RHIC/AP/33

November 30, 1994

A Generic Device Description Scheme Using a Relational Database

T. Satogata

I. Motivation and Problems	1
II. A Fundamentalist Approach to Entity-Relationship (ER) Diagrams	3
III. The Generic Device Description Tables	5
IV. Some Specific Examples	9
V. How to Implement Real Device Instances	12
VI. Implementation for the AGS to RHIC Transfer Line	14
VII. Concluding Remarks	15

I. Motivation and Problems

To efficiently install and control a large system such as an accelerator or transfer line, the relationships between its various elements must be defined in a clear and consistent manner. One particular problem for RHIC, both from an optics/electrical bus viewpoint and a controls viewpoint, has been the relationship between magnets and power supplies — which power supplies control which magnets? This is a trivial question to answer for correctors and trim magnets, but the wireup issue is nontrivial for complicated buswork such as that for the interaction region quadrupoles. This paper describes a scheme which handles arbitrary wireup problems with a relational database — this scheme is also shown to be extensible to a general description of design, including data flow in control applications as well as physical installation of complex buswork.

The quadrupole power supply busing for four of the RHIC interaction regions is shown in Figure 1, as taken directly from the RHIC design manual.[1] A natural way to view this or any other connection schematic is as a set of boxes (*devices*) with lines (also devices) drawn between them. Devices (magnets, power supplies and busing in this figure) have general attributes such as the name and type of device and the numbers of incoming and outgoing attachments for connections (*spigots*). Connections, the links between the spigots on devices, also have general attributes such as type (a hardware or software connection type). Devices and connections, and their connectivity relationships, can be generally

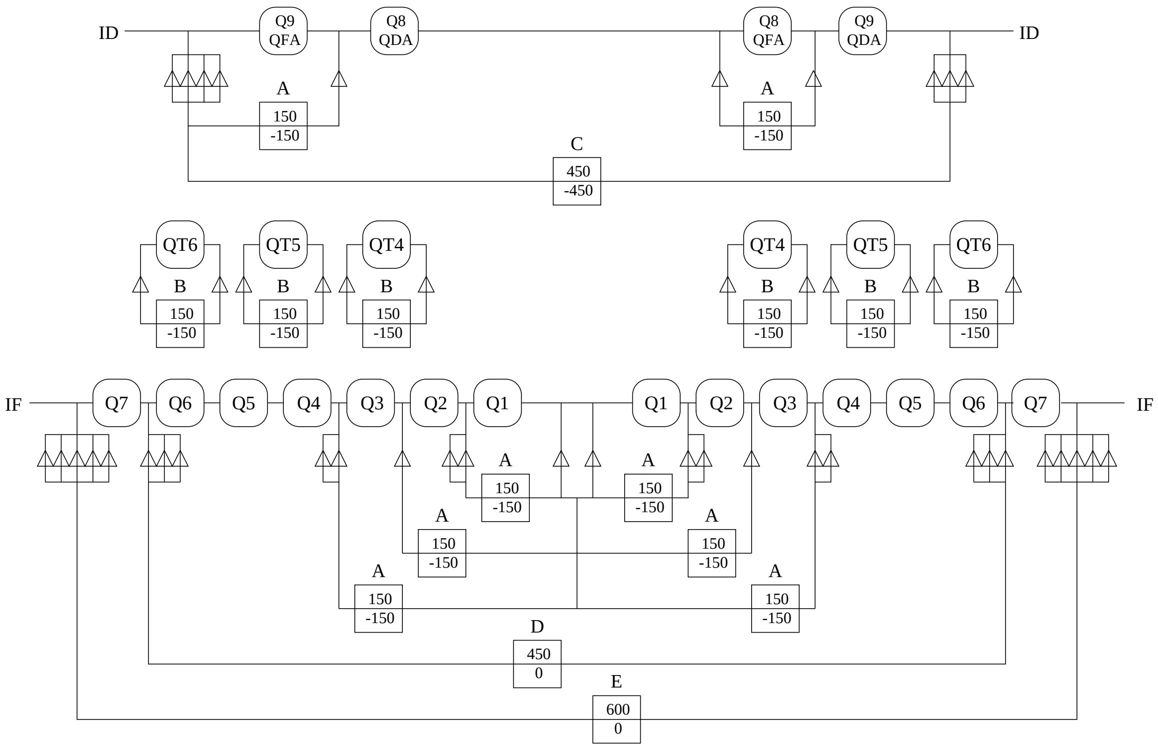

Figure 1: Power supply busing for quadrupoles in the RHIC interaction regions (IRs) at 2, 6, 8 and 12 o'clock. Triangles represent 150 amp cryogenic penetrations. This figure is from Bob Lambiase.

described by entries in a relational database.

