A POTENTIAL NEW NUCLEAR EXPLOSION MONITORING SCHEMA

Christopher J. Young and Sanford Ballard

Sandia National Laboratories

Sponsored by National Nuclear Security Administration
Office of Nonproliferation Research and Engineering
Office of Defense Nuclear Nonproliferation

Contract No. DE-AC-04-94-AL85000

ABSTRACT

Relational database schemas are widely used in nuclear explosion monitoring applications (Carr, 2004) and they work well for many aspects of monitoring, but have some limitations that can only be addressed through fundamental redesign. In this paper we present a new seismic schema that captures all of the information in the current core schema, but is better normalized, more extensible, and provides a better basis for next generation seismic monitoring research and system development.

Past modifications of the core schema have typically been minor, avoiding significant changes to the existing tables because software that interfaces with them may have to be modified also. Introducing custom tables is equally problematic and similarly avoided. In the short term, keeping the original tables makes sense because software modifications can be expensive. For conducting new seismic monitoring research and developing next generation monitoring software systems, however, keeping the current table structure can become a significant impediment.

Using the Object Role Modeling (ORM) component of Visio Enterprise Architect Edition, and starting from the basic object fact relationships, we have developed a data model and resultant schema that supports many-to-many event-to-origin relationships, completely captures complex model prediction information, properly records array membership, and properly normalizes network magnitude storage, to name just a few features. By using the Visio tool, we are assured that our resultant schema is highly normalized, that all tables are linked with settable primary, foreign and unique keys, and that important constraints are enforced within the database via stored procedures. The resulting schema favors efficiency and maintainability, but ease of interaction with the information in the database can still be achieved through the use of views. We have made great efforts to make our design easily extensible without modifying the schema, but should modifications be required they can be made easily through the ORM model. Because we include the ORM as part of the schema design, there is no confusion in our schema about why a table or column is included: every feature of the schema can be related directly back to a simple fact statement that can be validated by any seismic monitoring domain expert. To verify the accuracy of our new schema, we have developed sql scripts to set up the old schema tables as views of the new tables.

Our initial development has focused on seismic monitoring, but we have made efforts to keep the schema as generic as possible so that it can be used to capture similar types of information for other ground-based monitoring technologies (e.g., hydroacoustic and infrasound). Should our proposed schema be chosen for use with real monitoring systems, large amounts of old-format data would need to be converted to the new schema. We show in this paper how this would be done with a simple custom utility.

OBJECTIVE

The basic table structure of the seismic schema currently used for nuclear explosion monitoring was developed more than 15 years ago (Anderson et al., 1990) and has changed very little since then (Carr, 2004). We have had many years of experience with this schema, both using it to hold data as well as designing increasingly sophisticated software to interface with it. For most basic purposes, the schema works well, but our work with developing the NNSA Knowledge Base has produced a list of problems that is both too long and of too fundamental a nature to address with simple fixes to the current tables. In particular, we think that the current design has become an impediment to modern software design practices (e.g., object oriented coding) that rely on a well-designed schema architecture with settable primary and foreign keys. We believe that a fundamental redesign of the core monitoring schema is in order and present our first attempt at that in this paper. Our ultimate goal is to develop a schema that efficiently and accurately captures as much of the monitoring domain as possible and that will extend well with data sets and techniques yet to be developed. For our design, we used the ORM component of Visio Enterprise Architect Edition, and started from the basic object fact relationships (e.g., "an origin has zero or more arrivals"). By using the Visio tool, we are assured that our resultant schema is highly normalized, that all tables have settable primary, foreign and unique keys, and that important constraints are enforced within the database via stored procedures.

RESEARCH ACCOMPLISHED

The core tables are often divided into "Primary" and "Lookup," and we will do the same here. Lookup tables are those that contain static information (i.e., information related to networks, stations, etc.). Primary tables are those that contain dynamic information associated with a given candidate explosion event: location, time, magnitude, which stations recorded it, various amplitude measurements, etc.

To help clarify the discussion below, we introduce the following conventions: table names are in **boldface**, and **c**olumn names are *italicized*.

