Decision Support - A Quick Turnaround System Using SAS

Tony Brown and Ed Forman
Rapp Collins Worldwide, Dallas, Texas

Abstract

This paper describes some of the design and technical processes and steps Rapp Collins uses in rapidly creating a marketing decision support system using the SAS® System. These systems are designed to turn large volumes of legacy data into useful information. It includes the usage of SAS® and SYBASE® as data warehouse repositories, and the creation of SAS AF/FRAME® user interfaces. The paper includes specifics on data aggregation, shaping, and indexing for decision support. It also shows coding aspects of some of the SAS AF/FRAME/SCL® drill-down mechanisms used to navigate the warehouse. Client-server efficiency, maintenance, and operations considerations are addressed as well.

SAS® Software is installed on IBM 3090 Mainframes under DOS/VSE®, UNIX HP900 platforms under HP/UX®, and PC's using Windows NT and Windows 3.11. In addition to the BASE® product, the SAS® modules used include SAS/AF®, SAS/GRAPH®, SAS/STAT®, SAS/CONNECT®, SAS/ACCESS®, SAS/ASSIST®, and SAS/EIS®.

Business Case

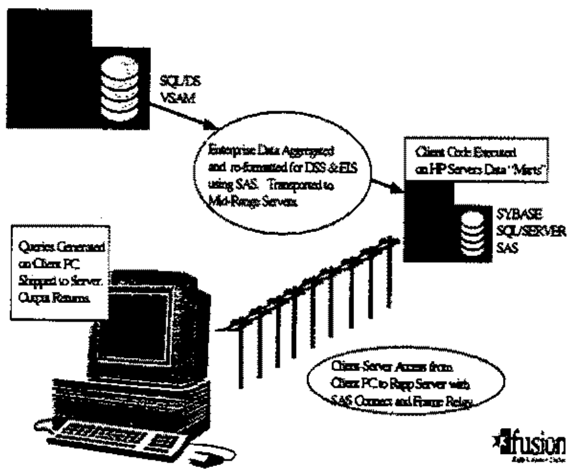
Direct Marketing clients require the ability to extract meaningful information from extremely large volumes of marketing data. This includes customer, transactional, and campaign performance related data residing in legacy databases often consisting of 100 to 300 million rows of data. Our clients need performance reporting on their marketing campaigns, ad-hoc and production level reporting, as well as diverse applications of statistical and analytical analysis.

The SAS® System was chosen because of it's ability to access numerous data sources on multiple platforms, turn that data into enterprise information, and easily allow accessibility. Cost efficiency is necessary, as is maintainability, and short development and implementation cycles. In addition, the systems must be deployable to clients with multiple platforms using dissimilar operating systems - quickly! Our systems are created and implemented within 6 to 8 weeks on average.

Overall System Architecture

The basic architecture of our applications is as follows:

Decision Support Architecture


SQL/DS
VSAM
Enterprise Data Aggregated and re-formatted for DSS & EIS using SAS. Transported to Mid-Range Servers
Client Code Executed on HP Servers Data "Marts"
SYBASE
SQL/SERVER
SAS
Queries Generated on Client PC Shipped to Server. Output Returns
Client-Server Access from Client PC to Rapp Server with SAS Connect and Frame Relay
fusion

SAS /CONNECT® is used at the startup of the PC based user application (client) to initiate a server session on an HP or NT Server. Client machines are located in client business sites, and the servers are located in our corporate data center. Queries, analyses, and report requests are assembled via SAS/SCL® code constructed by user inputs (clicking choices of objects - push buttons, selection lists, user entry fields, etc.). Once the user selects the "RUN" button the query is submitted via an SAS/SCL® submit block to be executed on the server. The primary processing takes place on the server, and the answer set (graph, file, or report) is shipped back to the user in interactive mode. For batch submits the answer set remains on the server, and a notification message is sent to the user to download the answer set at his/her convenience.

Data Warehouse Construction Issues

The effectiveness, maintainability, and performance in any decision support system relies heavily on it's data warehouse. The physical hardware, software, and data architecture play paramount roles in Effective data access. The primary issue involved at Rapp Collins is turning the huge volumes of

Mainframe VSAM and SQL/DS® legacy data into an accessible data warehouse for analysis and reporting.