The wiring diagram of IR quadrupoles in RHIC, like most of the wiring in most complex systems, is highly regular and duplicitous. There are six identical quad trims in Figure 1 (itself applicable to four of the six RHIC IRs), and this trim wiring scheme is duplicated hundreds of times throughout RHIC. A general hierarchical description of this diagram avoids the consistency issues that plague the update of many copies of this information.

To address these issues we have designed a database which can handle arbitrary wireup diagrams, such as that of Figure 1. This database is designed to be as flexible as possible, and includes generic templates for common connection schemes.

It must be stressed that these tables are engineered for one specific purpose — to describe connection and containment schemes for generic objects. They are *not* meant to provide a repository for information specific to physical instances of things, and their generality is lost if attributes are added for discrimination of physical instances. Other databases (such as inventory tables) should contain this information, along with references

Magnet Description

Slot SWN* `char[20]`	Serial Name `char[20]`	Type `char[10]`
...	...	...
uq5	ATRQSL008	quad
uq6	ATRQSS013	quad
uq7	ATRQSL007	quad
...	...	...

Power Supply Description

Supply SWN* `char[20]`	Supply Serial Name `char[20]`
...	...
psuq5	ATRPS032
psuq6	ATRPS033
psuq7	ATRPS034
...	...

Power Supply Wireup

Magnet Slot SWN `char[20]`	Power Supply SWN `char[20]`	Polarity `int`
...	...	...
uq5	psuq5	1
uq6	psuq6	1
uq7	psuq7	1
...	...	...

Table 1: Three example tables showing magnet busing. The acronym SWN stands for SiteWide Name and an asterisk indicates a primary key. Three rows are shown for each table.

into this database that show how these devices fit into the general wireup scheme. This is demonstrated by examples in later sections.

II. A Fundamentalist Approach to Entity-Relationship (ER) Diagrams

A database is comprised of tables, where each table consists of columns (and associated data types) into which data are placed. Data grouped by rows in a table are called table entries.

Three simple database tables are shown in Table 1. The Magnet Description table has three columns: the Magnet Slot SiteWide Name (SWN), which is a unique name for the *lattice position* of the magnet, the Magnet Serial Name, which is a unique identifier for the *physical magnet* which is installed in that slot, and the Magnet Type. The Power Supply Description table includes an SWN and Serial Name for power supplies. The Power Supply Wireup table associates entries in the previous two tables, providing design information on how magnets and power supplies are bused together.

There is sometimes a single column or group of columns in each table which is a "key", a unique identifier for any single table entry, or row. Some tables have no keys, while others may have several which key the same table in different ways. In the Magnet Description

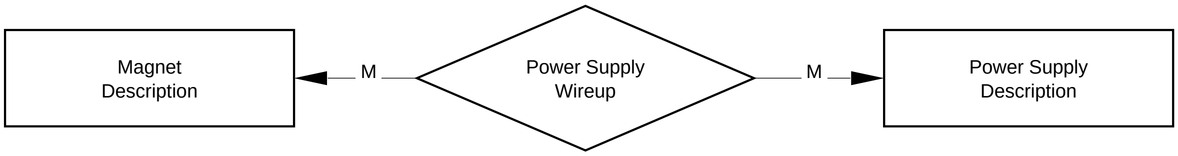

Figure 2: An entity-relationship (ER) diagram of the example tables from Table 1.

and Power Supply Description tables above, the SWN entries are declared as primary keys; during data entry any new entry in a table that duplicates a primary key is automatically rejected.

In the course of database design, tables naturally fall into two categories. One category is for entities or instances of things — magnets, magnet slots, power supplies, wires, cables, cards, people and so forth. Tables that describe these instances are called **entity tables**. The other type of table is a **relationship table**, which associates entities, the entries in entity tables.