Primary Tables

Figure 1 shows the Entity Relationship Diagram for the primary tables, with only the key columns (primary = PK, foreign = FK, and unique = UK) shown to simplify. To make our schema more easily understood by those familiar with the current schema, where tables are analogous to tables in the existing schema we have used the same name. Note, however, that none of the tables are exactly the same and some differ significantly, as we discuss below.

Events

An event can be thought of as the highest level object captured by the primary tables (i.e., every other object can ultimately be tied to an event). In the current schema, the **Event** table groups together one or more origins (hypothetical locations), one of which is the preferred origin. However, no origin can be part of more than one event; that is, in the current schema, event-to-origin is one-to-many. This is problematic for a data archive like the NNSA Knowledge Base where events (origin groupings) may have been put together by more than one author. The current design is limiting in another way as well. When multiple origins are relocated simultaneously using location algorithms such as Joint Hypocenter Determination, origin groupings need to be defined. For these reasons, we choose to instead model the event-to-origin relationship as many-to-many, which introduces an additional association table (**EvorAssoc**) that has foreign keys pointing to each of the tables it links. The many-to-many relationship also implies that the **Event** primary key *evid* is no longer needed as a column in the new **Origin** table. With our new association table, an origin can be linked to an arbitrary number of events.

Origins

We have made numerous changes to the **Origin** table for our new schema. Related to the location information represented by an origin, we introduce 3 new tables designed to store specialized information. **AzGap** stores information about station azimuthal coverage, **ModSource** stores information about the source of models used to predict observations, and **RefLocation** stores information about reference locations that an origin may be associated

