A number of SAS/AF decision support systems have been developed which make use of data stored in a corporate SQL/DS data warehouse. This paper reviews the evolution of the data warehouse and the underlying architecture and principles related to it. In addition the paper describes a few of the SAS/AF applications which use data in the warehouse. These applications all support a common user interface and some of them incorporate SAS/GRAPH® software output as well. A discussion of application development guidelines is also included.

INTRODUCTION

Headquartered in Kingsport, Tennessee, Eastman Chemical Company is a large manufacturer of chemicals, fibers, and plastics. Eastman employs roughly 18,500 people in manufacturing and sales locations around the world. Eastman Chemical Company is a recipient of the 1993 Malcolm Baldrige National Quality Award. Formerly a division of Eastman Kodak Company, Eastman Chemical Company became an independent company on January 1, 1994.

SAS® software is installed on a variety of operating systems including MVS, CMS, VMS and OS/2®. In addition to SAS/AF and SAS/GRAPH software, the product mix includes SAS/STAT®, SAS/FSP®, SAS/QC®, SAS/CONNECT®, and SAS/ASSIST® software. In addition SAS/ACCESS® interfaces for DB2®, SQL/DS, Rdb, and IMS are used.

Application Development Environments

Though the primary emphasis of this paper is on the decision support applications which have been developed, an application is a direct product of its application development environment. Elements such as data design, application development design standards, and an appropriate set of application development tools must all be in place before a successful application can be delivered. Therefore, this paper includes information pertaining to both the SQL/DS and SAS application development environments which serve as the foundation for decision support applications being developed at Eastman Chemical Company.

BACKGROUND

Like most large companies, Eastman is challenged to find the best way to provide information to decision-makers throughout the company. In the past, some progress was made as "data" rather than "reports" flowed to the seekers of information. Tools such as the SAS System became important as the means for turning "data" into "information". However, several barriers prevented this from being completely effective.

Data-Related Barriers

Though more data was available to people than ever before, it was not always in a readily usable form. Sometimes the data extracts had complex structures such as multiple record types or repeating data groups. At other times, the extracted "data" was actually in a report format, complete with headings and control breaks. There is no doubt an experienced SAS programmer could write a program to read data in complex data structures or in report formats but this was certainly not an ideal situation.

Another barrier often encountered was the need for data from multiple sources. For example, someone needing to combine production and employee data would have to deal with multiple owner organizations and systems groups. Even after data extracts were provided to the requestor, combining the data in a meaningful way was a formidable task.

When data extracts were stored in a database, problems with incompatible data formats occurred. For instance, one database file might have had a field stored as character data while another file might have had the same field stored as numeric data. Any attempt to combine data from the two files based on that field would be complicated by the need to convert the field to a common format.

Data integrity and security were also important issues as data was replicated in multiple formats and in multiple places. It was virtually impossible to secure data based on data values. People had access to entire files or entire mindsets.

Other Barriers

Data availability and complexity were not the only problems in this environment. For example, some people had neither the aptitude nor the desire to write their own data analysis programs. Too often, the person who developed the programs was transferred to another group or department and untrained, inexperienced people were left to support complex programs. Instances of "orphan" applications without anyone to support them are very common throughout the company.

Breaking The Barriers

One area of the company which made significant progress toward making data available was the Sales and Marketing organization. For some time, the people in this organization had access to a RAMIS database containing historical sales data. A front-end application allowed users to generate a limited number of pre-programmed reports from this database. However, most people used the RAMIS report-writing language to create their own reports. While this application satisfied a great number of needs, several remained. One of these needs was security based on data values. Specifically, access based on country was desired. Secondly, since the data was stored in a proprietary database format, it was not easy to combine sales data with data from other systems unless data were first extracted from the RAMIS database. Finally, a relational database format was preferred over the hierarchical RAMIS database format.

In order to address some of these issues, a pilot project was begun with the intent to provide the Sales and Marketing organization with historical sales data in a relational database format. This effort was divided into two general tasks. The first involved identifying the data needed and designing it to satisfy the reporting and security requirements. The
second involved selecting an environment from which the data could be accessed.

Critical to the completion of the first task was the realization that the sales data taken from various on-line transaction processing systems needed to be redesigned before it could be effectively used. Summarization and de-normalization of the data were key factors to making it usable in a decision support environment.

The task of deciding on an environment was not quite as difficult. The two main contenders were MVS with DB2 versus CMS with SQUOS. Since almost everyone in the company had access to CMS in order to use OfficeVision/VM™ mail and calendaring facilities, CMS was an attractive candidate. The MVS environment was considered because of the fact that DB2 was already being used by large operational support systems. The deciding factor was the relative cost of running interactive sessions in the two operating environments. CMS had the advantage on that count.

