103-30

Simplify Existing Projects with Ideas from Data Warehousing

Rick Aster

Data Warehouses and Project Objectives

What is a data warehouse? The definitions you find might emphasize such qualities as:

- large volume of data requiring compression techniques for performance
- comprehensive and homogenized data collected from several sources
- operating on separate hardware from transaction databases
- containing historical data to allow a long-term view
- requiring a minimum of technical knowledge to use

These suggested qualities of a data warehouse describe how the focus and direction of data warehousing differs from that of a traditional operations database. Ideally, an operations database contains data necessary to conduct the day's business transactions accurately and efficiently. The ideal data warehouse is quite different. The fundamental objective of any data warehouse project is bring together available data related to a selected subject and organize it to make it easy to use — easy to retrieve, summarize, and analyze. It is this objective that defines a data warehouse.

But the same objective could describe, at least in part, almost any project in business intelligence, decision support, or research. In short, the objectives of the typical SAS project have elements in common with the objectives of data warehousing. A typical ad hoc reporting project brings data together, organizes it, and uses it to generate a set of reports. The same could be said of a production system that does regular monthly performance metrics or one that identifies possible trouble spots in operations, or of the statistical study that comes at the end of a research project.

If the objectives of data warehouses are similar to the objectives of other projects, you might ask whether data warehousing's methodologies and ideas could be useful in everyday SAS projects. To a significant extent, they can be.

Applying Data Warehousing Ideas to Project Data

The ambitious objectives of a data warehouse require you to approach its data in a systematic way. When you intend to assemble the data of a data warehouse quickly and routinely, you can't afford the mishaps and extra work that a haphazard approach would create. It is this systematic approach to data that makes data warehouses so much more productive, and by using the same approach elsewhere you can improve the productivity of almost any project. After you acquire data in a systematic and streamlined way, all the subsequent stages of a project, such as analysis and reporting, tend to go more smoothly. To show how this works, I will consider these three key ideas from data warehousing:

1. Pulling data together
2. Using data correctly
3. Getting consistent results

I will show how these ideas can be useful in other kinds of projects where they traditionally might not be considered.

1. Pulling Data Together

Almost any SAS project starts with data of some kind. In a traditional project, data may be pulled in and manipulated at any point along the way; the programmer may mix together the program logic that acquires the data with the program logic that generates the desired results from it in any way that seems easy and efficient. Data warehousing takes a different approach. Obtaining the data is planned as a separate up-front process.

The data warehouse approach: Collect, clean, and organize data as a separate process before you start to analyze it.

If you apply this idea to a project that is contained in a single program, this would mean that the data is pulled together in the first part of the program. Whatever manipulations are necessary to prepare the data are done in this preparatory phase, rather than being spread throughout the program. This approach can make the project easier to manage. You only need to look at the first part of the program to see what is happening to the data. Also, if you decide to change the source of the data being used or details of the way it is handled, you can make those changes in just one place. The later parts of the program do not have to be changed.

In a larger project, a separate program or several of them could be used to acquire the data for the project. Often, in keeping with the data warehouse approach, the programs that acquire the data for a project can be run only once even if other stages of the project are revised many times.

Data Cleaning

This idea of preparing the data as a separate upfront process applies especially to any data cleaning that a project requires. What is data cleaning? This term applies to any changes that make data values represent more directly the information they provide. There are many variations of the idea of data cleaning, but I will provide one example.

Suppose a set of data is collected in a way that allows only a two-digit number for a person's age. An age of 99 is entered for anyone whose age is 99 or greater. Thus, a value of 99 for age does not indicate a person's actual age, but it does provide some information about their age.

One approach to this problem is simply to note it in any output. This statement, for example, might be used to add a footnote to a report or graph to clarify the interpretation of age values:

```
FOOTNOTE1 'AGE=99 INCLUDES ALL AGES 99 AND UP';
```

This provides information that can help the recipient of the data to decide how it might best be interpreted, but it does not improve the representation of the data itself.

To make the data meaningful in a statistical sense, you may wish to substitute an estimate of the partially known value. Suppose, for example, you find that the average age of people 99 and up in the population you are studying is 104. You could clean the age data with this kind of logic:

```
IF AGE = 99 THEN DO;
   AGE = 104;
   AGETYPE = 'R';
   END;
```

These statements would be included at the appropriate point in the part of the program that pulls the data together. In this example, AGETYPE is an indicator that describes where the value for age comes from, with the value R indicating a value estimated from a range provided. A format for AGETYPE might be provided as:

```
PROC FORMAT;
VALUE $ETYPE 'A' = 'Actual'
  'E' = 'Estimated'
```

```
    'R' = 'Estimated from range'
    'X' = 'Unknown';
  RUN;
```

The format is used to display the descriptive text of the data item in place of the code values that are actually stored with the data. Formats generally are necessary to display a data element in its most meaningful or recognizable form and may be associated with variables in FORMAT statements, for example:

```
FORMAT AGETYPE $ETYPE.;
```

The simplest kinds of data cleaning can be done entirely with informats. Value informats substitute one value for another in the source data. This is an example of SAS code to create a value informat (*Professional SAS Programming Shortcuts*, p. 164):

```
PROC FORMAT;
INVALUE MISN
    -9, -99 = .
    OTHER = _SAME_
    ;
RUN;
```

In some fields, the values −9 and −99 are code values that indicate missing values, and the informat MISN created in this example converts these code values to SAS missing values while leaving all other values unchanged.