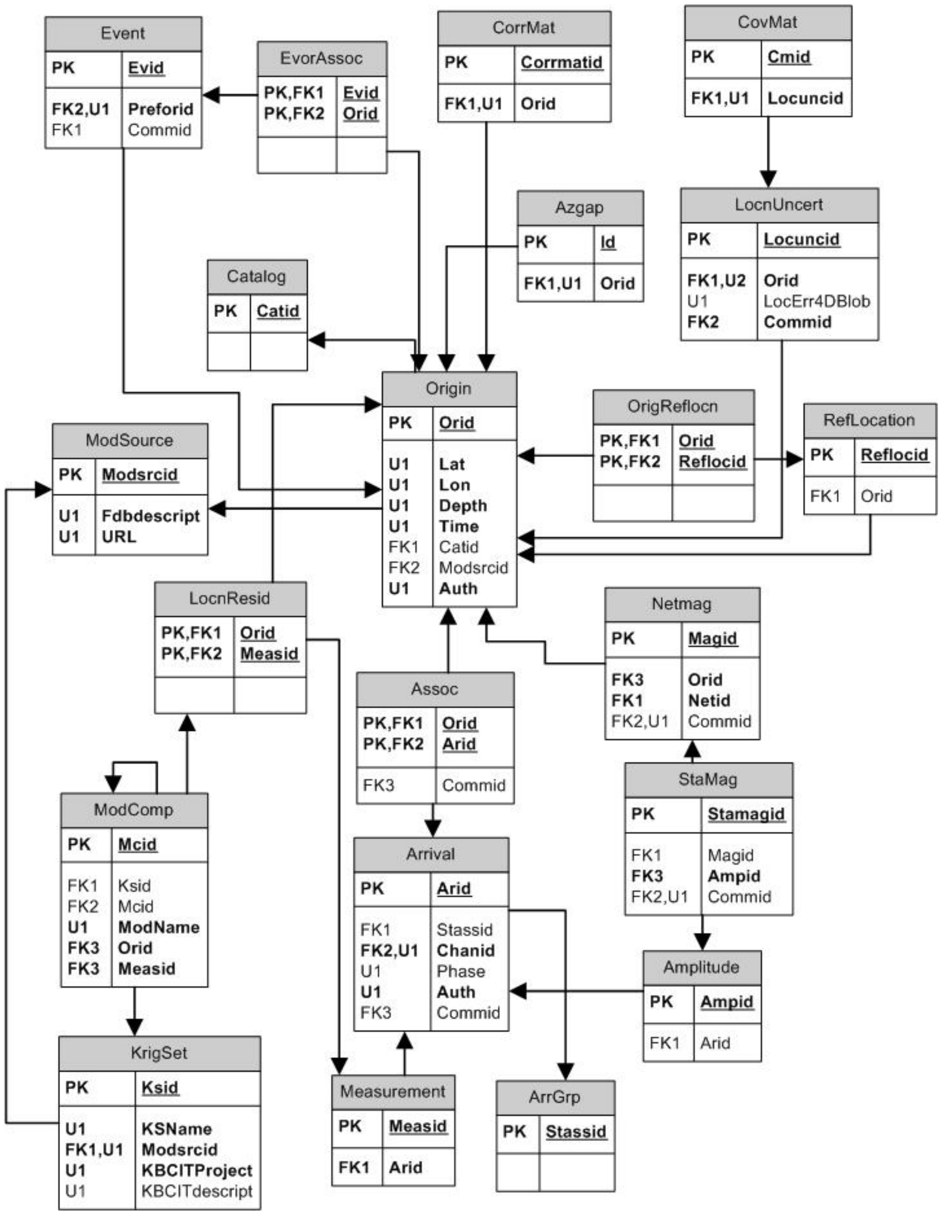

Figure 1. Entity Relationship Diagram (ERD) for the primary tables.

with (e.g., test sites). The **AzGap** ties one-to-one with a specific origin, so this table has *orid* as a foreign key pointing to the **Origin** table. The same **ModSource** information may be used for more than one origin, so this is a one-to-many relationship and hence *modsrcid* (the primary key for **ModSource**) is a foreign key column in the **Origin** table. An origin can have more than one reference location and vice versa, so this is a many-to-many relationship and an association table (**OrigReflocn**) is introduced in the new schema to link **Origin** and **RefLocn**.

Frequently, origins come from specific catalogs, so we have introduced a new **Catalog** table to capture information about the catalogs, including *name*, *description*, and *rank* (*rank* is used to determine which origin in an event is the preferred one). Because **Catalog** to **Origin** is a one-to-many relationship, *catid* (the **Catalog** primary key) is added as a foreign key in the new **Origin** table.

For magnitude information, the changes are all deletions. The current **Origin** table has *mb*, *mbid*, *ms*, *msid*, *ml*, and *mlid*. These ids are all foreign keys to *magid* in the current **Netmag** table, but this is unnecessary because the

Netmag to **Origin** relationship in the current schema is many-to-one, implying that a given origin could have any number of network magnitudes but each network magnitude is related to only one origin (thus *orid* is a foreign key column in **Netmag**). A simple query of **Netmag** by *orid* and *magtype* can fetch mb, ms, or ml values, so these do not need to be stored in **Origin**. Thus, 3 foreign keys in the current **Origin** that point to specific types of magnitudes (*mbid*, *msid*, *mlid*) in **Netmag** are redundant, as are the values associated with them (*mb*, *ms*, *ml*), so we remove all of these in the new schema. If there are multiple network magnitudes of the same type for the same origin, then it might be desirable to designate one of these as the preferred one. To implement this, we have added a new column "*is preferred*" to **Netmag**.

Arrivals

An arrival, also sometimes referred to as a "pick" or "detection," is the fundamental, station/phase-specific observation. Many types of measurements can be made for a given arrival, and one of the most fundamental problems we have found with the current schema is that the information for arrivals is very poorly normalized. An arrival may have travel time, azimuth, and/or slowness observation types measured for it, and the current **Arrival** table has columns to capture information for all three. This is inefficient in that it is very common that not all three types will be measured, so several columns are typically nulled. Further, for each observation type, the basic data to be stored (quantity and uncertainty) are the same, so the columns are highly redundant. Finally, we observe that if another fundamental observation type were deemed to be useful to measure for arrivals, additional columns would have to be added to the **Arrival** table.

We address all of these problems in our new schema by removing the measurement information from **Arrival** and putting it into its own table called **Measurement**. Each measurement has an *obstype*, a *quantity*, an *uncertainty*, and is tied to **Arrival** by the foreign key *arid*. Rows are added to this table only if a particular observation type has been made for an arrival, and there is no limit on how many could be added for a given arrival. In the future, if some new type of observation relevant to an arrival is deemed to be important, the information can be added to the existing tables without having to change the basic structure of any tables.