Entity tables are almost always keyed; they also have other columns which contain descriptive attributes that all entries in the table may share. It is an important and difficult design decision to choose a reasonable level of abstraction for a problem such that the information in entity tables is neither highly duplicitous nor irrelevant. For example, power supplies and magnets share some attributes (color, weight, manufacturer, serial name) but not others (magnet type, magnet half-core serial numbers and power supply limits).

Tables may be related to one another in various ways (hence the term "relational"). A convenient way of diagraming the relational database references between entity and relationship tables is with an **entity-relationship (ER) diagram**.[2]

In an ER diagram, entities are represented by rectangles and relationships are represented by rhombi; there is usually a one-one correspondence between these symbols and actual database tables. Directional **arcs** are drawn between entities and relationships to indicate reference, or dependence — in implementation the table at the base of the arc (the table that symbolizes the relationship) contains a column with the same data type as the primary key of the table at the end of the arc. Interpretation of arcs is sometimes simplified by using verbs as labels, which allows one to "read along the arcs". Association qualifiers (such as "M" for many, "alw" for always, etc.) are also used as arc labels. An ER diagram of the tables in Table 1 is shown in Figure 2.

Tables which have many arcs pointing to them are "fundamental" tables; their entries are referenced, by primary key, in many other tables. Fundamental tables are the first tables filled during data entry. In the next section the DeviceType table is an example of such a table — it contains a list of all possible DeviceTypes for Device table entries. The structure of the tables constructed by the ER method should prevent the entry of a DeviceType in the Device table that is not in the DeviceType table to maintain referential integrity. On the other hand tables which have many arrows pointing away from them are generally relationships between the various tables to which they point.

The program **erdraw** [2], developed at LBL, was used to implement these tables using ER methods. This program allows graphical editing of ER diagrams, including table attributes and fairly sophisticated delete and update rules. Most importantly, erdraw also produces SQL for table creation, table keying, referential integrity and metatables (tables containing descriptions of these tables) that can be read by most database SQL interpreters.

III. The Generic Device Description Tables

Figure 3 shows the ER layout of the generic device description (GDD) tables. The six lower tables compose generic instances of devices (including templates of wiring schemes) while the three top tables represent actual instances of devices that fit into these templates. This section describes the generic tables in more detail.

III.1: The fundamental entity tables

Since we seek to represent connection diagrams similar to Figure 1, it is reasonable to start with basic entities which represent objects on this diagram. Boxes with external connection points are called Devices and the connection points on devices are called Spigots:

Device: a hierarchical object which contains zero-many other devices and which is contained in zero-many other devices. Devices each also have zero-many "spigots", and have primary-key Name and Comment and Type attributes.

Spigot: an external connection point on a device. Spigots only have directionality within the context on a particular device. The only attribute of Spigots is a primary-key Name.

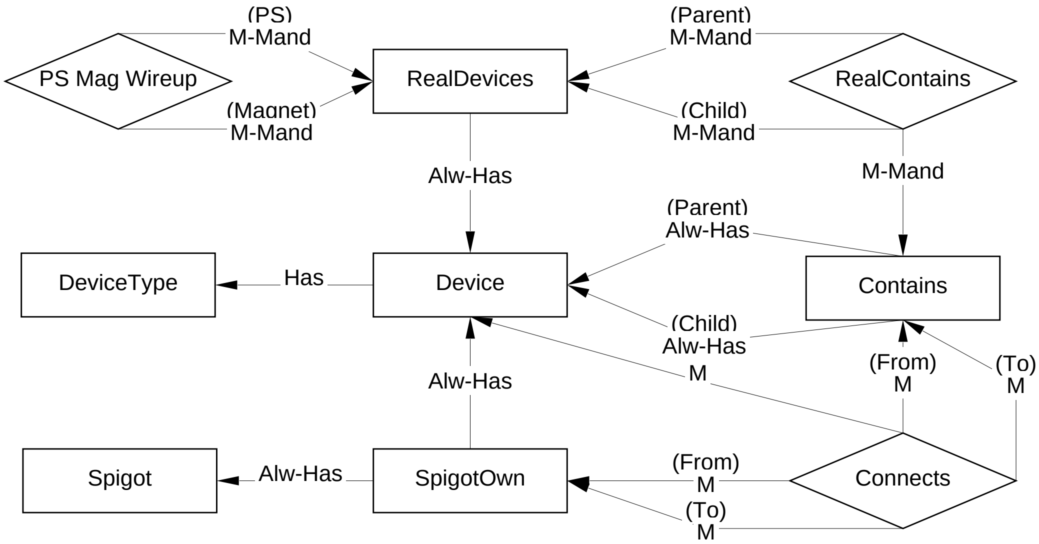

Figure 3: The generic device description (GDD) tables in entity-relationship (ER) format. The top three tables are not part of this description, but show the relationship of physical instance tables to GDD tables.

