2471-2018

Database-Ics II: Design, Standardize, and Control the Process for Better Data Management

Frank Ferriola, Charles Schwab & Co.

ABSTRACT

Developmental design is an art, and too many times we dive right into the coding and the results instead of taking the time to think through the process. This paper examines the entire development process and shows the audience different facets to think about during development. Areas covered are design, standardization, and control tables, as well as SAS® tools that can be used to make your process more efficient. This paper updates concepts presented in “Database-ics: A Primer for Creating a Usable Database in SAS®”, which I presented in 2005 at MidWest SAS® Users Group (MWSUG), Western Users of SAS® Software (WUSS), and SAS® Users Group International (SUGI), and adds additional concepts.

INTRODUCTION

In the 13 years since I first presented Database-ics, data and the tools that are used for ETL have changed in a lot of ways. Data Volume has exploded, timelines have become shorter as a need to know increases, and requrements are generally fluid and inprecise. This makes design even more important. But in my experience, design is given even less time to consider because of the quick turnarounds and deadlines become more important.

With that reality in mind, we will delve into the concepts of good design even though it’s likely you will not be able to thoroughly go through the entire process up front. Keeping these ideas in mind will allow you to partially design on the fly and then refine the process with repeated iterations as you go.

Finally, this paper is primarily directed toward the Extract, Transform and Load (ETL) data process, but many of the elements of this paper will work for any development process.

TRUTHS OF DATA

From the beginning of time, there has been data all around, but it has not always been captured and retained. From my years in the working world, I have watched data collection go from pen and paper, to 80 column cards, to floppy disks, tape systems and hard drives to digital storage to the cloud.

In Database-ics I introduced my Truths of Data, and they have changed little at least in the last 40 years:

- **Data is everywhere**
- **Everyone uses the data in some form**
- **Bad data will always guarantee bad results**
- **One version of the truth is always better than many**
- **SAS® is a great tool for creating a Database**
- **Big Data Does not necessarily make data good**
- **More Sources complicates the ETL Process**

In data management, we try to assemble the data in a way that makes it easier for the analytics to be performed on it, which, in turn, is disseminated to stakeholders. In many cases, the data is passed from one system to another within an enterprise, and it’s important that the integrity is kept from the original source all the way through to the most downstream users that will be using the data you put together. Mistakes in the process will happen, but all the people in the data chain need to be diligent in preserving the integrity of the system and passing it upstream to correct if necessary.

DEVELOPMENT PROCESS

Let’s spend some time on the development process There have been many methodologies over the years, such as Waterfall, Extreme Programming, several versions of Agile, and Rapid Application Development.

Regardless of the method you prefer, the process has several elements that are necessary to be successful:

Requirements

Design

Development

Testing

Implementation

There could be other elements that go into specific processes such as integration and maintenance or different names for some of the items, but generally these elements are necessary to follow to create a production capable process. These are not necessarily sequential as parts of the process can be done concurrently. However, Implementation should only occur after the previous steps are completed when creating a closed process.

For instance, have you ever gotten full clear requirements before you finished developing? That’s may sound absurd as the requirements tend to change throughout the process. One thing to keep in mind though: After delivery changes to the requirements becomes change control, but during the process it just becomes annoying, and you need a clear cutoff date for the requirements that have to go into the version you are developing. I have been asked more than once to deliver something prior to getting any factual requirements. If you do not have acutoff in advance of implementation, you will most likely have problems running it.

The items listed above are all important, but in my papers and presentations, I concentrate a lot on design, because I feel it is the most overlooked aspect of the Development Process. Many times, it can cause the development to fail to achieve its objective. For that reason, I won’t go into all the other aspects of the process but will concentrate on the Design.

DESIGN

I mentioned the term “closed process” earlier. This means to me a process where you can run the process in a production environment without changing any of the code or manual intervention into the process including parameters.   Parameters and other information can be injected into the code via other means, which we will discuss later.

To accomplish this, you need to think through the process and create the instructions or recipe for how to best develop the solution. Next to clear requirements which happens occasionally, design is probably the most important part of the process. The problem is that many times you do not have the full requirements prior to starting your design and the requirements tend to change, which affects your design. It is also the rare instance when you can spend an ample amount of time on the design.

