Fermilab MTF

Doc Number: MTF-92-0004
Version: 1.6
Category: DATABASE DOC

# Design of the **admin** Virtual Database

Bruce C. Brown and Kelley Trombly-Freytag
*Fermi National Accelerator Laboratory* *
*P.O. Box 500*
*Batavia, Illinois 60510*

10/1/92

*Operated by the Universities Research Association under contract with the U. S. Department of Energy

# Contents

**Abstract**

The admin virtual database contains administrative information for the MTF database. Information related to the personnel, including their computer usage and roles, the serial number system, and a general parameter system to be used when needed in this and other virtual databases are the items currently implemented in this database and documented here.

# 1 Introduction

This document is written in conjunction with the information stored in the TOT database where detailed items are described and in the admin idraw pictures admin_1.idr and admin_2.idr.

# 2 Serial Number System

Using serial numbers for the primary key in most tables provides several benefits to the MTF Database Schema. Since the primary key of most tables is a serial number, a column of datatype sntype can reference a row/record of information (data) in a variety of tables. For tables for which the primary key was generated, one can reference the serials table where the serial numbers are enumerated in order to know where that data is stored. For tables which reference the serial number generated for another table, a foreign key should be used to insure referential integrity. The foreign key relation will allow one to find the primary key table.

When creating and registering a new row in most tables, a new serial number must be created, and the serial number and table in which it is to be used, stored. When one inserts a serial into this table one must update the value stored in the current_serial table in which the present value of the serial number is stored as 'lastval'. We have selected for our serial server a trigger-based system using the serial generator code shown here:

```
Serial Generator code

create procedure admin.serial_generator
(@target_table varchar(200),@nextval int output)
as
begin transaction
declare @dt datetime
```

```
update admin.current_serial set lastval = lastval + 1
select @nextval = lastval from current_serial
select @dt = getdate()
insert admin.serials (serial, issued_dt, target_table_name)
values (@nextval, @dt, @target_table)
commit transaction
```

TOT generates a trigger for a (arbitrary) table which, when observing that upon storing a row in a table that the primary key is NULL, executes the trigger which utilizes the above stored procedure and then stores, as the key (of that table), nextval.

We have not established true guidelines of the desirability of ever deleting a row from the serials table. But it clear that the delete trigger must be created such that a row cannot be deleted from this table while the serial number is stored in any other table. Note that the serial numbers are used not only in the primary table in which they are key but also in other tables in which they are foreign keys. The enforcement of referential integrity on these tables of course ensures that only the primary table need be checked for the key to be deleted before deleting a serials table entry.

## 2.1 Description of serials Table

The serial table is a sequential enumeration of serial numbers and tables in which they are employed and the datetime when they were issued.

### 2.1.1 serial

This column holds the real, actual, authentic serial number. Yes, the one used in all tables everywhere. It is created here and copied everywhere else.

Currently we implement serial numbers as SYBASE integers. Since we label them in all cases as of type 'sntype' we are able at some later date to replace the integer type with some alternative data type with a minimal impact on the general database system. A new serial number generator would be required.

### 2.1.2 target_table_name

The serial defined in this row is *created* for use as the primary key of the table whose name is stored in this column. We store this to help locate data. Elsewhere in the database, in a column of datatype sntype one can reference a data row of any table whose primary key is a generated serial number. The target_table_name specifies the table for which this serial number is the primary key.

Since we have not provided a separate data owners column in this table, implicit in our design is the assumption that table names are unique across the domain of data owners for which this serial server will be used. This is a requirement which we enforce for tables created using the TOT tool.

Table Maintenance Information

- **Inserts:**MAX
- **Updates:**NEVER
- **Deletes:**NEVER (at least, this is present policy-Mar91)
- **Frequency of Access:**REAL_TIME
- **Maximum Size:**HUGE

## 2.2 Description of current_serial Table

### 2.2.1 lastval

The current_serial table contains only the last value used for a serial. It is only here for use by the serial_generator. It is the non-volatile storage of the 'seed' for the serial_generator.

Table Maintenance Information

- **Inserts:**NONE !!!
- **Updates:**DO NOT TOUCH THIS! This is updated by stored procedure only. Note however, that a *new series* of serial numbers can be created by updating this number when creating a system for a related group of users on a separate SYBASE server
- **Deletes:**NEVER
- **Frequency of Access:**REAL_TIME
- **Maximum Size:**One Row

## 3 Parameters System

When using the database to drive measurements, it is valuable to have the ability to attach data values to specific records. For example, we may need to attach a data value, or parameter, to a specific checkitem, such as the type of instrument needed to carry out the checkitem. Every entry in a table, however, would not need this particular data (the item may have nothing at all to do with instruments); and different entries might need different datatypes for their parameters. Using the last example, the type of instrument may be defined via a serial number type, pointing to another table of instruments types. A different checkitem may need an integer parameter to initialize a checkitem loop. Inevitably, the need for attaching values to records is not specific to any one virtual database, but can be found in various places in the virtual databases.

