SAS® as a Means to a Different and Better End

David J. Ross
Maria Lupetin
Lupetin Consultants, InfoMaker Inc.

Introduction

The perception of SAS in corporations and institutions has frequently been that of a list of products outside of the core Information Services (IS) community. Often, pockets of dedicated users tout SAS as their solution for understanding and presenting results. To some extent, SAS Institute has changed this perception by promoting itself as a complete information delivery system. This change has piqued the interests of IS members, which have taken a fresh look at SAS as an alternative to their present systems.

The greatest obstacle is the fact that companies and institutions have a vested interest in knowledge already in place. SAS may not be part of this chosen platform. Existing resources and staff will assume responsibility for support, maintenance, and enhancement of systems. Furthermore, there is a requirement that any solution be part of the larger picture. IS focused on where development is today and where it is going and not the power of an integrated solution. Costs incurred are measured in times of limited budgets, right sizing, and the migration to client server solutions. Consequently, it is an uphill climb for SAS to find wider acceptance.

The approach we follow in developing and delivering complete information systems to clients is to use SAS as a toolbox of powerful components to prototype complex systems. Our goal is to use SAS as the means to understand the information and model the relationships, entities, and attributes of a system. Data is loaded into SAS files using a variety of methods, arranged for use, collapsed and redefined for better understanding, presented in simple audit reports, and modeled for usefulness and statistical significance.

Once we've "got it right", we translate SAS programs and procedures into clear SQL and 4GL statements, screens, and reports of a relational database management system. Thus, the different and better end for the client is a product that suits their specific needs and fits into their strategic position of supportable products.

A case study is presented based upon Lupetin Consultants participation in Federal Grant #4192 ("Violence Reduction in Urban Areas") which was administered by the Illinois Criminal Justice Information Authority and implemented by the City of Chicago to reduce domestic violence. Using SAS Base under OS/2®, we were able to comprehensively construct all of the elements needed for a relational application. This resulted in the eventual development of the DVIS® Domestic Violence Intervention System, using INFORMIX® under SCO UNIX® as the eventual outcome system.

Building a Relational Model for SAS

The development of a relational model of database management with SAS as the starting point requires the following helpful guidelines. First, it is a simple linguistic change, using terms and phrases from the nomenclature of relational tomes. Next, a concentrated effort need be made to utilize the elements which are common in all data management approaches. Finally, a conversion process is needed to think of things differently.

Someday SAS may publish a SAS to Relational phrase dictionary for the uninhibited or timid. Until then, the use of a good SQL book and the SAS Guide to the SQL Procedure: Usage and Reference will suffice. The elements needed in this process are established by the following principles:
Physical data independence moves away from the DATA step and into the use of PROCs.

Logical independence enforces the relationship between tables (SAS data sets), columns (variables), and rows (observations).

Information manipulation, definition, and administration focus on the realm of nonprocedural tasks for selection (WHERE), projection (V AR listing), and joining (MERGE).

Structured views of information for retrieval or update restrict data.

Data is arranged and summarized using available methods to correctly understand the relationships.

SQL is used when appropriate, resulting in the smooth transition to the final application.

Information is presented clearly to draw conclusions and model the final approach.

SAS to Relational translation occurs to create the better end.

Data Conversion and Interface

Most projects involve the use or export of information previously collected. Consequently, some form of interface will be required for the prototype and possibly for the final processing of the system. Interfaces can best be described as "Simple things taking LOTS of time to get right". In part, this is due to each operating system imposing a set of rules which are most efficient for the world it deals with. Furthermore, the proliferation of PC and workstation software with their own optimized structure has often expanded this confusion.

