sources: physicist's assumptions and requirements, calculation, or other specialized databases (such as Magnet and Survey databases). The list of other columns is summarized in the table 1:

Table 1 : List of bucket tables.

Bucket type	Table	Columns	Description
bend	bend_buckets	hangle	horizontal bend angle
		vangle	vertical bend angle
mfield	mfield_buckets	knl	array of normal multipole components
		ktl	array of skew multipole components
rffield	rffield_buckets	volt	array of RF voltages
		lag	array of phase lags
		harmon	array of harmonic numbers
alignment	alignment_buckets	x	offset in the x-direction
		y	offset in the y-direction
		z	offset in the z-direction
		phi	rotation around the x-axis
		theta	rotation around the y-axis
		psi	rotation around the s-axis
aperture	aperture_buckets	shape	aperture shape
		x	horizontal half-aperture
		y	vertical half-aperture

3. Future Activities

The presented schema defines a frame of the Optics database. Currently, we are working on a mechanism for the initialization of its tables. Data can be inserted into the database in many different ways. In the past, each vendor offered the specialized user-friendly environment that facilitated access to the database records (e.g. Oracle Forms). The choice of the particular vendor or tool was very important and determined the structure of project applications. At this time, the Web technologies have changed the situation by defining several industrial standards for each layer of database-based multi-tier infrastructure. It shifts the emphasis in the software development process from the selection of the particular tool to the selection of the most appropriate and promising industrial standards. The essential part of the modern infrastructure is occupied by the Extensible Markup Language (XML) technology. XML has become very popular in many areas and is considered as a universal mechanism for the initialization and configuration of various software systems. In our environment, we plan to employ the XML software for integrating off-line and online simulation facilities and developing interfaces based on Accelerator Description Exchange Format (ADXF [5]) files (see Fig. 2). ADXF was developed to provide a complete and uniform description of accelerator data used in diverse beam dynamics programs. The structure of this description is mapped from the common UAL accelerator object model and theoretically consistent with other accelerator model representations. Our next application will integrate together three representations: Accelerator Model classes, Optics database, and ADXF files. It

allows us to test the compatibility of their structures and initialize the Optics database tables from the ADXF file.

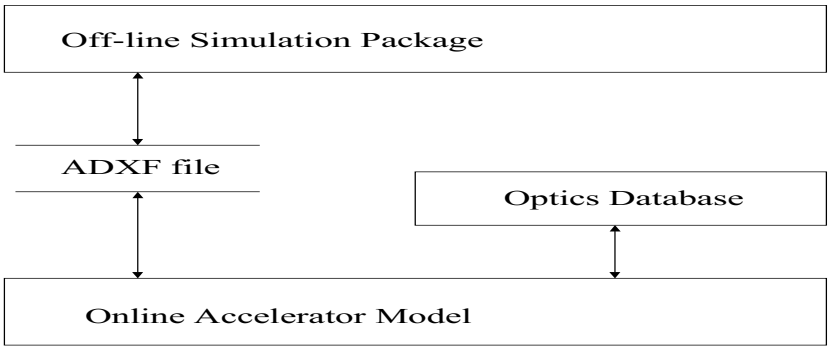

Figure 2 : ADXF-based interfaces.

4. Acknowledgments

The author would like to thank T. Nepsee, S.Sathe, J.Smith, and J.Wei for many valuable discussions.

References:

[1] F.C.Iselin, E.Keil, R.Talman. *Request for the Accelerator Description Standard*. 1998.
[2] S.Peggs et al. *LAMBDA Manual*. RHIC/AP/13, 1993.
[3] N.Malitsky and R.Talman. *Unified Accelerator Libraries*. CAP, 1996.
[4] M.Blaha and W.Premerlani. *Object-Oriented Modeling and Design for Database Applications*. Prentice Hall, 1998.
[5] N.Malitsky and R.Talman. *Accelerator Description Exchange Format*, ICAP, 1998.