The kinds of data cleaning mentioned here have an advantage that might not be obvious at first, but is an everyday occurrence in data warehousing. Substituting the best available value makes it possible to combine this data with other data that does not have the same data anomalies. Returning to the age example, you could combine the age data with another set of data in which actual age values are provided, data that has a different set of anomalies, or even with data in which all age values are derived from demographic classifications that indicate age, sex, and other variables. After data has been cleaned, no extra effort is involved in combining it.

Sources

Data cleaning can scarcely be discussed without mentioning a closely related idea, that you may be able to select the data you use as a starting point. This is another idea that may be helpful in ad hoc reporting and similar projects but is not always considered. The designers of a data warehouse look for the best available data sources for the data warehouse. By contrast, in small projects, the data source for the project may be assumed as a starting point; it may be indicated as part of the description of the desired outcome, even though the data source is, strictly speaking, not an outcome, but a resource. Reviewing the available data sources might lead to a better source for a particular need or a combination of sources that provides a more complete or accurate look at the subject.

The data warehouse approach: Obtain information from the best available sources. Select all the data that matters. Leave out data that doesn't matter.

After data sources are selected, a common impulse is to acquire all the data that is available from each source. However, it is more efficient to select only the specific data elements or general categories that you expect to have a bearing on the subject area of the inquiry. For example, transaction systems often provide detailed information about the sequence of actions by which a transaction came about. These details may not matter in a retrospective analysis, so they may be left out when the data is acquired.

2. Using Data Correctly

Data cleaning goes a long way toward ensuring that data will be used correctly, but a well-designed data warehouse goes beyond this. A data warehouse uses additional variables and documents to describe the meaning of data elements, values, and records. This additional data is called metadata because it describes the data being used.

The data warehouse approach: Use metadata to describe the meaning and source of data.

The variable AGETYPE in the previous example is one kind of metadata: an added data element that provides information about the source and quality of another data element. You might also have a code variable to indicate the source of an observation. This is especially useful when data is combined from multiple sources that have different approaches to collecting or delivering data. A metadata element may also indicate the time when an observation was recorded or reviewed, or it could tell the level of review that was applied.

Equally important is the metadata that provides descriptions of each data element and the observations in each table. A definition of AGE, for example, might say:

> The person's age in years, as indicated by the person in the initial interview

The definition of a table should indicate the meaning of records in the table, for example:

> The table contains one record for each person interviewed.

Such information about data is usually well-known among the people who work with the data regularly. In their original context, these details might seem too obvious to mention. However, when data is taken a short distance away and used for a slightly different purpose, it is surprising how quickly this level of understanding of the data can be lost. Metadata provides an easy way to overcome this difficulty.

Even in the immediate original surroundings of a set of data, it can save work to have metadata systematically collected. Information available as metadata stored in data and documents is less expensive in the long run than background information that individual programmers and analysts have to carry around in their heads.

3. Getting Consistent Results

It is often hard for programmers to explain to business managers why different projects seem to yield such different pictures of the same business. Managers tend to be suspicious of the possibility that an error has been made, and programmers may have to explain in painstaking detail the varying assumptions that lead to the different results. For example, the accounting department treats Wheeling, West Virginia, as part of Ohio, but the marketing department includes it in the Pittsburgh region; sales tax may be collected without a sales transaction being counted if the original sale was refunded in full; operations people may record projects as occurring in later time periods than sales people do.

Getting consistent results is one of the most commonly stated objectives of data warehousing. It is not that a data warehouse locks people into one single view of data, but the differences among the different views are clear and well explained, and it is easy to return to the same view of data when you want to make one report or analysis consistent with a previous one.

The data warehouse approach: Use a consistent view of data to get consistent, reliable answers.

The traditional ad hoc reporting project takes the opposite approach. There are no clearly defined views of data, and ad hoc data changes, not necessarily documented, may be made at any stage along the way. You might find the same variable name used to mean several slightly different things, even within the same program, leading to the various kinds of confusion mentioned earlier. In addition to this concern, a further complication is created if you want to make two ad hoc reports consistent with each other. All the ad hoc data changes used in the earlier report must be duplicated in the code that creates the newer report, and any additional data changes required for the newer report must be retro-fitted to the earlier report. If there are more than two such reports, the effort of keeping them consistent is substantially greater, as each report must be compared to each of the others. A data warehouse system sidesteps this entire problem by making the data consistent at the outset, and then by storing descriptions of the data elements so that they can be used correctly. After this is done, you can obtain consistent results just by selecting the same data elements.