A further arrival normalization problem occurs with the many amplitude measurements that may be made for a given arrival (peak-to-trough, zero-to-peak, RMS, etc.). The current **Arrival** table has single columns for *amp* and *per*, so these are quickly used. This problem has long been recognized and extension tables have been introduced to capture multiple amplitude measurements per arrival. The USNDC P2B2 schema (Oo et al*.*, 2003) introduces an **Amplitude** table that we have adapted almost exactly for our new schema. This is a generic table with *arid* as a foreign key, so an arbitrary number of amplitude measurements can be made for any arrival. With this table as part of the new schema, we drop the old amplitude support columns in the current **Arrival** table (*amp*, *per*, *SNR*).

Arrivals Linked to Origins

We believe the association of arrival information with origins is also fundamentally flawed in the current schema. Currently, the **Arrival** table and **Origin** tables are linked with a many-to-many relationship via the **Assoc** table. On a high level, that is correct, but if we examine the relationship in more detail, problems emerge.

First, an arrival can be associated with an origin in two ways, and this is not well-modeled. An arrival can simply be tied to an origin in a general sense (it corresponds to a seismic wave caused by the origin), or it can be part of the process that was used to locate the origin. The latter is the reason for the *timdef*, *azdef*, *slodef* columns in the current **Assoc** table. The problem with the current schema is that these two types of information are both put in the same table (**Assoc**). Thus, **Assoc** has columns that are often nulled if an arrival is not used for location at all or is used only for travel time, azimuth, or slowness. Further, as was discussed for the current **Arrival** table, the information recorded in **Assoc** for each type of observation (defining, residual value, and weight) is the same, so the columns are redundant. Finally, though the **Assoc** table might have all three current observation types as defining, and hence using a model to drive location, there is only one column (**vmodel**) to capture the model information, which could easily be different for each observation type. A further problem with the current **Assoc** table is that there is only one *wgt* column when in fact separate weight columns are needed for each observation type. Omission of the weight information for azimuth and slowness observations in the **Assoc** table makes it impossible to recompute the sum squared weighted residuals of the observations associated with an origin if there are any defining azimuth or slowness observations. This is a major flaw since the sum squared weighted residuals are the quantity minimized by the location algorithm that produced the origin to which the assoc entry is linked.

To fix these problems, we made several significant changes. First, we split the information in the current **Assoc** table into two new tables: **Assoc** and **LocnResid**. The new **Assoc** table is basically a subset of the old **Assoc** table, with all the columns relating to locating the origin removed. Retained are columns to hold basic information such as station-to-event distance and azimuth, etc. The removed information is put into the **LocnResid** table, which is a many-to-many link between **Origin** and **Measurement**. This makes sense because the location information is all related to measurements (travel time, azimuth, and slowness) made from the arrival, not the arrival itself. The **LocnResid** table is fully generic, suited to holding information for travel time, azimuth, slowness, or any new observation types that might be introduced in the future.

LocnResid has separate rows for each observation type, so the problem with being limited to a single weight entry and model designation for all observation types is eliminated. Our new schema goes further than this, however, and is able to fully capture the complexity of location models. For many years, the models used for location have actually been composed of a base model plus a set of corrections (elevation, ellipticity, path correction, etc.), so specifying a single model has long proven problematic. Further, in the latest models emerging from location calibration research, the single resultant models developed for a given station/phase are actually composites of several region-specific models blended together. To capture all of this complexity requires a heavily genericized set of tables that can handle arbitrary hierarchies of complexity and we think we have achieved this in our new schema. We capture any of the various model components (base model total or by region, various corrections total or by region) in the new **ModComp** table. **ModComp** has a foreign key pointing to **LocnResid**, so it is easy to find the model components that go with any measurement-origin combination. The hierarchy of model components (e.g., the total base model may come from blending base models from two regions) is captured by endowing the **ModComp** table with a foreign key pointing to itself. To find out if a model component has subcomponents (children), all that is necessary is to query for **ModComp** rows that point to that row. This hierarchy can be extended to as many levels as desired, though we have had no reason to go beyond two (total and regional subcomponents).

To capture the information about the actual source of the models used for a given location (i.e.. file name and location, brief description), we have introduced a new table **ModSource**. Based on our knowledge of the current location models being developed and those anticipated in the future, we expect that all of the model information used to locate an event will come from the same source, so we tie this table to **Origin** rather than **ModComp**. The relationship between **ModSource** and **Origin** is one-to-many, so this introduces the primary key from **ModSource** (*modsrcid*) as a foreign key column in the new **Origin** table.