In most cases, the mainframe systems serve as the data provider for daily, weekly, or monthly updates. Data is off-loaded from the mainframes and cycled through aggregation routines and de-normalization techniques. Where appropriate, SAS® PROCS are used (SUMMARY and MEANS) to aggregate the data when the resulting data is to reside in a SAS® dataset. Often times unit level data is required on the decision support system and SQL Server® tables are used as opposed to SAS® datasets. Most of our decision support data warehouses contain a combination of both SAS® datasets and SQL Server® tables.

Data Marts

Each of our decision support data warehouses are composed of many different 'Data Marts'. Each individual Data Mart supports a unique analytical need. For example, one Data Mart might support a promotional analysis component allowing the user to perform response analysis for select direct mail promotions. They can answer questions such as: "What was the overall response rate to the promotion and my return on investment?", "Is there a significant difference, statistically speaking, between the treatment and control universe for this promotion?", and "Did this promotion creating a sustaining lift in customer purchase behavior or just a spike that leveled off after the promotion was over?"

For comparison sake, another Data Mart might support correspondence analysis allowing the user to track over time the reasons customers correspond with the business. This type of analysis provides insight into how the program is perceived by the customer allowing the business to react to the customers needs and provide better service.

In each of the above examples, two different major analysis areas are supported by two different custom Data Marts. Each Data Mart contains only the necessary data elements to support it's functional component. Therefore, the creation of unique Data Marts is driven by the different analytical needs of the user community. A few additional examples of Data Marts are Transaction History, Survey Analysis, Member Demographics, and Member Lookups.

SAS® Datasets vs. SQL

Two key factors are used when deciding whether to store a Data Mart in SAS® datasets vs. SQL tables.

1. Can the data be aggregated or is unit level detail required?
2. Which method produces the fastest query response time?

If the analytical need is to produce parameter driven summary reports, aggregated datasets are used. Aggregates require less storage and provide fast run times simply because the queries are less I/O intensive. These aggregates tend to have short record lengths containing only the fields required to support the analysis.

SQL Tables are used when unit level data is required. Unit level data tends to have much longer record lengths to accommodate fields such as name, title, company, and address. The maintenance of unit level data is more conducive to the SQL environment.

Performance goals for analysis run-times range from 10 to 15 seconds for a quick non-complex analysis running against aggregated datasets and SQL Tables, to as much as 5 minutes for extremely large complex analyses covering wide data populations.

The frame application isolates the end-user from having to decide between SAS® datasets or SQL Tables. The end-user simply sets up the parameters for the analysis, and the application decides whether the query runs against a SAS® aggregate or a SQL table.

Aggregation Techniques

The most common concern regarding aggregating data is how to support ad-hoc analytics using aggregate datasets without loosing the granularity necessary for complex analyses. To address this question, we must explain how we use 'By' variables and 'Analysis' variables.

The aggregates are composed of two types of fields: 'By' variables and 'Analysis' variables. 'By' variables are used in defining criteria for the query. We sometimes refer to these variables as drill down variables because they are used to further define subsets of data that reveal relevant trends. These variables usually appear as variables on a report where the numbers are distributed across the user selected 'By' variables for the analysis. Examples might be state, member tier, status, income range, etc.

'Analysis' variables are sometimes referred to as the 'count-by' variable. This variable represents the field(s) that will be counted or summed to indicate some level of behavior or activity. Examples might be member count, sum of dollars spent, sum of purchases, etc...

When designing an aggregate, the 'By' and 'Analysis' variables needed to support the analysis are decided upon with input from the users. Most importantly, thorough interviews with the users are conducted to determine the types of queries that will be asked. For example, do they need the ability to run by date (daily, monthly, quarterly, or year over year). This information is used to determine the highest level of aggregation possible that will still support the analysis. Once determined, the aggregation process properly orders the 'By' variables and uses the 'By' variables as it's group-by key. The 'Analysis' variables are summed-up across each unique combination of the group-by key.