Parameter tables will be placed in each of the virtual databases where a need is perceived, but there is always the possibility that only one or two records in one table would need parameters, and it would be silly to create a table in the virtual database for that purpose. Therefore, generic parameter tables are placed in the admin virtual database to provide for these infrequent cases, and to provide for any parameters that might be needed by the admin tables themselves.

Any parameter tables are created with the basic premise that knowledge of what type, if any, of parameter goes with a specific record is known to the user of the record, and this knowledge is stored outside of the database system, probably in a program. Therefore, that user/program must know that a parameter is needed for that type of record, and check each time for parameters in all of the parameters tables. To make the parameter concept as general as possible, a single record may have more than one parameter attached. To identify which value goes with each possible parameter for a record, an identifying label, conceptually identical with a variable name, is a part of each parameter table.

Five parameter tables are created, one for each basic datatype (integer, float, ASCII, and serial_number), plus one to handle arrays of floating point values. The structures of the four basic datatype tables are functionally identical. They hold the serial number of the record which the values belong to, the label for the specific value, and the value itself. Obviously, the datatypes of the serial number and label fields are the same in all of the tables; only the datatype of the value field varies among int, float, varchar and sntype in the specific tables.

The table that contains the "arrays" of floating point variables is the same in concept as the basic datatype tables, but adds a sequence number. The need for this table is foreseen for a measurement which needs to be initialized with an array of values. This could be done via the float parameter table, but the label for each item in the array would have to be unique, as in x_coord1, x_coord2, x_coord3, etc. If the array is of any length, this make for clumsy data entry as well as excessively lengthy program code. By using the sequence number, one label, like x_coord, can be used for all of these related entries.

Note that the only way to get from a value in a parameter table to the record the value helps to define is through the admin.serials table.

### 3.1 Description of ascii_params Table

The ascii_params tables contains those character string values that need to be attached to a specific record in some virtual database.

Table Maintenance Information

- **Inserts:**LOW
- **Updates:**DBA
- **Deletes:**HOUSEKEEPING
- **Frequency of Access:**SUPPORT
- **Maximum Size:**LARGE

### 3.2 Description of float_params Table

The floating point values for specific records in some virtual database.

Table Maintenance Information

- **Inserts:**LOW
- **Updates:**DBA
- **Deletes:**HOUSEKEEPING
- **Frequency of Access:**SUPPORT
- **Maximum Size:**LARGE

### 3.3 Description of int_params Table

The integer values for specific records in some virtual database.
Table Maintenance Information

- **Inserts:**LOW
- **Updates:**DBA
- **Deletes:**HOUSEKEEPING
- **Frequency of Access:**SUPPORT
- **Maximum Size:**LARGE

### 3.4 Description of sn_params Table

The serial number values for specific records in some virtual database. This will obviously link two records in the database. The entire contents of the record specified by the serial number in the value field can be considered the "values" of the records specifying this parameter.
Table Maintenance Information

- **Inserts:**LOW
- **Updates:**DBA
- **Deletes:**HOUSEKEEPING
- **Frequency of Access:**SUPPORT
- **Maximum Size:**LARGE

### 3.5 Description of float_table_params Table

Sets of floating point numbers are the values in this table. The set is linked by the same item_sn serial number and ordered via the sequence_number.
Table Maintenance Information

- **Inserts:**LOW
- **Updates:**DBA
- **Deletes:**HOUSEKEEPING
- **Frequency of Access:**SUPPORT
- **Maximum Size:**LARGE

# 4 Personnel and Related Computer Application Administration

We want to know the persons involved in activities. We will use this information to clarify questions when uncertainties about actions or results are needed. Each computer account will have definite roles. Some persons may have more than a single account so that their actions in various roles can be carried out with suitable authority and permissions which can be defined within UNIX, SYBASE and perhaps other software.

This implementation of permissions will be based on a hierarchical model in which each higher group has the permissions of lower groups plus, perhaps, additional ones. This is the intention of the present design but the implementation does not require this hierarchy.

## 4.1 Description of personnel Table

This table will have one row for each person of interest to the project. Their identification will be in terms of personal information - names, and institutional information - institution and personnel_id. In addition to the alternate key, we will try to be sure we have unique identification of individuals by creating an unique index on the combination of last_name and entry_dt.

### 4.1.1 alias

This string should be the convenient short 'nickname' for a person which will allow each data entry for identification. It must be unique. As the alternate key for the table, this will be guaranteed by the index which will be automatically created by TOT.

### 4.1.2 full_name

This should be the full name a person would want for any formal use. It is the responsibility of persons making entries to see that this is suitably consistent with the names shown as last_name, first_name, and middle_name_or_initial.

### 4.1.3 e_mail

This column is to contain a sufficient description of an Electronic Mail address that a program can use it to send mail from a UNIX machine using Internet addressing (*i.e.* SMTP - Simple Mail Transfer Protocol). Note that we have attached this to the person rather than to the user_name. We assume that each person wants mail at a single location, not separate locations for each computer account.