Currently, only state-of-the-art location algorithms account for correlation between observations, which has been shown to significantly affect both the calculated locations and the uncertainty estimates in some cases (Chang et al., 1983). We have introduced tables to capture this information in the new schema. The individual correlation values are stored in the **CorrEl** (correlation element) table that links together two rows from the **Measurement** table. **CorrEl** rows are in turn grouped into a correlation matrix via the **CorrMat** table. The **CorrMat** table has a one-to-many relationship with **Origin** (i.e., *orid* is a foreign key in **CorrMat**).

Location Uncertainty

As almost no candidate explosion locations are known with perfect accuracy, properly capturing the location uncertainty is essential. The current schema does this with the **Origerr** table, which is a simple extension to the **Origin** table (i.e. it has the same primary key—*orid*—and so just adds additional columns related to location uncertainty). This table captures the full covariance matrix, the one-dimensional (1D) time uncertainty, 1D depth uncertainty and the two-dimensional (2D) horizontal uncertainty ellipse. Three-dimensional (3D) and four-dimensional (4D) uncertainty information is not stored by the current schema. We have introduced separate tables for each type of uncertainty information (**LocErr1D**, **LocErr2D**, **LocErr3D**, **LocErr4D**). The covariance matrix information is stored in its own table (**CovMat**).

Magnitude

We discuss network magnitudes in the Origin section above. Station magnitude measurements are captured in the new **Stamag** table, which is fairly similar to the current **Stamag** table. The changes in this case are deletions. The current **Stamag** table contains numerous redundant convenience columns (*evid*, *orid*, *arid*, *sta*, and *phase*) that can be obtained via joins with other tables, and we have removed these from the new table. If using these joins to access this information proves cumbersome and/or degrades performance significantly, we may reintroduce some of these fields. We have also given the new **Stamag** its own simple primary key (*stamagid*) rather than the current composite primary (*magid*, *sta*), to make it easier to set up foreign keys pointing from other tables to this one.

<u>Lookup Tables</u>

Figure 2 shows the Entity Relationship Diagram for the lookup tables. Again, to make our schema more easily understood by those familiar with the current schema, where tables are analogous to tables in the existing schema we have used the same name. Also again, however, note that none of the tables are exactly the same and some differ significantly, as we discuss below.