When properly defined, aggregates can reduce run times because less records are passed to satisfy the query. In one case, an aggregate was created from 7.5 million transaction records. This aggregate contained 1.6 million records and satisfied the transactional queries while passing only a fraction of the total unit level transactional data.

The user has the flexibility of picking unique values as query parameters across any and all of the 'By' variables in the aggregate. This allows the user to build literally hundreds of unique queries. While this method does not provide 100% ad-hoc functionality, we have found that our clients are able to fulfill approximately 80% of there query needs using aggregates.

Indexing Techniques

If a "Data Mart" is composed of a SAS® aggregate dataset , the user interface allows the user to build queries using any combination of 'By' variables available in the aggregate. To support the dynamics of this approach, we build simple indices on each of the 'By' variables in an aggregate, and composite indexes where 'By' variable combinations warrant them.

SQL tables require less analysis to index due to the fact most inquiries on SQL tables are to unit level data that is referenced by a single unique identifier such as an account number. Therefore, in the case of a member record look-up, the only index required would be on member account number. This would facilitate the look-up of unit level member information.

User Interfaces & Data Access - Panel Features to Support Data Access

Once the data warehouse, and client analysis and reporting needs are established, our attention turns to designing the user interfaces. We choose to design user interfaces exclusively in SAS AF/FRAME® for several reasons; ease of use, object oriented support, the ability to build new and composite objects through sub-classing and method customization, the ability to easily assemble and submit SQL and SAS® code to SAS® sessions on local and remote servers using SAS/CONNECT®, etc. Staying in SAS/FRAME allowed us a seamless interface to use SAS/STAT®, SAS/GRAPH®, SAS/CONNECT®, SAS/ACCESS®, and SAS/ASSIST® in our graphical user-based applications. In addition by changing only host operating system specific code lines (less than 4% of code in our case), our entire application is portable to other operating systems using SAS®. We have executed these ports in as little as 30 minutes due to our generic system design.

Rapid Application Design (RAD) techniques are utilized in working with our clients to create the user interface and application characteristics. This iterative design process requires rapid screen prototyping. Sample user panels are created based on initial requirements, with "fixed" output results to simulate working functionality. User comments and input on the panel functioning, appearance, human factors, etc., and their suggestions for change are iterated into the subsequent panel designs until a "firm" design is arrived at.

In correlation with usual panel design considerations suggested by most methodologies, panels need to stay simple, yet highly functional. This will be illustrated in the following paragraphs. Panel designs are intentionally kept "flat" - an application panel usually does not go more than 1 additional sub-panel deep in selection operations. The following panel illustrates a geographic "drill-down" approach to analysis and information mining for a global company.

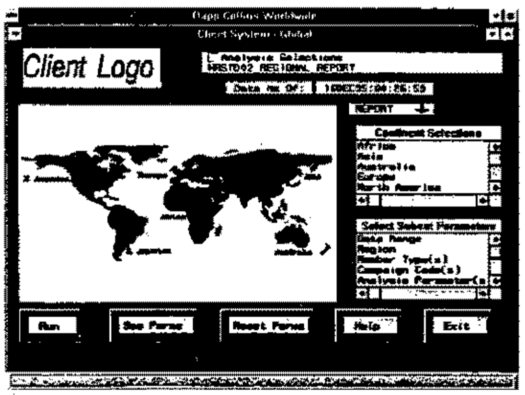


Figure 2

The panel allows the user to run an unlimited array of parameter driven analyses, continuously using more detailed parameters and analyses to "drill down" through the data for answers - from one place. This is made possible by the aggregated Data Marts. Figure 2 presents the user with a world map, subdivided into continental regions. The maps were extracted from the SAS® Maps datasets, and overlaid with hotspots. Hotspots can be precarious to deal with since they only have a few methods available to manipulate them.

Therefore, SCL code is used to monitor the hotspot states when choices are made in the Continent Selection box, or by clicking on the hotspots with a mouse. The following sample code corresponds to Figure 3. The USA Map appears when a user selects North America from the Global Panel, clicks a "Drill Down" arrow for North American countries, and chooses USA. The following discussion addresses how hotspots are used to select values to fill a dynamic SCL list that in turn is used to create subset code in the analysis. Each state (on this map) is designated as a hotspot (green when not selected, red when selected).