Table Name	Attribute	Type
DeviceType	**Type***	char[20]
Device	**Name***	char[20]
	Purpose	char[60]
Spigot	**Name***	char[20]
SpigotOwn	**Name***	char[20]
	Direction	int
Contains	**Name***	char[20]
Connects		
RealDevices	**Name***	char[20]
RealContains		
PS_Mag_Wireup	**Polarity**	int

Table 2: Attributes of the GDD tables. The order listed is the order in which tables should be filled for referential integrity. A star indicates a primary key; attributes in boldface are mandatory for each table entry. Table columns corresponding to arcs in Figure 3 are not included in this list.

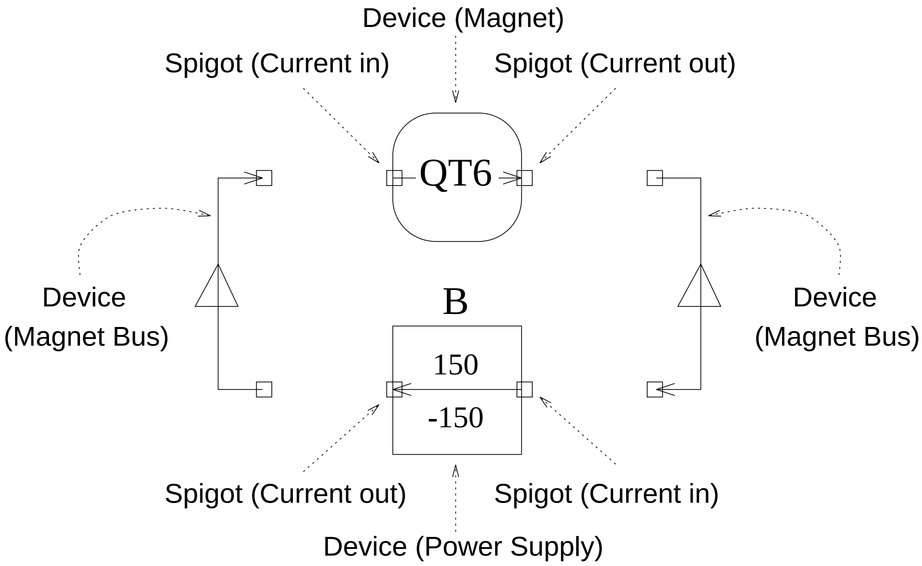

Figure 4: A closeup of a corrector magnet bus. Arrows show nominal spigot polarity, and each device has two spigots. The entirety of this diagram is a single generic device template called "1PS-1Mag".

The majority of arcs in Figure 3 (as well as most of the tables) are concerned with many-many relationships between these fundamental entities. A closeup of a quad corrector from Figure 1 is shown in Figure 4 to clarify this terminology.

There is one more fundamental entity, **DeviceType**, which lists the acceptable entries in the Type attribute of each device. This constraint is shown by the "has" arc between these two tables. Note that this is not an "alw-has" (always-has) arc, so devices may exist with `NULL` Type or a Type listed in the DeviceType table, but none other.

III.2: Many-many relationship tables

There are two relationships between the fundamental entities Device and Spigot that are evident from the example figure: each device "owns" zero-many spigots and each device contains zero-many devices. Note also that the converse is also true — each spigot is owned by zero-many devices and each device is contained within zero-many devices. More succinctly, there are many-many relationships between the Spigot and Device tables (ownership) and between the Device table and itself (containment). These relationships are represented by the **SpigotOwn** and **Contains** tables in Figure 3.

There are interesting things to note about these two tables. First, even though they are many-many relationships, they are entities themselves. For connectivity it is important to be able to distinguish between the same types of spigot on a particular device, as well as the same types of device contained within a larger composite device. Each of these tables must therefore have its own primary key; for lack of better nomenclature this is a Name.