Don't spend lots of time getting the data in. Either the creation of a fixed field or common delimited file will suffice for data transfer. The available methods in SAS to perform such tasks are legion: use them to capture information from an "uncontrollable" source: i.e., whatever form the data is sent to you. Some of the more common examples of these are:

**ASCII Source**

**INPUT Statements:**
- Header Detail with Flagged Record Types (mailing List with detailed information)
- Column Specific (keypunch data)
- Common Delimited (ASCII dump from word processor or spreadsheet)

**Binary Files**
- Formats for Native Mode translation: IB, PD, PIB, RM
  - Formats for 370 Mode (S370FIB, FPD, FRB, PIB)

**Direct Database or Spreadsheet Load**
- PROC ACCESS for data views to DBF, DIF, or DB2.

The last item involves the purchase of the SAS/ACCESS software. When the applications are clear cut and static, SAS/ACCESS is a useful addition to a developers toolbox. However, when the structure of the information is dynamic (changing regularly) or complicated (in terms for formats, indexes, and links) the amount of work and expertise becomes burdensome. Most soft packages have a direct or intermediate step to a common file structure which can load into SAS files, making ACCESS unnecessary.
Arrange and Summarize Data

The data in SAS Data Files for analysis follows a many variable and many observations structure. For example, 4 variables and 2 observations is represented as:

<table>
<thead>
<tr>
<th>PT_ID</th>
<th>Q1</th>
<th>Q2</th>
<th>Q4</th>
<th>Q7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>34</td>
<td>23400</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>37</td>
<td>27060</td>
</tr>
</tbody>
</table>

This structure lends itself to performing a PROC FREQ for TABLE Q1 * Q7 query.

Unfortunately, what works best here violates the relational rules of data normalization. We are able to transpose data so that data becomes normalized into its elementary components. The output of a PROC TRANSPOSE results in the following:

<table>
<thead>
<tr>
<th>PT_ID</th>
<th>COL_QUE</th>
<th>RESPONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>34</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>23400</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>37</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>27060</td>
</tr>
</tbody>
</table>

Consequently, a useful approach is to keep the data in the SAS form for analysis and periodically transposing data to verify that the rules of normalization still apply.

Summarization can result is smaller, more manageable groupings of information. Although the _TYPE_ variable is one of the power elements in SAS, perform summarization at the NWAY level. This limitation results in clearer groupings of data and is directly translatable into relational models of information.

Use SQL When Appropriate

Structured Query Language (SQL) is a powerful tool which has gained wide acceptance in the commercial data management community. Many products have followed the ISO-ANSI standard while adding unique extensions which optimize their own structural elements. In this regard, SAS is different, with a number of missing elements from the language. For example, permissions (SQL GRANT and REVOKE), entity and referential integrity, transaction control, concurrency, and performance are not part of the PROC SQL implementation.

All of these fall under the data administration. Since we are using SAS to prototype the eventual system, these shortcomings can be addressed at the time that the SAS to Relational database translation occurs. If they need to be addressed in the SAS prototype, then password protection of files, SAS/SHARE for locking, and SAS/CONNECT for performance should be investigated and implemented.

Our goal is to prepare the prototype for a smooth translation into the final application. To do so, we should think of things differently, using PROCs instead of DATA steps and the linguistic equivalence of separate PROCs within the terms of SQL. A short but far from exhaustive list of examples are:

- SELECT columns replaces KEEP and DROP of variables.
JOIN and DISTINCT perform basic SAS MERGE steps.

ALTER TABLE and SELECT column AS instead of RENAME, LENGTH, and ATTRIB.

UPDATE rows to change values through computations.

ORDER BY and CREATE INDEX for SORT, WHERE, and BY statements. These improve performance and reduce the need for additional steps.

Perform summary values, statistics, and tallying in the SELECT list or GROUP BY.

Present selections using PMENU features for screen simulations.

With a list of things to use, experience suggests a number of items to avoid:

- Data step MERGE: it is often faster and easier to implement, but difficult to explain or translate.
- Implied JOIN which creates a Cartesian result of immense size.
- Combining 3 or more files in SQL: use a SET instead.
- Small disks. Multiple CREATE statements result in additional WORK files.