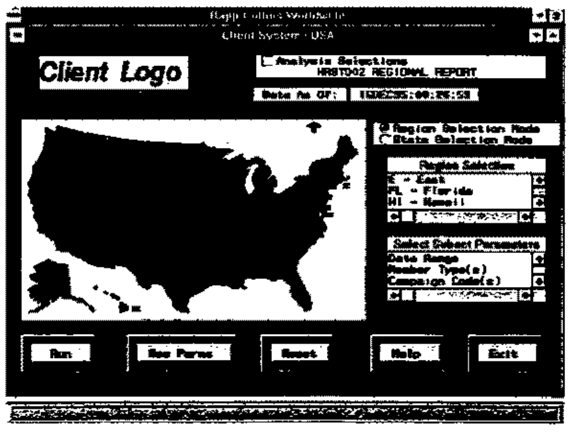


Figure 3

When the user clicks on the state of Washington on the map, the hotspot invokes the HOTWA labeled section, which creates values to pass to the HOTSPOT labeled section. The HOTPSOT labeled section identifies if the hotspot is "off" or "on" by it's color, and acts accordingly to take the state value out of the subset list if this is a "de-selection" by the user, or adding it if it is a "selection".

```
HOTWA:
  hstate='HOTWA';
  state='WA';
  link HOTSPOT;
RETURN;

HOTSPOT:
  if upcase(selmode)='STATE' then do;
    call notify('MAPGRAPH',
   '_GET_HOTSPOT_COLOR_',hstate,hcolor);
   if upcase(hcolor)='GREEN' then do;
      howlong=listlen(statlist);
      if howlong >0 then do;
        pos=searchc(statlist,state,1,1,'Y','N');
        if pos=0 then do;
          statlist=insertc(statlist,state,-1);
         call putlist(statlist);
        end;
      end;
      else statlist=insertc(statlist,state,-1);
    end;
    else if upcase(hcolor)='RED' then do;
       howlong=listlen(statlist);
       if howlong >0 then do;
         pos = searchc(statlist,state,1,1,'Y','N');
         if pos^=0 then
            statelist=delitem(statlist,pos);
       end;
    end;
end;
RETURN;
```

From this panel the user may choose from a large number of batch or interactive analyses to execute for all international sites, or subset the selection to a particular continent. A Drill Down arrow appears on the map when a continent that is sub-dividable into populated countries or regions populated countries is selected. Pressing the Drill Down arrow brings up a pop-up list of the countries, choosing a country produces a subsequent map for that country with hotspots for it's sub-dividable areas (states, provinces, etc.). The drill-down from the panel in Figure 2 to the panel in Figure 3 occurred this as mentioned earlier. This implementation allows the user to begin with as wide a view of the data as possible, and continue restricting it to define narrower geographic data segments.

In addition to the geographic drill down, the Select Subset Parameters box on the right side of the screen provides dates, and variables to further subset the data by. These represent the 'BY' variables mentioned earlier. Selecting any of these will yield a pop-up entry for dates, and extended table selection lists for other discrete parameter values. Only parameters germane to the analysis selected for the panel will be displayed. When other analysis are selected, the Select Subset Parameters List Box choices dynamically change accordingly. SCL Lists and sublists are used to govern parameter captures, and passing. The following simple code illustrates the dynamic updating of the Select Subset Parameters List Box when a new report is chosen:

```
/***** Drill Down Report *****/
If substr(analysis,1,8) in ('HBSTD01') then do;
   rc=clearlist(parmlist);
   parmlist=insertc(parmlist,'Date Range',-1);
   parmlist=insertc(parmlist,'Region',-1);
   etc.
End;
If rc ^=0 then ......
```

When the user selects one of the subset parameters, extended tables or list boxes pop-up with values the user can select from.
An example pop-up extended is shown below:

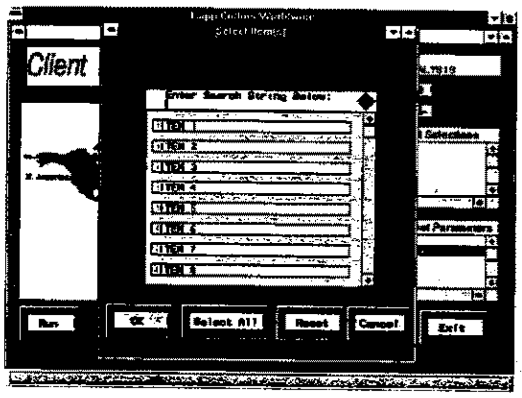


Figure 4

The extended table in Figure 4 is already subset based on previous parameters chosen. SCL lists built to store previous subset choices are passed to each panel if they exist, are examined, and the values they contain are used to help subset the current panel choices. This prevents users from choosing report parameters that are mutually exclusive. Example: A user chooses a Continent of North America, and a Region of South West. When the Items subset is chosen from the Subset Parameters List box, the list box is populated with Items that fall within that Continent and Region. Sample code placed into the SCL INIT section of Figure 3 to subset the dataset feeding the extended table is illustrated below:

```
Submit continue local;
   Data Items;
         length Items $20;
         set lib.Items;
Endsubmit;
/*** Check the Parameter Lists for Subset ***/
howlong=listlen(Contlist); **Cont list check*;
howlong2=listlen(Reglist); **Region list check*;
etc......
If (howlong >0 or howlong2 >0 ........) then do;
   submit continue local;
     where
   endsubmit;
   if howlong > 0 then do;
     submit continue local;
        CONT in (
     endsubmit;
     do I=1 to howlong;
        string=trim(getitemc(contlist,i);

        if I<howlong then do;
           submit continue local;
              "&string5",
           endsubmit;
        end;
        else do;
           submit continue local;
              "&string5"
           endsubmit;
        end;
   end;
   etc............
```

In this manner, each subset Parameter that is chosen, is limited in choice to what already corresponds with previously selected parameters. Therefore, users cannot run reports with conflicting or disjoint parameters.

By Combining broad based, and very narrowly focused analyses in the Analysis Selections check box, users can run general reports, spot trends and immediately "drill down" using more refined analyses and subset parameters of choice.

Note the use of the Search text entry field and direction arrows above the extended table.
This collection of objects can be consolidated into a single composite object. SAS/AF/FRAME® provides the capability to build composite objects (objects composed of other objects), and custom objects to

provide custom functionality. In this case, string searches can be made in extended tables. While discussion of custom objects is beyond the scope of this paper, it must be noted that they provide flexible and reusable objects.

The above panels only represent a minor foray into the realm of possibilities Object Oriented Programming (OOP) techniques available in SAS/FRAME®. The achievement of innovative and highly capable user interfaces is limited only by the creativity and design expertise of the developer.

Maintenance & Operational Considerations

In order to maintain cost efficiency in system maintenance, system updates can be done from the system host server. The application can be coded to query a host table immediately after logon to check if "update flags" are present.

These "update flags" can be maintained in a table for each user (updated by system administrators or batch programs). They can denote when application updates are available, additional client machine datasets need updating, or even if SAS needs a new SETINIT. The value of these flags can be retrieved through a remote process in SCL and automatically trigger update panels (prompting the user to OK an update or wait), which prompt the new catalog, data, etc., download to refresh the system. This entire process can actually be incorporated into the host batch system for total automation.

Many areas addressing maintenance and operational efficiency do not fit into the confines of this paper. But the Decision Support developer should be aware of development library management techniques to govern multiple users working from the same development catalog, building object and code toolboxes, and custom and composite widget toolboxes for rapid development (plug and play). In addition particular concern should be paid to region attachments for Frame regions (they can save big headaches porting applications across operating systems).

Summary

Decision Support Systems are challenging endeavors in terms of creating supporting data warehouses, efficient access methods, and user interfaces. This paper provides only a few notes from our experiences using the SAS® System to accomplish this enterprise-level goal. The point we hope we have made is that the SAS® System allows rapid design, development, and implementation of client-server systems - from data warehouses to end-user interfaces.

The SAS System will continue to play a key role as the enterprise level information delivery system for the increasing demand for Decision Support and Executive Information Systems from our customers.