With SpigotOwn the context is also established for directionality. It is clear that on some devices a current spigot is incoming while on others it is outgoing. It is also clear that on devices where this distinction is not immediately apparent (e.g. magnets, ground buses), there are still assumptions of polarity that warrant this distinction in all spigot-device associations. The SpigotOwn entity has a Direction attribute (± 1 or 0, indicating polarity or lack thereof).

III.3: Connections

Circuits are created by attaching spigots together; this is akin to physically performing a connection such as attaching a cable to a socket. Using the device/spigot terminology, a connection is a relationship between a spigot on a contained device (a SpigotOwn entry) and another spigot on another contained device. Both the SpigotOwn entry (specifying a device and spigot on that device) and a Contains entry (specifying which instance of a device within a composite device) are needed for each end of the connection. Different connections may share the same SpigotOwn or Contains references, but not both.

Connections are implemented as the paired many-many relationship **Connects** in Figure 3. Here there are two many-many relationships, the *To* pairing and the *From* pairing, which are associated within a composite device. It is also possible (even preferable) to create a Sybase *view* which lists all contained devices and their spigot lists, and associate entries in this view within the Connects relationship — however the ER methodology does not appear to implement this approach.

For a circuit tracing program to work with this data, all circuits must be closed. Internal connections are supported by this framework — if the Contains entry is absent in a Connects table entry, the Spigot listed is presumed to be a spigot on the internal side of the device containing the connection. This will be made clearer in the next section by example.

Every entry in the Connects table can now be interpreted, "Within a certain composite Device, there is a connection from SpigotOwn (an instance of a spigot on a device) on Contains (an instance of a device in the composite device) to another SpigotOwn on Contains". Completely general wireup and connection schemes are supported by this design.

IV. Some Specific Examples

Here we consider two examples. The simple case of corrector and trim magnets is meant to clarify the ER design of the GDD tables. Second, we consider the more complex scenario of IR quad busing as depicted in Figure 1; this example also depicts how complex device hierarchies are implemented.

IV.1: Corrector and Trim Magnets

Consider the simple case of corrector and trim magnet busing, Figure 4. This composite device, generically called "1PS-1Mag" here, is duplicated six times in *each* IR quad bus design (Figure 1), as well as hundreds of times for correctors and trims throughout ATR and RHIC — Figure 4 thus serves as a *template* for this wireup scheme. Table 3 shows the entries in the GDD tables for this diagram.

There are three DeviceTypes in Figure 4, a Power Supply, a Magnet and a Bus. The composite generic device 1PS-1Mag representing the entirety of the figure has a DeviceType "Template". There are three other Devices, a one-tap magnet, a one-tap power supply and a magnet bus; the only Spigot necessary is a "Current".

The magnet, power supply and magnet bus each have two Current spigots, in and out. The template here does not have any external currents or control points and thus has no spigots. The Contains table entries are self-explanatory, but note that there are two different instances of the Magnet Bus device in 1PS-1Mag.

The Connects table first lists the four connections that are obvious, those that attach together the four Devices that make up 1PS-1Mag. The last three connections are internal device connections, and signify that current that comes into the "in" spigots goes out the "out" spigots. This may seem trivial, but when more realistic descriptions are included (external control points for power supplies, voltage and thermal taps on a magnet, etc.) these connections are necessary for a circuit-tracing program to follow current paths within these devices.

This is still a generic representation; 1PS-1Mag is a simple template for wiring which holds for all trim magnets and power supplies of this type. Section V explains how physical instances of magnets and power supplies relate to the GDD tables.

Another way to implement a trim magnet template is to ignore the buses as uninteresting and simply to join the input and output currents of the magnet and power supply together.

DeviceType Table

Type char[20]
Magnet
Power Supply
Bus
Template

Device Table

Name char[20]	DeviceType char[20]	Purpose char[60]
Magnet1Tap	Magnet	One-tap magnet
Power Supply	Power Supply	One-tap power supply
Magnet Bus	Bus	
1PS-1Mag	Template	Device Template...