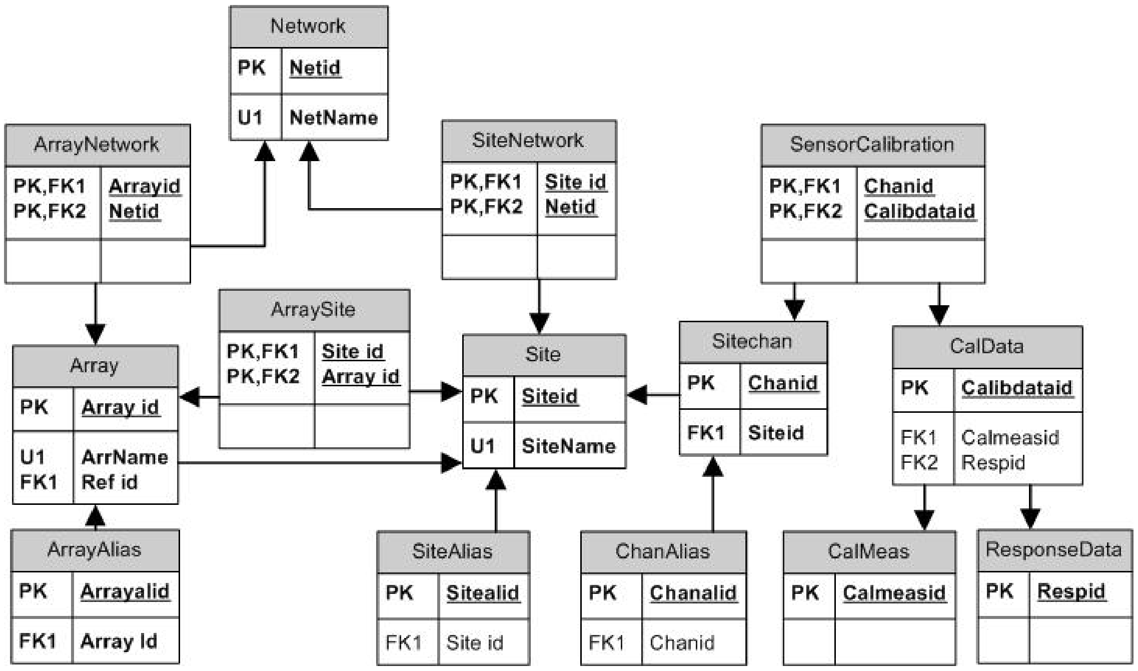

Figure 2. Entity Relationship Diagram (ERD) for the lookup tables.

Network

The top level object for the lookup tables is a network, so we begin there. Our **Network** table is very similar to the current table, except that we add *starttime* and *endtime* to capture the time period during which a network is operational.

Sites

In the current schema, the **Site** table captures the information about the name and location of where seismic instrumentation has been deployed, and we do essentially the same thing. There are however, a few important differences between the new and current **Site** tables. First, we introduce a simple primary key (*siteid*) for the new **Site** table to make it less cumbersome to create foreign keys to point to this table (the current primary key is *sta*, *ondate*). Second, we remove the array-specific columns (*dnorth*, *deast*, *refsta*) from the **Site** table. Storing these in Site is both inefficient because many sites are not part of arrays, and confusing.

Site Groupings into Networks

In the current schema this is done with the **Affiliation** table, and we use a similar table, though we give it a different name—**SiteNetwork**—due to the way we choose to handle arrays. Also, we add *starttime* and *endtime* columns to capture the period during which a site was part of a network. Note that **Network** to **Site** is a many-to-many relationship, so **SiteNetwork** has foreign keys pointing to the primary keys of each of these tables (*netid*, *siteid*).

Arrays

In the current schema, an array is treated just like a network, and so networks of arrays are captured with hierarchies of networks. Though some see this as clever and efficient, we have found it to be confusing and inefficient. In the current schema, for a given array, all elements, including the name given to the array center or beam position (e.g., NORSAR), are rows in the **Site** table. The fact that they are all part of the same array is supposed to be captured in two ways (but seldom is). First, all are given the same *refsta* value in **Site**, which should be the name of the array itself. Second, the array is added as a row to the **Network** table and **Affiliation** rows are added to tie each element to the array. A row is then added to the **Affiliation** table to tie the array to a **Network** (e.g., IRIS). Because the latitude, longitude information stored with each site may not be accurate enough for array processing (e.g., FKs), offsets relative to the array center are stored in the **Site** table (*dnorth*, *deast*). As pointed out above, this is inefficient because all site rows carry these columns, even though most sites are not array elements. A further problem is that if a site is to be used as an element of more than one array (easily done given that arrays are virtual entities) only one set of offsets can be stored.

In our new schema, we take a much more direct approach and model arrays according to the simple fact statements "arrays consist of one or more sites" and "zero or more arrays may be associated with a network." In our approach, there is an **Array** table that captures the basic information about the array (e.g., name, and a foreign key pointing to **Site** to indicate which site position is the center or beam position). **Array** to **Site** is a many-to-many relationship, so a new association table **ArraySite** is introduced. This is the table where the offsets (*dnorth*, *deast*) are stored, as well as the *starttime* and *endtime* of each site's membership in the array (which is not captured at all in the old schema). The many-to-many array membership in networks is captured with another association table, **ArrayNetwork**. Again, this table has *starttime* and *endtime* columns to capture the period of membership for each array in a network.

Channels

Each site is composed of one or more seismic instruments to record the motion of the Earth. Information about each instrument deployed at a site is captured in the current schema with the **Sitechan** table, and our **Sitechan** table is nearly identical. One important difference is that we explicitly model the one-to-many relationship between **Site** and **Sitechan** by giving our **Sitechan** table a foreign key pointing to the primary key in **Site** (*siteid*). We use *starttime, endtime* columns in **Sitechan** to record the operational period for the particular channel, which may be different from the *starttime, endtime* appropriate for the site as a whole.

