The SAS Data Warehouse: A Real World Example

Martin P. Bourque, SAS Institute Inc., Cary, NC

Abstract

This paper discusses building a data warehouse for the Technical Support Division at SAS Institute. It starts with the decision to build a data warehouse, and proceeds through the planning stage to the exploitation. It focuses on the need to be flexible at all stages, and the reality that building a data warehouse is an iterative process.

Introduction

Nightmare or golden opportunity? The division head tells you to build a data warehouse. He also tells you to do it using a release of software that has not yet passed through Quality Assurance. In addition, you will use a product that is still in the development stage, and finally, your resources are scattered across several departments. While these constraints are unique to this project, you can be sure that your data warehousing project will probably be equally unique.

So where do you start? In my case, the company library. I had heard about data warehousing, read about it in the trade publications and even listened to some marketing people discuss the 'market potential'; but what did those words mean? As you might expect, the library had many books and articles on the subject, and after reading a half dozen books and too many articles to count, I was able to see that the experts agreed at the 'macro' level, meaning 'data warehousing is a good thing,' but disagreed at the 'micro,' 'this is how you should implement your data warehouse'. This was the first lesson learned: listen to the experts but be prepared to design for your own unique situation. In addition, play to your strengths. In our case, that meant the SAS System.

What is a Data Warehouse?

We read and heard many definitions of data warehouse and all seemed reasonable. The one that we devised to explain our work is as follows:

A data warehouse is a repository of accurate, time related information that can be used to better understand your company.

Organizing the Design Team

After researching data warehousing and estimating the required human resources to develop one, the initial team was assembled. The team was responsible for designing and planning the data warehouse. As the only full time member of the team, I became the Project Coordinator. All of the other team members had to perform their Technical Support duties before they could assist in the project. As you might imagine, this caused a few task scheduling problems. On the plus side, I had the opportunity to learn more about many features of the SAS System.

Goal Setting

Once the design team was in place our first task was establishing our goals. We were fortunate that no one required a detailed time line, but we knew that we needed a functional system by the end of the year. With this in mind, we matched our strengths against our requirements and came up with our goal:

Build a data warehouse using the SAS/Warehouse Administrator toolkit and Release 6.12 of the SAS System for UNIX Environments. The data warehouse will have a graphical user interface to be used by Technical Support management. In addition, the data warehouse and the interface should exploit as many of the SAS System enhancements as possible.

Project Planning

Now we needed a project plan. Although many of the books we read during the research stage provided guidelines for building a data warehouse, none of them fit our situation exactly so we used them only as a template. After we 'cut and pasted' various ideas from other sources and added our own unique requirements, we came up with a relatively short set of guidelines. We did not go into a lot of detail, keeping our project plan a set of guidelines meant to keep us focused on our goal, rather than a road map to project success.

Interviewing our Potential Customers

Our next step into the unknown was to interview our customers, the Technical Support managers. Our intent was not to find out what they wanted, but rather what data they used. If we could determine what information they used on a daily basis, we could decide what data had to be collected for the data warehouse. Once the data were collected, reporting it would be a snap, or so we thought.

We did not have a set list of questions for our interview subjects, and we did not require them to prepare in any way. However, we did set a few guidelines for ourselves:

- Pretend that you do not know anything about your interview subject's data. This is tough to do when you have worked in Technical Support but necessary if you do not want to prejudice the results.

- Concentrate on the types of data that your subject uses and the relationships between them.

- Finally, do not lose control of the interview. You ask the questions and keep the subject focused on data and relationships.

Data Modeling

We started the interviews by asking the customers to name ten types of data that they use, which we listed on a white board. Then we started to ask about relationships between specific types of data. We had to be very careful to stay focused and not let the conversation drift off into the realm of deliverables, timelines, specific reports, etc. As nicely as possible, we had to make them understand that they were there to provide us with information about the data used to run the division.

Once the topic or the customer was exhausted, we copied the white board diagram to paper so that the information could be shared with the other design team members. When all the interviews were complete, it became apparent that each customer had his/her own unique perspective and no one knew it all. We laid all the diagrams on a table and derived one uniform picture of the data used and the inter-relationships. Many of you will recognize the process we went through as the beginning stages of creating a logical data model, which is commonly used in the design of relational data bases. Data Warehousing experts disagree on the value of a logical data model in the making of a data warehouse. In our case, we were simply attempting to identify what data were important, and therefore, required to be collected.