Spigot Table

Name char[20]
Current

SpigotOwn Table

Name char[20]	DeviceName char[20]	SpigotName char[20]	Direction int
Mag1Iin	Magnet1Tap	Current	1
Mag1Iout	Magnet1Tap	Current	-1
PSIin	Power Supply	Current	1
PSIout	Power Supply	Current	-1
BusIin	Magnet Bus	Current	1
BusIout	Magnet Bus	Current	-1

Contains Table

Name char[20]	Parent char[20]	Child char[20]
1PS1Mag-Mag	1PS-1Mag	Magnet1Tap
1PS1Mag-PS	1PS-1Mag	Power Supply
1PS1Mag-Bus1	1PS-1Mag	Magnet Bus
1PS1Mag-Bus2	1PS-1Mag	Magnet Bus

Connects Table

Device char[20]	SpigotOwn From char[20]	Contains From char[20]	SpigotOwn To char[20]	Contains To char[20]
1PS-1Mag	Mag1Iout	1PS1Mag-Mag	BusIin	1PS1Mag-Bus1
1PS-1Mag	BusIout	1PS1Mag-Bus1	PSIin	1PS1Mag-PS
1PS-1Mag	PSIout	1PS1Mag-PS	BusIin	1PS1Mag-Bus2
1PS-1Mag	BusIout	1PS1Mag-Bus2	Mag1Iin	1PS1Mag-Mag
Magnet1Tap	Mag1Iin		Mag1Iout	
Power Supply	PSIin		PSIout	
Magnet Bus	BusIin		BusIout	

Table 3: GDD table entries for the 1PS-1Mag template.

This is feasible, and works if that association is all that is needed, but it ignores the fact that the busing is real and has properties of interest itself (such as penetration limits).

IV.2: Complex Buswork — RHIC IR Quads

It is natural to view the RHIC IR quadrupole busing of Figure 1 as a single template that is instantiated four times, once for each of the 2, 6, 8 and 12 o'clock IRs. This template (unlike the 1PS-1Mag template) has four external spigots for the main quadrupole buses. It also quite naturally breaks down into eight smaller templates, six that are closed 1PS-1Mag instances and two that are the main quad focusing and defocusing buses.

The most worrisome aspect of the RHIC IR quad busing is the many-many relationship

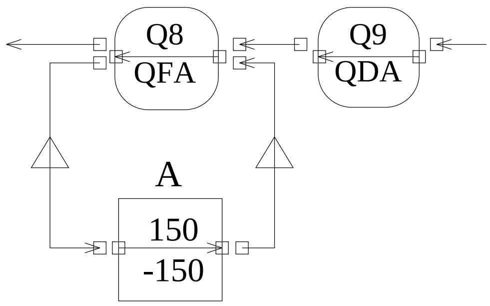

Figure 5: A closeup from Figure 1, showing how current tees are implemented.

between power supplies and magnets — most of the quadrupoles are not on a single bus dominated by a single power supply. A closeup is shown in Figure 5, showing how this can be implemented in the GDD tables; basically parallel buses are connected to the same input and output spigots on magnet Q8.

A bigger problem arises on the focusing bus, where there are both tees and four-bus junctions. A simple solution is to break the bus between the Q1 magnets into two sections, and to use the junction between them for feed-ins and returns for the six A-type power supplies. The directionality labels on spigots indicate *polarity* of flow, and do not indicate that this is the only direction that current can flow. This being the case, return paths for power supply current are not always those listed — a circuit tracing program using Kirchoff's laws should use these directions only as polarity references.

To describe the entirety of Figure 1, eight templates are needed — one for the focusing bus, one for the defocusing bus and six for the quad trim packages. Each quad trim package uses a single magnet and a single power supply, and implies two internal buses. The defocusing bus template requires four magnets and three power supplies, and contains eleven bus connections (two for each power supply and five for the main bus sections). The focusing bus is most complicated; it requires fourteen magnets, eight power supplies and thirty-two bus connections. Similar diagrams and table entries can be created for the other main buses of RHIC.

Once the templates for the IR and arc quad buses are created, a template for the entire RHIC main quad bus can be created. This main quad bus layout is depicted in Figure 2-3 of the Design Manual Magnet Electrical System section. [1] A set of roughly a dozen main templates are sufficient to cover the entire main bus system (dipole and quadrupole systems) of RHIC.

V. How to Implement Real Device Instances

Figure 6 shows the references into the GDD required to resolve a wireup scheme for a real trim magnet, in this case `yo4-tq4`, the outer Q4 trim in the yellow ring.