Aliases

A further source of confusion with lookup information not captured by the current schema is that arrays, sites, and channels can all have aliases (i.e., synonyms). For example, a short period vertical channel may be SHZ or SZ. The fact that such names are equivalent is not captured in the current schema, and this can lead to errors when using data. We fix the problem by introducing a set of very similar tables (**ArrAlias**, **SiteAlias**, **ChanAlias**) to link the various aliases to the "true" name. Each of these tables has *starttime* and *endtime* columns to capture the duration of the alias.

Mapping the Current Schema to the New Schema

Should our proposed schema be chosen for use with real monitoring systems, large amounts of old-format data would need to be converted to the new schema. This is easily done, as we show in Figure 3 for a few of the key tables.

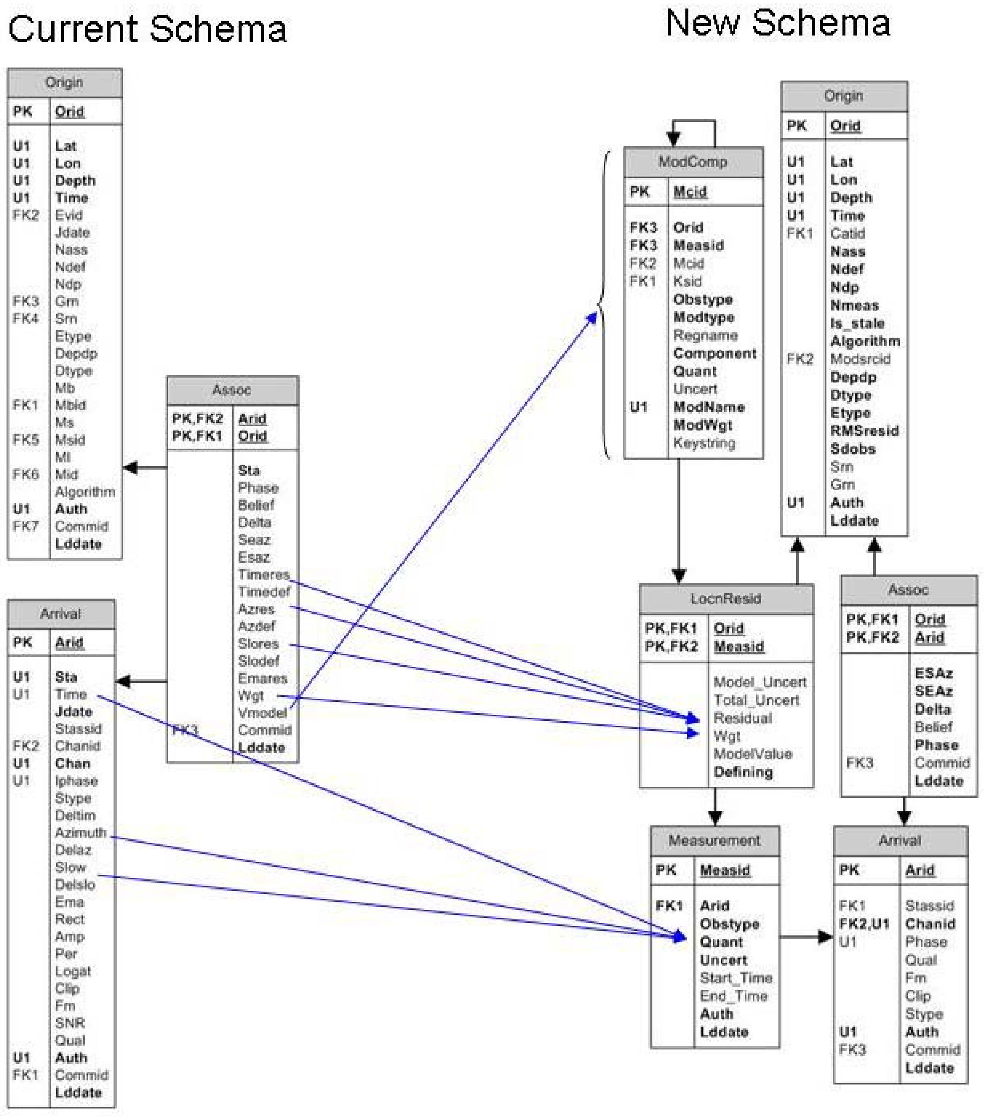

Figure 3. Remapping current schema tables to new schema tables.