Report Information Clearly

In using SAS as a toolbox of powerful components, we can gain a clear understanding of the information and model the relationships, entities, and attributes which will form the final system. Every project will lend itself to a different approach, although a number of the suggestions generally apply. These are:

- Look at different collapsing and redefinitions of data for usefulness.
- Evaluate missing data for meaningful information gathered.
- Create lists of observations using PROC PRINT.
- Summarize and tabulate results clearly using PROC REPORT.
- Use PUT statements as a last resort because of problems of translation.
- Use the tools of PROC FREQ and PROC TABULATE for respondent level of analysis.
- Use PROC SUMMARY, and PROC UNIVARIATE to model the aggregate levels of analysis.
- Use statistical analysis options for determining importance and guiding conclusions.
- Draw conclusions and model the final approach.

From this and other approaches, we can structure components for eventual translation. Even complex results obtained from a PROC UNIVARIATE can be similarly computed in the 4GL of a relational product or through one of the many available calling routines to programming languages such as "C", BASIC, or FORTRAN.

Translate

The final step in the process is to perform a translation from SAS to Relational. If we have followed our goal well, we have developed a powerful group of components in the development of the prototype of a complex system. By thinking in terms of relational phrases, using elements which are common between SAS and relational database management systems, and thinking in
terms of PROCs instead of Data Steps, we can now smoothly translate our code and approaches.

The following is a partial list of things which guide the translations:

<table>
<thead>
<tr>
<th>SAS Tasks</th>
<th>Relational Database System</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUT, formats, informats</td>
<td>DBLOAD</td>
</tr>
<tr>
<td>RENAME</td>
<td>SELECT xx AS</td>
</tr>
<tr>
<td>MERGE</td>
<td>JOIN &amp; DISTINCT</td>
</tr>
<tr>
<td>LENGTH &amp; ATTRIB</td>
<td>ALTER TABLE</td>
</tr>
<tr>
<td>TRANSPOSE</td>
<td>Normalization</td>
</tr>
<tr>
<td>SQL</td>
<td>SQL and 4GL</td>
</tr>
<tr>
<td>PRINT, REPORT</td>
<td>FORMS Generator</td>
</tr>
<tr>
<td>FREQ, TABULATE, INPUT</td>
<td>4GL Statements</td>
</tr>
<tr>
<td>SUMMARY, UNIVARIATE</td>
<td>GROUP, AVG, etc.</td>
</tr>
<tr>
<td>PMENU</td>
<td>SCREENS Generator</td>
</tr>
<tr>
<td>Statistical Results</td>
<td>4GL or FUNCTION</td>
</tr>
</tbody>
</table>

When we translate SAS programs and procedures into clear SQL and 4GL statements, screens, and reports of a relational database management system, our work is done.

Thus, the different and better end for the client is a product that suits their specific needs and fits into their strategic position of supportable products.

References


About the Authors

David J. Ross is Senior Consultant at InfoMaker Inc. He has extensive experience in the design, development, programming and delivery of enterprise systems. He has implemented solutions in healthcare, human resources and criminal justice applications. Mr. Ross has a M.A. in Sociology from Loyola University of Chicago.

Maria Lupetin is president of InfoMaker Inc. InfoMaker specializes in computer system development, relational database and Client/Server technology. InfoMaker clients include Fortune 100 firms and government agencies.

Maria's expertise is in the development of mathematical and statistical models for manufacturing, distribution and marketing applications. Prior to founding InfoMaker Inc., she

652
has held various management position at Morton International Inc. Ms. Lupetin has an M.S. in Operations Research from the University of Minnesota, and an MBA from the University of Chicago.

The authors welcome questions and comments.

Maria Lupetin, President
David J. Ross, Senior Consultant
InfoMaker Inc.
5901 N. Cicero Avenue
Executive Towers Suite 103
Chicago, IL 60646-5611
Phone 312.736.4059