There *is* an alternative method of gathering information about your data. Put everyone into one room and conduct the interview once. This method has the advantage of getting consensus on what the data are and how it all relates. The disadvantage is the logistics of getting a dozen managers to sit through a couple of days of this activity. In our case, we believed it would have been impossible, but this method might work for your company

Constructing a Logical Data Model

The interview data were used to construct a logical data model of the Technical Support Division. Once again we did this because some books on data warehousing implied that it would help us to understand our environment and make our design more appropriate. The authors appear to disagree on how formal the logical data modeling process should be. According to some authors, it should be as rigorous as that used in designing a relational data base. For others, it is a worthless expenditure of time. Based on our experience, I believe it is somewhere in the middle. If you have a formal, complete and accurate logical data model already in existence, by all means, use it. If your are starting from 'scratch' then construct a logical data model that provides enough information so that you can identify and understand your operational data sources and how they interact. In its simplest form, logical data modeling is used to identify 'subjects', 'entities' and 'attributes'.

Subjects represent a group of types of data to define one aspect of a business. An example of a subject is the types of data that define the problems or questions that we receive from our SAS System customers. Each subject is comprised of closely related types of data elements that are called *entities*. An example of an entity is all the data on a customer who has contacted SAS Technical Support. Each of these entities is further subdivided into attributes, such as customer phone number.

When we constructed our logical data model we arrived at five unique subjects.

- PROBLEMS - inquires entered by Technical Support consultants or electronic services for individuals and the companies they represent

- ORIGINS - the source of inquires entered by Technical Support consultants or electronic services

- STAFF - data about Technical Support Consultants

- TSDEPT - data about Technical Support departments

- MISC - information about related information.

Of all of these subjects, we selected one, PROBLEMS, to initially implement to gain an understanding of the whole data warehousing process. I would recommend this approach. We went to great lengths to explore the selected subject. The process started and ended with a simple statement defining the subject.

Once we had defined the subjects, we subdivided each subject into its entities. This process also started and ended with a simple but succinct statement defining the entity.

We then defined all the attributes that comprise the subject. An attribute can be a single data element or field, such as a customer phone number. This process is complex in that the characteristics and domains of all the attributes must be defined.

- Give all of the attributes a unique name. This is very important. Your operational data may have the variable 'phone' in several files, each one with a unique meaning. In a data warehouse there is only one 'phone' column, although there may be many columns that represent phone numbers.

- Define the attributes. Characteristics such as numeric or character, field length and value ranges are identified here.

- Identify the data source of this attribute. From where will this column be drawn? Will it be calculated? If so, from what and how?

- Identify the 'domain' of the attribute. What are the valid values that it can contain? This is very important when the issue of data scrubbing must be addressed.

- Define the data transformation or data scrubbing that needs to be performed on a piece of operational data before it can be placed into the data warehouse.

This analysis and the documents that resulted became very important as the project progressed. Some of you might recognize it as 'metadata', that is, data about data.

Designing a Physical Data Warehouse

At this point in the project we had to decide on a physical data warehouse design. Naturally it was a foregone conclusion that we would use SAS data sets, but what kind of schema? There are two major schools of thought on this issue, Star schema and its derivatives or Relational schema and its derivatives.

A Star schema has 'facts' and 'dimensions'. A fact is a piece of data that is additive. For example, gross sales can be accumulated by day, month or year. A dimension is a grouping of similar data, such as all information related to sales, less its facts. One advantage of star schema is that it originated with data warehousing as its focus. It is therefore optimized for fast access to large tables of data.

The Relational schema was designed with online transaction data collection as its focus. It is optimized for getting data into and out of a data base. Relational schema has the advantage of being well understood. For this reason, we chose to design a relational schema.

Choosing our Development Platform

In conjunction with the planning and documenting noted above, we were exploring Release 6.12 of the SAS System for UNIX Environments. We wanted to showcase the new features of the release in our data warehouse exploitation phase. Originally we had planned to use PC's running Windows 95 instead of UNIX workstations, but our request for additional PC hardware was turned down. Once again we learned the valuable lesson that we must be flexible at all times.

Working With SAS/Warehouse Administrator

The best way to think of SAS/Warehouse Administrator is as a repository for all the information about your data warehouse. It is not a DBMS, and it is not the warehouse. Data warehouses can and have been built without SAS/Warehouse Administrator, but we found it to be a great way to leverage our metadata into 'meta-information'.