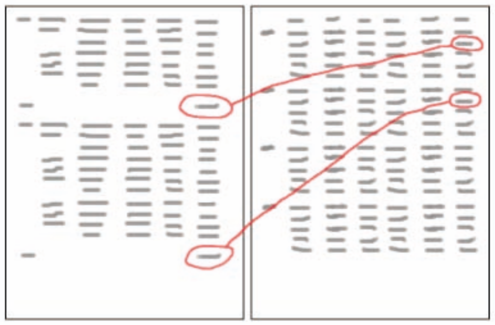
Suppose that the code below generates a periodic transaction report. Note that this code does not include any logic to transform, recode, or otherwise alter the data it reads from the SAS dataset CORP.YPAY.

```
PROC REPORT DATA=CORP.YPAY (WHERE=(MONTH=&THISMONTH
    AND CUSTOMERSTATUS = 'N')) NOWD;
   COLUMN SEGMENT CUSTOMER ACCOUNT BALANCE EVENT AMOUNT;
   DEFINE SEGMENT / ORDER;
   DEFINE CUSTOMER / ORDER;
   DEFINE ACCOUNT / ORDER;
   BREAK AFTER A / SUMMARIZE OL;
RUN;
```

A new program, using the code below, generates a trend report from the same data.

```
PROC TABULATE DATA=CORP.YPAY;
   CLASS MONTH SEGMENT CUSTOMERSTATUS;
   VAR AMOUNT;
   TABLE N AMOUNT, CUSTOMERSTATUS*SEGMENT, MONTH;
RUN;
```

This code starts from the same SAS dataset and also does not transform the data it uses. This results in consistent output. You can compare the two reports and find some of the same numbers in the output, as shown in the diagram below.



This kind of agreement between two reports, happening automatically in the sense that no additional programming logic is required to create it, is one of the most visible benefits of the data warehousing approach. For the users of the reports, the value added comes from being able to use each report to help understand the others, without any worries about the possibility of divergences between the way the report programs handle the data they use. This is possible because of the separate data preparation phase that has already taken place; this is a requirement when creating a data warehouse, but it is just as easily done and just as valuable in an ad hoc reporting project.

Adding the Data Warehouse Approach

Adding the data warehousing approach to an existing project can start by getting to know the source of the data used in the project. In some cases, this is a simple matter of writing down and formalizing knowledge that has already been acquired in an informal way. It could be a more substantial undertaking if the understanding of the source data is more distant and tenuous, but then the likely rewards include the ability to find and correct misuses of the data that have found their way into the process. After the source data is properly documented, you might take a moment to consider whether a better or auxiliary source for the data might be available, or whether some of the source data might be omitted.

The next priority is to separate the data handling into phases. If data cleaning or other transformations have been coded as part of the reporting or output code, move that logic to a separate data cleaning phase. This might be a simple matter of cutting and pasting to move statements from one data step to another, but it will often require further analysis and design changes in order to make consistent use of each data element — that is, to have one variable mean only one thing.

The code that results from this effort is much easier to maintain than traditional project code. New analysis and reporting can be added easily because the data is already prepared. If changes are required in the way data is cleaned or interpreted, these can be made in just one place — it is no longer necessary to examine every program in the project to implement corrections or changes in the data. The later parts of the new code may also be warehouse-ready; if a data warehouse is subsequently constructed, these programs could probably be used with it with minimal changes.

The Accidental Data Warehouse

If you follow these ideas from the small beginnings of project, it may eventually grow and evolve into what is virtually a data warehouse. That is, just by applying the essential ideas about data as it is used in data warehousing, you could create a data warehouse almost by accident. Of course, there is more to a data warehouse than this, and I do not mean to minimize the importance of such things as scheduled data loads, security, performance tuning, formal view definitions, usage metrics, and issues related to scale and detail in data warehousing. Nevertheless, the most difficult aspect of data warehousing is data design, and this is easier to do one detail at a time than it is to try to do it all at once. Indeed, the only cause of data warehousing failures in my experience has been that many of the small details of the data have been neglected. Creating a data warehouse is primarily a project of sorting out all the intricate details to be found in the data elements and sources.

Just by approaching data with the thought in mind that you will find the best data, pull it together, make it consistent, document it, and use it in a consistent way, you will have gone a long way toward the data design that could create a data warehouse. And even if a data warehouse is not your objective, you can still get benefits of data warehousing in bits and pieces, by putting ideas from data warehousing to work in smaller everyday projects.

References

Rick Aster. *Professional SAS Programming Shortcuts*, 2nd edition, 2005, Breakfast Communications Corporation.

—. *Professional SAS Programming Logic*, 2000, Breakfast Communications Corporation.

Ron Cody. *Cody's Data Cleaning Techniques Using SAS Software*, 1999, SAS Publishing.

Alejandro Gutiérrez, Adriana Marotta. "An Overview of Data Warehouse Design Approaches and Techniques," white paper, 2000, Instituto de Computación, Universidad de la República, Montevideo, Uruguay.

Contact

Rick Aster, Breakfast Communications Corporation, P.O. Box 176, Paoli, PA 19301-0176 U.S.A.;
http://www.globalstatements.com; ra@globalstatements.com