Figure 3 showed three tables not in the GDD. **RealDevices** entries are actual devices as referenced by their SWNs or, in the case of Templates, some other unique identifier such as `yo4-tq4-tpl` from Figure 6. **RealContains** associates real devices within a real template, and references the GDD Contains table to discriminate separate instances of the same device type in a template. The **PS_Mag_Wireup** table associates magnets and power supplies, with polarities — this table is automatically filled by a wireup application that uses the GDD tables.

To enter another real trim instance, first the power supply and trim magnet should be entered as real devices in the RealDevices table. The template that represents the "trim package" must also be entered, and these should all refer to appropriate generic descriptions in the GDD Device table. (See bold arrows in Figure 6.) The RealContains table should then be filled with a pair of entries denoting where in the template the magnet and power supply should go, again with reference into the GDD Contains table.

There are many "generic implications" within a template. A real trim, generically wired and with an "uninteresting" power supply, can be entered as a real magnet and a 1PS-1Mag template instance, with all other elements implied within the template. Above, even the instances of the magnet busing are implied, even though they should be expressed as real instances of busing (with penetration limits, etc) in some other database. **Details of individual elements belong elsewhere** — it is the associations between generic instances that are represented by the GDD tables.

A pair of tables similar to RealDevices and RealContains is sufficient to reference the GDD structure from any database listing physical instances of devices. Examples that are currently under construction include ATR and RHIC magnet busing (physical power supplies and magnets), ATR instrumentation (magnet coil taps, BPMs, etc.) and an instrumentation group cable database.

The need to create real instances of all templates is an apparent inconvenience of the GDD scheme. However templates provide natural encapsulations of connection schemes, just as slots in the accelerator optics database provide natural encapsulations of removable beamline equipment. Templates are an organizational tool, and should not be avoided simply to "streamline" database contents. Also, devices may themselves be templates, and