Data warehouses are capable of creating a lot of metadata. Many times the metadata consist of hand written documents or computer text files. SAS/Warehouse Administrator offered us the opportunity to create and save metadata in an easy to use manner, as well as a set of tools to automate and organize the building and maintenance of a data warehouse. At the time, SAS/Warehouse Administrator software was under development. This was not necessarily a bad situation as it gave us a chance to provide user feedback to the developers. As we learned the product we reported 'bugs' that were rapidly fixed and we requested features, many of which were implemented. In essence, we helped shape the product, and not too many users have that

opportunity. On the down side, we were using code that was very new and we constantly had to deal with the question, ' is it a bug or a feature?' Occasionally, after we had learned how to use a feature, it was re-written. If it was not for developers who wanted the best product they could build, life would have been very difficult.

Eventually we reached a point in our analysis and knowledge where we could start loading information into SAS/Warehouse Administrator. This meant specifying the operational data that was used to feed the warehouse. It also meant defining the initial subject and the detail tables that made it up. The detail tables had columns with attributes, so this information had to be entered. We also had to define to SAS/Warehouse Administrator the data scrubbing routines and the data transformation to perform.

At this point we found out that our earlier analysis was sometimes right on target, and at other times way off. The time we spent on defining our subjects, entities and attributes were 'right on target,' but our assumptions about capacity planning, granularity and data scrubbing were not. Fortunately, SAS/ Warehouse Administrator made adjustments relatively easy. The most valuable features were its use of global metadata, its ability to generate code from entered metadata, and its interface to the SAS System Access products.

When all of this was completed, SAS/Warehouse Administrator was able to generate SAS code to load the operational data into the data warehouse.

Building the Warehouse

Some of the issues we had to resolve as we built the data warehouse were:

- Job Scheduling

  SAS/ Warehouse Administrator had a job scheduler but it only supported UNIX system V cron. Also, it would only schedule on the processors where you were running SAS/Warehouse Administrator. There was also no facility for conditional execution of jobs. We got around these situations by saving the SAS/Warehouse Administrator generated code into SAS catalogs located on other UNIX processors. We then issued our own cron commands and wrote a small set of utilities for managing job control. We did not invest a lot of time in this as the SAS/Warehouse Administrator development team recognizes the deficiencies of the job scheduler and has plans to remedy them as soon as possible.

- Disk Space

  There is a joke that 'you can not be too rich or have too much disk space'. It is true. When we did our first capacity planning estimates, we figured the size of our major tables, factored in some growth and some padding. We did not realize that work files, summary files, multiple dimension data bases (MDDBs) and other warehouse exploitation tools can take as much space as the data warehouse tables upon which they are based. Luckily, we watched our disk usage carefully and requested additional space before it became a serious problem.

- The temptations to circumvent SAS/Warehouse Administrator were great. It would have been relatively easy to write and then schedule jobs outside of SAS/Warehouse Administrator. But once this was done we would have lost the concept of a central repository of metadata. For a data warehouse to have long term maintainability, all the metadata should be under its umbrella. This provides one location for people to search for information about the data warehouse. Another plus is that the SAS/Warehouse Administrator developers are planning metadata exploitation tools, such as an API that will make our lives easier in the future.

- As we examined the loaded data warehouse tables, we noticed that our data scrubbing was not as complete as we had hoped. This only becomes apparent if you perform a proc freq against a table or if you attempted to use the data in a meaningful manner. In an ideal world these issues would have been resolved at the design stage but in reality many data scrubbing procedures are not written until necessary. This was our experience.

- We did not have a dedicated processor to do our data warehouse loading. At first we had to share a 50MHz UNIX workstation with the UNIX technical support department. When they used this processor, it was for testing purposes, but if they were testing and we were building then someone was going to suffer. Usually that was the data warehouse because, after all, we were the guest. After having loads fail for a lack of space and having loads run for hours longer than we had anticipated, we decided to search for a new home. We were able to find a 75MHz processor that we were able to upgrade to 128 Mb of memory. This helped our load time greatly. The down side was that we still share

the processor with another department. My recommendation to others building a data warehouse is to get dedicated processors with as much memory as you can from the very start.

Maintaining the Data Warehouse

Was our project complete? Not even close! We now had to address the issue of data warehouse maintenance.

One ongoing data warehouse maintenance issue is additional operational data sources and additional columns. No matter how much you may think you have identified all the inputs that you need, someone will identify a missing piece of data. If this is operational data from an existing defined data source or an entirely new data source, SAS/ Warehouse Administrator makes adjustment relatively painlessly. The same is true of a new data warehouse detail table or column. The need for a new column sometimes comes about when fields need to be combined or broken down into different fields to make exploitation easier. For example, we have several date fields that we disassembled into individual month/day/years fields. This need may not be apparent from the start but evolves through usage.