This inevitably costs more time later with rework, complete reconstruction, and doing redundant coding work.

CONCEPTS OF DESIGN

Once again in an ideal world, I would recommend you consider the following in your design:

- **A thorough understanding of the Requirements you have.**
- **Utilize existing programs if possible/available**
- **Entire Development team should be involved**
- **Think through and whiteboard the entire process.**
- **Time spent in the Design process saves time down the road.**

TOOLS OF DESIGN

- **Flow Charts**
- **Pseudo Code Steps to Make Sure they work properly**
- **Data Dictionary**
- **Determine Your Rules for coding, process security, as well as Standardization**
- **Take advantage of tools available to you.**

HOW TO START DESIGNING

So how do you start the design process? Ideally, you should begin with a rough draft of a flow chart of the high-level process. Where is your data coming from? (Extract) What needs to be done to prepare that data? (Transform) Where does the data need to go to be stored and how will you store it? (Load) This is the basic ETL process.

UNDERSTAND THE ARCHITECTURE AND INFRASTRUCTURE

In many situations, you are developing on an existing architecture but sometimes you are the pioneer and will help establish the architecture you need to make your process successful. This introduces a lot of other elements that are not in scope of this paper, but make sure your design of this is thorough and complete for your needs.

ENSURE THAT THE SCOPE IS CLEAR

From the requirements, it is essential to define the scope and agree with the stakeholders on what you expect to accomplish with the implementation. Sometimes, as in an Agile Environment, you take bite-size pieces of the requirements in each sprint to accomplish a part of the process. Sometimes developers will split up and work on different parts of the process, which requires all of them to be in sync when the parts come together.

CREATE CODING STANDARDS

If there are already coding standards at your shop, make sure you keep a copy near you while designing and developing so you can follow them. If no coding standards exist, develop them. They don't have to be complex, but cover any items that are necessary to help your team be consistent across all developers. A few of the items below could be incorporated into the coding standards.

EXTRACTING FROM SOURCES

Examine your sources and how they are organized. Chances are you will need to get data from several tables and libraries. Create a list of the tables you need and variables from those lists. After that you need to create a hierarchy of the data and a Data Dictionary that becomes a road map of what data you are sourcing. This Data Dictionary will be used and updated throughout all the steps in the ETL process. Sometimes it is easier to do it all at once, and other times, you may just want to list everything you think you may need in the Extract step.

STANDARDIZE WITH NAMING CONVENTIONS

While examining your data, consider the naming conventions used in the sources. When naming your variables, you can use names that are similar to the source naming. If the source doesn't have a clear naming convention or if the sources have different naming conventions, use your own, but the key here is to be consistent within your own process. This is where you need buy-in from the whole team. In general, I prefer to use prefixes to describe either the source of where the data is coming from, or a common grouping of what the variable is used for. Names like BUS_, CLIENT_, ADDR_are good prefixes. For Suffixes, I suggest using an abbreviation for what type of information it is: _FLG, _IND _AMT, _ID, _CODE with each of these having a precise meaning. A flag should have values of Y or N or ON, OFF; Indicator could be 0 or 1. Amount should only be used for something with dollars (even if it is not formatted that way.) Percent should be a Percentage and always in the same form either the actual percentage 48.6 or the decimal .486 for example. Consistency is important.

Permanent tables should always follow a naming convention. Work tables are easier to follow if they also have a naming convention. Use a sensible way to name your tables, using prefixes or suffixes—or both.

When using dates in the name of tables always use YYMMDD format including leading 0's on month and days less than 10, this will cause your tables to fall together in order when you list them. Since we are almost two decades into the new century, it is probably no longer necessary to use a four-digit year, but there is no problem with continuing to use it except for the 32-character limit for names of SAS Data Sets.

CREATE AND USE SWITCHES

Earlier I mentioned that the closed process needs parameters sent to it. This can be done through various methods.

Control Tables

Control tables contain changeable information that can be used to control the process to set parameters, update information, create formats or data lookups, data mapping or even to store code that needs to be placed in the process. In the Reference Section I have a paper that looks at control tables and how to use them. When you name your control tables think about what you will use it for. Create a system of prefixes/suffixes to make it easier to know what the data is used for. Some examples are LKP_, XREF, FMT_ DM_ and PRM_ .