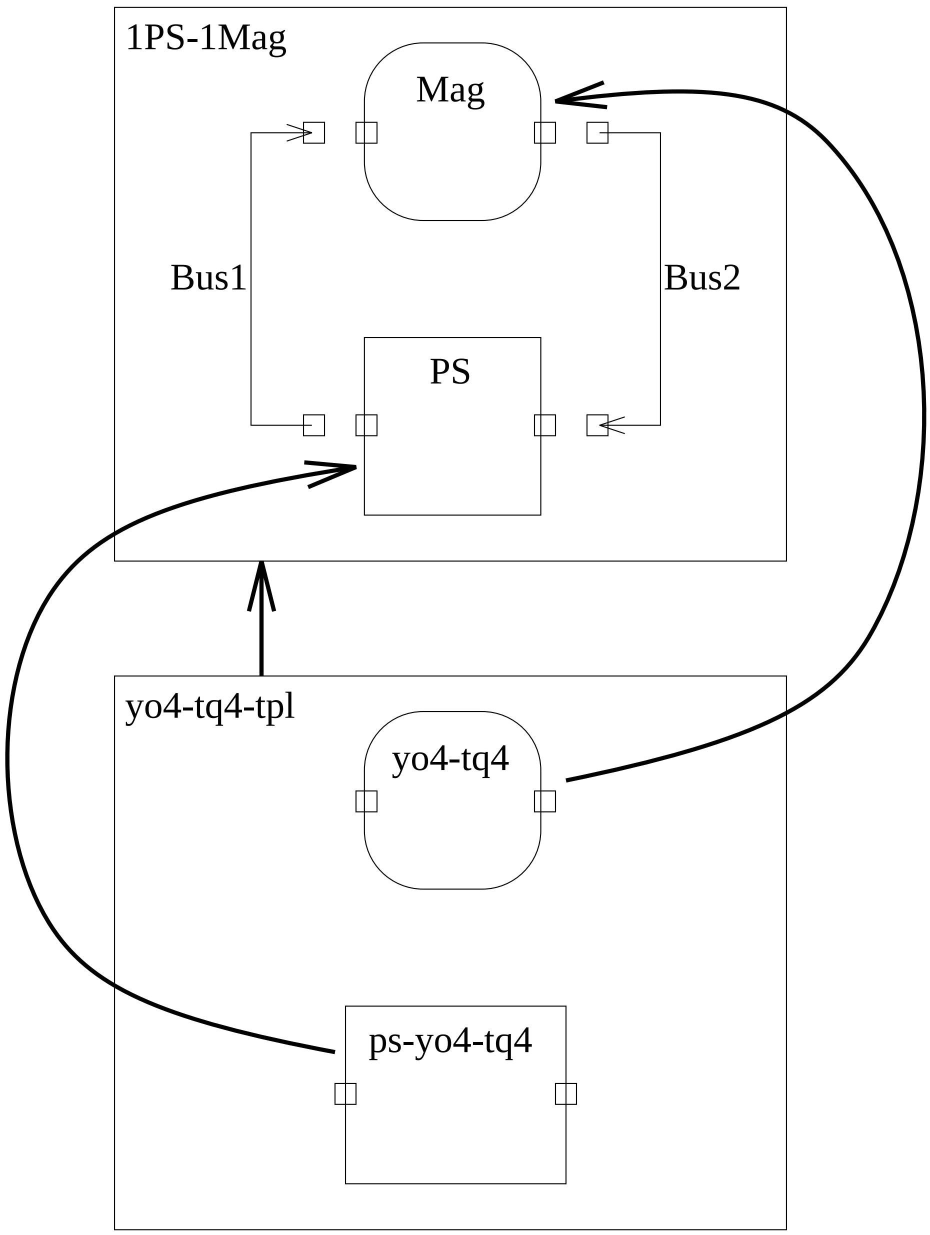

Figure 6: GDD references for an actual instance of a generic trim. Note the template `yo4-tq4-tpl`, which ties together the magnet and power supply associations.

imply other generic structures beneath — this actually means that the number of physical entries of things is *minimized* in this scheme because duplicate implications can reside within the GDD.

VI. Implementation for the AGS to RHIC Transfer Line

To test the viability of this scheme, the magnet busing for all magnets in the ATR was implemented in Sybase tables produced by erdraw from the ER diagram of Figure 3. Table creation took less than an hour, and data entry for the 147 magnets in these transfer lines took less than a day.

The vast majority of ATR magnets are individually powered; only the dipoles and lambertsons are on buses which require any templates other than the 1PS-1Mag template described above. The dipoles on these buses are also 4-tap magnets, with two internal coils and two internal return buses. To discriminate between these internal "magnet coil" and "return bus" devices were used. A "current source" internal device was also used to connect the input and output current spigots within power supplies. The 4-tap dipoles, except for xd31 and yd31, were jumpered internally to look like 2-tap dipoles with a single magnet coil and a single return bus.

The xd31 and yd31 dipoles, and the lambertsons, also have trim power supplies jumpered across their main buses. This situation was handled the same way as multiple magnet buses in RHIC IR quad wireup, by simply connecting these power supply buses to the dipole and lambertson in parallel with the main arc buses.

A program, **wireup**, goes through the list of all magnet devices in the RealDevice table, and finds the set of top-level templates which contain all these magnets. For each of these templates wireup collects information on all internal connections, including recursively traversing other templates, and assembles a list of devices and connections. It singles out the "current source" devices from this list, and traces all circuits that originate at this current source. Both closed circuits (and the magnet coils that reside on them) and open circuits (indicating there is something wrong) are reported. Wireup is also smart enough to avoid infinite recursion during template expansion and circuit tracing.

Wireup consists of a 450-line C library of generic routines for traversing the GDD table structure (basically a software implementation of views), and approximately 700 lines of highly recursive C code. When run on the ATR table entries, it produced correct magnet and power supply associations for all magnets in a few seconds of run time. Scaling to

RHIC gives an estimated wireup time of a few minutes.

VII. Concluding Remarks

This paper concentrates on describing the GDD and its applicability to magnet busing and wiring schemes. There is, however, nothing specific to this application within the GDD tables. Connectivity diagrams for control flow and for hierarchical relationships between objects within a control system, such as RHIC ADOs [3] and high-level controls hierarchies [4], are quite easily represented within the GDD design. Additional table attributes (such as a ConnectionType in the Connects relationship) can also lead to more application-specific GDD tables.

*

References

[1] RHIC Design Manual, H. Hahn et. al., editors, 1994.

[2] V.M. Markowitz and A. Shoshani, "An Overview of the Lawrence Berkeley Laboratory Extended Entity-Relationship Database Tools", LBL Technical Report LBL-34932, 1993.

[3] Accelerator Device Object (ADO) Specifications, RHIC Controls group, 1994.

[4] ATR Commissioning Software Task Force Report, C. Saltmarsh and G. Trahern, editors, 1994.