In this example, we show how an origin and associated arrivals used for location would be converted from the current schema to the new schema. Much of the information in the current **Arrival** table (*phase*, *qual*, *fm*, etc.) would map directly to the new **Arrival** table. Travel time, azimuth, and slowness observations and estimated uncertainties (*time*, *deltim*, *azimuth*, *delaz*, *slow*, and *delslo*), however, would be mapped to separate rows in the new **Measurement** table. Information from the current **Assoc** table would be mapped to 3 tables in the new schema: **Assoc**, **LocnResid**, and **ModComp**. Residual values for travel time, azimuth, and slowness (*timeres*, *azres*, *slores*) would be mapped to separate rows in the **LocnResid**. The same would be true for the weights for each observation type, except that the current schema only stores one of these (*wgt*), so we would have to specify which observation weight to map to in the new schema. The woefully inadequate *vmodel* column in the current **Assoc** table essentially gets remapped to the entire **ModComp** table, though there is not enough information available for historic events to properly populate the new table. Presumably the string in *vmodel* could be mapped to *modname* in the **ModComp** table. The remaining information in the current **Assoc** table (*esaz*, *seaz*, *phase*, *belief*) maps directly to corresponding columns in the new **Assoc** table.

Similar remaps can be made for the rest of the information in the current schema, and using these remaps we have written some simple scripts to translate information archived in the old format schema into the new schema.

Unfortunately, by using the current schema in the first place, some critical information was never stored properly (e.g., detailed information about the models used for location), so the new tables will not be fully populated for historical data. There is still a gain in doing the translation because the new schema is more efficient and more amenable to code development, but the full benefit of our data model can only be realized when it is used from the start to archive data.

CONCLUSIONS AND RECOMMENDATIONS

We have designed a new database schema to manage information related to nuclear explosion monitoring that captures all of the information stored with the current schema but overcomes many of the problems with the current design. Guiding principles to which we sought to adhere in our design include

- all tables that store fundamental quantities (**Origin**, **Arrival**, **Site**, etc.) should have single, numeric, primary keys;
- having columns that are very frequently null in a table is to be avoided whenever possible; and
- the schema should form a cohesive, integrated entity with settable primary, unique and foreign keys.

Our design

- allows **Origins** to be grouped together in multiple different ways to reflect **Events** defined by different researchers/institutions and to allow the definition of groups of **Origins** that were located simultaneously using Joint Hypocenter Determination;
- properly normalizes magnitude information;
- adds tables that document the source of earth model information and the various components that constitute the predictions of observed quantities;
- defines the **Measurements** made on an **Arrival** separately from the **Arrival** itself, thereby reducing redundancy and enhancing extensibility;
- associates **Measurements** with **Origins** using **Residuals** in addition to the currently defined association between **Origin** and **Arrival**; and
- defines **Sites**, **Arrays** and **Networks** in a way that is more intuitive and that promotes discipline.

Our schema is backward-compatible with the current schema in that it is possible to capture all information stored in the current schema, and views of the new schema can be constructed which allow current software to access information stored in the new schema.

ACKNOWLEDGEMENTS

We thank Richard Stead, Julio Aguilar-Chang, Stan Ruppert, Dorthe Carr, and Mark Harris for countless discussions about problems associated with the current schema and how they should be fixed.

REFERENCES

Anderson, J., W. Farrell, K. Garcia, J. Given, and H. Swanger (1990), Center for Seismic Studies Database: Version 3.0 Structure, SAIC document.

Carr, D. (2004), National Nuclear Security Administration Knowledge Base Core Table Schema Document, Sandia National Laboratories, SAND2002-3005.

Chang, A. C., R. H. Shumway, R. R. Blandford, and B. W. Barker (1983), Two Methods to Improve Location Estimates—Preliminary Results, *Bull. Seismol. Soc. Am.* 73: 281–295.

Oo, K., D. Irons, A. Henson, and C. Morency (2003), Database Design Description (DBDD) Phase 2, Rev C, United States National Data Center (US NDC), SAIC-01/3047 (REV C).