Macro Options

When writing macros, make sure you make them dynamic by adding parameters that can be turned on and off, to run various scenarios, process based on certain variable values, and any other conditions that you want to set. This allows the control tables to feed information into the macros and have them perform differently depending on those parameters. Writing one macro to do one thing regardless of the situation, is much more efficient than writing several macros that do the same thing under different circumstances. If you find yourself trying to write the same macro twice, think about consolidating it into the first macro even if it means adding another parameter to the macro call.

As always use a naming convention for your macros as well. Your prefixes here can group the macros together by process (EXTR_, TRFM_, LOAD_, indicate what it does (CREATE_, GET_, RUN_) or any other way you want to name them.

WRITE PSEUDO-CODE

As you design, you should create small snippets of your code to test your design ideas. Some of this can be throw away code, but much of it can be put into the actual code either in a macro or other code in the final process.

Programming is an evolution, in that the first code you write usually can be consolidated in some way to be more efficient. Writing pseudo-code allows you to make the smaller snippets of code efficient faster and debugging goes easier, before creating the larger version of the code.

ELIMINATE REDUNDANCY, REPETITION, AND REPLICATION

As you go through the code together as a team, you will usually see ways to consolidate the process as you go. This is much harder to do when you write a lot of code first, and then try to go back to make it more efficient or change it. As you create other processes, always keep in mind the processes you have already built and see if you can leverage that code or even that process for the new process. Sometimes it's as simple as modifying macros with new parameters. Other times it may be a case of performing a larger extract of more data before using different transforms to clean, filter and process the data for loading.

DO NOT HARDCODE

Any information that can be changed should not be hardcoded into the code. Even if you think it won't be changed, do not hardcode it. Any value other than permanent binary values like "Y" or "N", 0 or 1 should not be hardcoded. Even data that seems to have limited choices, for example gender can change over time and should not be hardcoded. For your organization, there will be certain definitions and values for data, but I would anticipate any of them adding additional values down the road.

CREATE DOCUMENTATION AS YOU GO

Documentation often gets left behind in the process, but should not be ignored. Creating a data dictionary flow chart and other pieces of the process, like inventorying the programs macros, parameters and other information are essential to a long life-cycle for your process. Unless you have specific members of the project team to do the documentation, all of the developers should be responsible for the documentation. Set aside a block of time at least a couple of hours a week to concentrate on the documentation of the process.

CONCLUSION

At this point, I hope I have given you some ideas on how important it is to do extensive design on a process. My best arguments for it are:

1. Better design makes better closed-processes
2. Saves time in not having to rewrite information.
3. Catches mistakes that you might make in developing.

The biggest detriment to design is the lack of time provided, but I would argue that in the long run, time is saved down the road to do more things better.

REFERENCES

Ferriola, Frank, 2005, Database-ics A Primer for Creating a Usable Database in SAS, Proceedings of the Western Users of SAS Software, 2005, San Jose, CA. Available at http://www.lexjansen.com/wuss/2005/data_warehousing_and_database_management/dwdb_databaseics_a_primer. pdf

Long, Avery; Ferriola, Frank, 2015, Standardizing the Standardization Process, Proceedings of the Western Users of SAS Software, 2015, San Diego, CA Available at http://www.lexjansen.com/wuss/2015/61_Final_Paper_PDF.pdf

Larsen, Erik; Ferriola, Frank, 2015, A Practical Introduction to SAS Data Integration Studio, Proceedings of the Western Users of SAS Software, 2015, San Diego, CA Available at http://www.lexjansen.com/wuss/2015/64_Final_Paper_PDF.pdf

Ferriola, Frank, 2015, Avoid Change Control by Using Control Tables Proceedings of the Western Users of SAS Software, 2016, San Francisco, CA Available at http://old.wuss.org/Proceedings16/29_Final_Paper_PDF.pdf

CONTACT INFORMATION

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

Frank Ferriola
Charles Schwab & Co.
Frank.Ferriola@Schwab.com