Another area of maintenance comes about as enhancements and bug fixes appear for your software, in our case SAS/Warehouse Administrator. In order to take advantage of enhancements, we had to examine what we had done, and in some cases, do it over. If you are using well established software this may not be the problem it was for us, but it is a consideration.

Finally, as your knowledge of your software toolset increases, you will want to reconsider your earlier work. Our knowledge of SAS/Warehouse Administrator was an obvious area of concern for us, but Release 6.12 of the SAS System was still unreleased at the time we were doing our initial work. We stumbled into many features that helped us, and sometimes forced us to examine prior work.

Exploiting the Data Warehouse

A data warehouse without a user interface is like a building without doors or windows. All of the reading and preparation we did for the data warehouse has no value if people cannot use it to solve problems or analyze business. We spent as much time and effort on the user interface as we did actually building the data warehouse. SAS/Warehouse Administrator is an excellent tool for building and maintaining a data warehouse, but it has no pretensions about being a user tool. It is strictly a tool for the warehouse administrator. This may change in the future, but for now we had to build our own warehouse exploitation tools.

We started by gathering ideas from data warehousing books and other sources. One idea was that the user interface should be accessible to several different categories of customers. For the casual user, you should provide a means of viewing reports or graphical objects that your data warehousing loading generated over night. This allows relatively fast response times and all the user has to do is point and click on their selection. For the user who wants to do ad hoc reporting, we provided access to SAS/INSIGHT software, the SAS Query window and SAS/ASSIST software. These products allowed people to access the warehouse detail tables or summary files that we created overnight, and then do custom reporting or querying at their speed to match their needs. Finally, for the people that want to slice and dice multidimensional data, we provided large MDDBs that, although relatively slow in displaying, provide a level of analysis that is difficult to accomplish with standard report formats. We generated these MDDBs overnight during the data warehouse loading process.

Once we had an idea of what we wanted , we decided that SAS/EIS software was the quickest and easiest tool to use.

We were well aware that any GUI we constructed must be able to grow and adapt as the warehouse grew and adapted to the customer needs. It had already become apparent that more subjects would soon be added and the few customers that had seen the prototype interface were already offering suggestions. The exploitation stage also exposed the issues of capacity planning and data scrubbing. For one, we were going to use MDDBs to allow our customers to do analysis in ways that a report or graph simply would not allow. But MDDBs require significant CPU and disk storage resources. Although we were forewarned of these issues, we never understood how significant they were until we processed actual data.

The GUI also exposed a weakness in our data warehouse design. This was especially true for our time variables. Our data warehouse had several SAS date format fields used by the GUI but sometimes we wanted only a component of a field, for example, the year or the month. Now, we can let the user interface calculate these values every time they are needed or, as we decided to do, calculate them at data warehouse load time so that they are available to all exploitation applications. We believe that this

is an area that a Star Schema would have handled better than the relational model that we used.

Certainly there are always last minute changes in any project. Just as we had 'frozen' our data warehouse user interface and were content to slowly build on our one subject data warehouse, something new appeared. SAS Institute purchased a new phone switch! This new device was capable of providing 'cradle to grave' phone call logging. This was the kind of information that the technical support managers had been yearning for. The new switch was to be installed over the Christmas holiday and everyone wanted its data included in the data warehouse as soon as possible. We are working on this as this paper is written.

Conclusion

Even though we have not had time to reflect on the lessons learned from a one subject data warehouse, we believe that it is worth the time and effort to attempt to take this slow and evolutionary approach. We can guarantee that there will be delays, unforeseen problems, and unusual demands in the construction of your data warehouse. The secret to success is to assume that things will go wrong before they do, have contingencies plans in place, and make quick implementation changes as the environment changes. In other words, be flexible.

Acknowledgments

The following staff members of SAS Technical Support contributed to the building of the data warehouse, the building of the user interface and the preparation of this paper:

Art Alexander, David Driggs, Yvonne Selby, Greg Cooper, Tina Hobbs, Chris Noto, Jake Jacobs, Joy Reel, Jon Schiltz, Danny Hamrick, Jason Moore, Phil Gibbs, Dorothy Proulx.

Martin P. Bourque

SAS Institute Inc.
Cary, NC 27713
sasmpb@unx.sas.com