#### 4.1.4 entry_date, exit_date

We store these values for two reasons. This will permit procedures to validate the personnel specified in data entry. It will also assist the data aficionados to recognize people or understand why they do not.

#### 4.1.5 entry_operator_sn

The only special thing about this column is that it is uses a foreign key to this table! In usual parlance, this a a bit strange since when the full trigger is in place, one could not enter data into an empty table. However, we will solve this by adding some critical users before attaching the trigger. We assume that other users of this design will be likewise creative. We think that the design is correct, if clumsy in this aspect.

Table Maintenance Information

- **Inserts:**LOW
- **Updates:**TEST_MANAGER
- **Deletes:**NEVER
- **Frequency of Access:**SUPPORT
- **Maximum Size:**SMALL

### 4.2 Description of user_accounts Table

The user_account (username in VMS or login account for UNIX) for each account in the measurement system computers will be listed here. This will allow applications to determine, using system services, the personnel_sn of a user from the account in use without further entry by that user. Some measurement system roles require specialized computer or database privileges for access to specific resources. We utilize specific accounts for these specific purposes such as *e.g.* MEASURER account and DEVELOPER account to facilitate management of these resources. The same person may have accounts for different roles he plays in the organization. Note that the person (personnel table entry) does not have the 'group' status but that accounts are created for each significantly different role a person may have so that suitable access and privileges are granted to him for that role.

#### 4.2.1 user_account

Note that we permit the name to have 30 characters (namestr). UNIX account or login names are limited to 8 characters, but 31 characters are permitted for a VMS USERNAME. Often is it useful to limit VMS user names to 8 characters; larger than 12 will break some utilities. 30 characters are permitted in SYBASE for the user login names.

#### 4.2.2 sybase_account, unix_account, vms_account

These bit fields are used as a set to identify where the user_account named in this row is valid. Note that where the sybase_account is true the user will both have an account in all MTF SYBASE servers and be a user in the 'mtfmsa' databases on those servers.

Table Maintenance Information

- **Inserts:**LOW
- **Updates:**DBA
- **Deletes:**HOUSEKEEPING
- **Frequency of Access:**SUPPORT
- **Maximum Size:**SMALL

### 4.3 Description of groups Table

This table is just a list of valid groups. It identifies the groups into which work roles in the measurement system are divided for the purposes of granting access and permissions. Control of the access to data and hardware by these groups will be defined by the various applications employed in the system. The currently defined groups include:

public

data_clerk

measurer

test_manager

developer

administrator.

### 4.3.1 group_name

We might have chosen to call this column 'group' but that is a reserved word for SYBASE.

Table Maintenance Information

- **Inserts:**LOW
- **Updates:**DBA
- **Deletes:**HOUSEKEEPING
- **Frequency of Access:**SUPPORT
- **Maximum Size:**SMALL

## 4.4 Description of sybase_permissions Table

This is just a list of valid Sybase permissions. SYBASE permissions which we choose to set up for our groups will be listed here.

Table Maintenance Information

- **Inserts:**LOW
- **Updates:**DBA
- **Deletes:**HOUSEKEEPING
- **Frequency of Access:**SUPPORT
- **Maximum Size:**SMALL

### 4.5 Description of group_sybase_permissions Table

This table allows one to associate permissions from the permissions table with groups from the groups table. Since a group is identified for each user_account, one can use this table to specify the permissions for each user. Its purpose is to allow an application, such as TOT, which creates tables for the measurement system, to automatically generate relevant permissions for all of our defined groups.

Table Maintenance Information

- **Inserts:**LOW
- **Updates:**DBA
- **Deletes:**HOUSEKEEPING
- **Frequency of Access:**SUPPORT
- **Maximum Size:**SMALL

## 5 Miscellaneous Tables

These are tables that need to be in the admin virtual database by virtue of its application to the entire system (or more than one part of the entire system).

### 5.1 documents Table

This table provides information on any documents that may be written that pertain to specific entries anywhere in the measurement system. For instance, documents that provide procedures on how a measurement should be taken will be described here and used in the logbook.activities_log table. Documents that explicitly describe the content and functionality of conceptual instruments, or conceptual instrument classes will be referred to here by the instruments.conceptual_instruments and instruments.concept_instr_classes tables.

#### 5.1.1 alias

The alias column, a short "handle" for the document is needed for the logbook.activities_log entry, where it was decided not to use the serial number of a record to ensure an accurate historical record of a measurement. The entries in this column should be unique, therefore a unique index will be used. For documents which are part of the MTF document system, the MTF Document Number will be the alias. Similarily, Fermilab TM or FN numbers or similar document series labels can (and should) form the alias when appropriate.

#### 5.1.2 type

The type column has free-form input, but meant to give an application a clue as to how to present the document to a viewer. A type is something that tells you about the documents format: e.g.,TeX, troff, MicroSoft Word, etc.

#### 5.1.3 location

The location is a path to a computer file, or a description of where a paper copy may be found. This dual purpose prevents using the locations table setup in the facilities virtual database.