

Paper AA300

Database-ics A Primer for Creating a Usable Database in SAS

Frank Ferriola, Pacioli Companies, Highlands Ranch, CO

ABSTRACT

Companies today drive key business strategy from the analysis and reporting of corporate data. Using data with integrity checks improves the outcome of this strategy along with eliminating the pitfalls of determining which report is the truest measure of the company's operations. By addressing gaps in the data and fully populating all fields, organizations can create centralized databases giving them a "single version of the truth." This presentation overviews the strategy and standards used by a major financial services organization in deploying databases utilizing SAS® Software. We will look at building databases with integrity checks and review the steps for taking inventory of the data, defining/re-defining its use, and populating fields. These practices ensure the construction of "single version of the truth" databases providing organizations with credible information to drive successful strategies.

INTRODUCTION

In this paper we are going to explore ways to create a usable database from corporate data. We will go through the general steps of ensuring that the database that is created serves the purpose it was created for: to limit the versions of information coming forward in the future. You will see that this reads pretty much like a project plan, because essentially it should be done in that manner. To do anything less is to shortcut your process and will ultimately lead to flaws in your data and ultimately you have to run through the process again. Let's look at the steps:

- Decision making
- Scope
- Requirements
- Design
- Creation
- Implementation
- Ongoing Maintenance

Before we get too deep into the paper, however, I must clarify what it is we are actually creating. I have worked at many different places over the last 20 years, and at all of these places, I was required to create some sort of data source at some point. You may have looked at this paper expecting to build a database. Others may think what we are creating here is a data warehouse, or a data store, or data mart. The approach I am presenting here, could be any of those things, however, it could also be none of those things. All of these data sources are in some way pre-defined, although it may still be different for each of you. So in pondering that idea some more I decided to come up with my own data terminology with my own definitions. And what we will be creating is something I call a databurt.

DATABURT

The first known databurt was created in 2004, at a major financial services company in the Midwest. It actually started out as a datamart, however the project team ran into all sorts of problems because they weren't in the right department to create a datamart and were getting flak from those people who were allowed to build such datamarts and our project team was finding itself spending more time explaining themselves than actually building it. So I went to our project manager and told him to start referring to it as a databurt. It was all a matter of semantics. We continued to build and created the databurt (although it is still referred to as a datamart, but not out loud.)

The databurt has three criteria:

- 1) The databurt is one self-contained library of tables. In SAS this means that the end user should be able to access the files with one libname statement. It can have one or many tables, and these tables do not all have to loaded at the same time.
- 2) The databurt is the data source that is the closest user. I refer to this also as the most downstream.
- 3) The Databurt is completely ready-made data. The end-user should make no transformations in getting the data from the source to reporting. Formatting is acceptable, however the format catalog should be controlled by the Databurt.

TRUTHS OF DATA

Over the years, I have compiled a list of what I call the Truths of Data. It seems to be true no matter what industry I happen to be in, or even whether it's a small or large company.

- 1) Data is Everywhere
- 2) Everyone uses the data in some form
- 3) Bad data will always guarantee bad results. Unfortunately good data does not always guarantee good results.
- 4) One version of the truth is always better than many
- 5) SAS is a great tool for creating a databurt

DECISION-MAKING

Many factors have to go into the decision to create a databurt. The first is to see what data presently exists. There are several questions you can ask to determine if you need a databurt.

Do you need a Databurt? You may currently have something that could be a databurt. Even if it doesn't exactly match the definition above, perhaps with a little modification it can be turned into one.

Is the current databurt adequate? The quick way to determine this is look at how many transformations are occurring after reading the databurt.

What are the benefits of creating a databurt? The most important reason is that you will have one version of the truth. The secondary reason is that you will have less processing time to get your analysis and reports out, since there are no transformations each time you run.

What are the costs of creating a databurt? To build a databurt properly, it will take an extensive amount of time. Also depending on how large the databurt is, there could be storage considerations.

SCOPE

After going through the above questions, and then deciding to proceed with a databurt, the first thing to consider is the scope. The first part of defining the scope is to look at the users who will use the databurt. The more users you can define, the better your databurt will be, however, it also causes more problems as you expand into other areas. The key to defining the users is making sure that whatever group of users you define, there is some level of management that can effect consensus agreement on the business rules.

Once you have determined who the users are, the next step is to identify the uses of the databurt. This should be fairly simple. Any of the applications, reports, or other programs that are run by the defined users for any kind of production uses that the identified users run or receive.

The measurement of success is that the defined users should never have to go outside the databurt in order to process the defined uses.

REQUIREMENTS

The next step in the process is to gather the requirements. There should be existing programs which run against source data already and creates transformations of data. A business analyst together with a programmer can comb through these programs and determine the variables that are being transformed, and can continue to look for consistencies and inconsistencies in the transformations.

What you will find is there will be several variables with different names that are defined the same. You will also find that several variables that appear to be the same and may even have the same name, are defined differently. A good way of compiling a list of all the variables is creating a data dictionary or a map of all the current programs. (see Figure 1) After compiling a list of all of these variables, their sources and transformation rules, analyze the inconsistencies between variables and the variables that appear to be the same or at least similar. These need to be returned to the users to decide what is the best use of a variable, and the list can be refined until all redundant and inconsistent variables are resolved.

It is also helpful here to create a data dictionary which will help define and maintain the variable list needed on the databurt. You will also see inconsistencies between similar variables or different users more quickly. Once the rules are defined they can be put into the data dictionary and used as pseudo-code later.

DESIGN

Once all the requirements have been defined, the next step is design. This is a very important step which is many times overlooked before moving on to writing the actual code. Spending a good portion of time on design however will ultimately make your coding go easier.

Design Structure

First you need to define the structure that you application will take. Things to consider here are where you will process—folder structure of the area which will be different based on the platform you use and what control you have over security. Consider the other aspects and limitations of your application and how your program will overcome this.

Determine Variable Rules

When determining variable rules consider naming convention. The more you can have your variable names make sense the better off you are because you will get fewer questions in the end. One rule to absolutely have is that if you do any transformation to another variable, it needs to be renamed to avoid confusion with the variable that resides elsewhere.

Other things to consider are prefixes and suffixes. Prefixes will group all like variables together. Examples would be business, program, cycle original, or current. Suffixes can tell the user what to expect in the field. Flag or ind would indicate that it is a some kind of indicator. Make sure that it is always used the same way. If you use 1 and 0 for one variable designated as flag then use 1,0 for all variables with that name. Same with “Y” and “N” or “Yes” and “No”. Other suffixes are amount or amt, percent or pct, code, or value. You can come up with your own, but remember, consistency is most important.

Flow Chart

Make a flow chart of your design. You should develop this early, to start although it may change as you add more information. With SAS I like to have a main program or group of programs calling in macros and formats. This is pretty streamlined and modular and helps when trying to make changes or debugging the individual macros rather than one big program. You should also use macrovariables as much as possible to pass parameters to and from the programs and macros.

Pseudo-Code

You can now start creating the pseudo code that you used in the data dictionary to outline the macros that you need. The convention I use for the macros is a verb like create_ and then the variable name or group of variable names. For example, if I'm writing code for a variable named business_line_code I will call the macro create_business_line_code.sas. All this macro will contain is the coding for business_line_code. The pseudo code will go directly into the design document.

Step out all of the code in your design sessions. Also make sure the macrovariable parameters are passed properly in and out of programs and macros. Finally, use whiteboards and other tools to help work through the design.

Communicate

Make sure that you involve non-programmers in your design process. Their perspective will be different from yours, and they will raise different questions and make your design even better. They shouldn't determine how you process, however they could ask questions that make you consider other options. Ultimately the users must buy-in to what you are trying to create, and they are more likely to communicate changes, when they feel they have a role in the ultimate product. And remember, if the users go outside of your databurt, then it is unsuccessful.

DEVELOPMENT

Once you have the design document completed, then the development of the application to build the databurt simply becomes the pseudocode from the design put into the proper files. Drop the code into the proper files, document the code and test each macro as you build it. Once all macros are finalized, run the program that executes all the macros and test again. This piece should go very quickly if all of the previous steps were thorough.

You can also put in a GUI front end using SAS IntrNet to create the parameters and execute the process. Also make sure all of the information that you have can be published including the data dictionary/metadata, any validation and testing reports. The more documentation that is run, evaluated and published helps ensure data integrity.

IMPLEMENTATION

As part of your process plan, you should have an implementation plan. This should include the migration to production usually through a testing phase with proper sign off. Make sure you separate duties, so that one person doesn't have control over all parts of the migration. After all the work that was done in the previous steps, you don't want to lose or cause problems with your databurt now. Also make sure your application is locked down and limited users have access to the implemented version.

CHANGE CONTROL/CHANGE MANAGEMENT

You should also create a thorough change control process. Essentially you should ensure that requested changes are approved by all the affected users. If you don't clear it through everyone, then they will be surprised when a change happens to something they are using.

From a SAS standpoint, because you used the programs calling macros, it is now much easier to implement changes, by identifying the macros that are affected, and only changing those macros. If it is determined that a new macro is created, or one is taken away, think through the whole process and make sure that the affected programs have the macro calls in them properly.

CONCLUSION

In order to ensure that you have a usable databurt:

- It is extremely important to be thorough and go through all the steps. Don't shortcut the process, it actually takes longer to figure out what went wrong down the road, than to take the time to do it right and control the process in the first place.
- Get user buy-in from the start which will allow you to perfect your databurt as well as ensure that it will be used.
- Ultimately you will create a single Version of the "Truth" which will eliminate second-guessing of reports and analysis.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Frank Ferriola
Pacioli Companies
3110 W. Deer Creek Dr.
Highlands Ranch, CO 80129
Work: 612-333-9900
Mobile: 303-548-0278
Email: fferriola@pacioli.com
Web: www.pacioli.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.