Once the data design and environment selection were complete for the Sales and Marketing pilot project, a full-scale effort was begun to create a "data warehouse" environment for making additional data available.

SQL/DS DATA WAREHOUSE ENVIRONMENT

While this section of the paper discusses activities specific to the data warehouse environment residing on CMS using SQL/DS, many of the same concepts and principles apply in any environment using any relational database management system.

SQL/DS Coordination Committee

Along with the decision to use SQL/DS as the data warehouse environment came a need to coordinate the efforts to populate the warehouse. Since data from several different operational support systems needed to be extracted and transferred to the warehouse environment, systems analysts from several different groups needed to be involved. In many cases, the systems analyst responsible for extracting the data from an operational support system was not the same analyst using the data for a decision support system. Furthermore, data analysts (DA) were involved in coordinating data design and database analysts (DBA) were involved in creating and populating database tables. Additionally, assistance from VM systems programmers and software support personnel was required. Collectively, this group of people became known as the SQL/DS Coordination Committee. This group is self-governing and participation is voluntary.

Designing Data for Decision Support

Mentioned earlier was the need to recognize that data design for operational support systems is different from that for decision support systems. In most cases, data are summarized up to a level higher than that required by an operational support system. Also, some denormalization of data is desired for ease of use and performance gains. The additional data redundancy resulting from de-normalization is not nearly as much of an issue in a read-only environment as it would be in a transaction processing environment. This is because there is no risk of updating a data value in one table and forgetting to update it in another table.

liberal use of indexes is another difference found between decision support and operational support systems. Since indexes are not continually updated in a read-only environment, decreases in performance are not a concern. Some additional disk space is required to store the indexes, but this is completely offset by the performance gains.

View Structures

The foundation of our SQL/DS data warehouse architecture is a set of conventions governing the naming and creation of tables and views in the database. The "creator" portion of a table or view name corresponds to the system which "owns" the data. For example, master data such as a table of country names, belongs to the creator "MAST". The underlying table name begins with "TB_" followed by the descriptive portion of the name, for example, MAST:TB_COUNTRY:CODES. All data in the database is accessed via views rather than directly through tables. After the table is defined, the database analyst creates a "base view" of that table. This SQL/DS view has the same name as the table except for a prefix of "VW_" instead of "TB_", for example, MAST:VW_COUNTRY:CODES. When an application requires a combination of tables joined together to create a view, that entity is referred to as an "application view". For example, if the Marketing Decision Support System (creator MDSU) needed a view which joined sales data with customer data, the new view might be called MDSU:VW_SALES_BY_CUSTOMER.

For every "base view" and "application view" in the data warehouse, there is an additional SQL/DS view called a "SAS view". These views are created from the exact same underlying data as the base or application views but the view name and the column names are all eight characters or less in length. This greatly facilitates the use of the SQL procedure pass-through feature since longer SQL/DS column names do not have to be mapped to shorter SAS variable names. A "SAS view" has a prefix of "SV_". Using the same examples used earlier, we might have views named MAST:SV_COUNTRY and MDSU:SV_SLSBYCOUN. Each column in the underlying tables would also have its own character or less "alias" assigned as the column name in the "SAS view".

Someone familiar with the SAS/ACCESS interface products might raise a question at this point as to why SAS/ACCESS view descriptors would not be used in place of the SQL/DS "SAS views". Especially since the primary purpose of a SAS/ACCESS view descriptor is to accomplish the column name mapping just described. SAS/ACCESS view descriptors also serve to make data access "transparent", so that a programmer need not be concerned with the underlying data storage format. While these things are certainly true, our experience over time resulted in the realization that we had to give up those features for the sake of performance. We use exclusively the PROC SQL pass-through feature, so that data are subset, summarized, and sorted using native SQL before returning that data to a SAS program. Depending on the size of the underlying table (ours range from .1 to 1.5 million rows) and the nature of the query (especially when summarization is involved), this can make an enormous difference in the performance of a program.

Security

There are basically three types of security in place in the data warehouse: 1) none, 2) global, and 3) data value specific. No restrictions are applied to data that is considered to be "master" data or non-sensitive data. Examples of this type of data might include the aforementioned country code table. "No security" means that anyone in the company can read the data contained in a table or view. The second or "global" type of security is "all or nothing" security. In this situation, authority to read the data is granted to only the people who have an identified business need. Those people can see all of the rows of data available in a table or view. The third level of security restricts the rows of data based on specific data values. For instance, someone in a particular country might be authorized to see sales data pertaining only to that country.

The latter two types of security are accomplished through a data-driven approach. Security tables are maintained which identify the system which owns the data (creator, e.g. "MDSU"), a user, and for all but global security, one or more data values. Then for each level of security,
a special view is created with a join to the security table. For example, a
view of sales data which restricts rows to specific countries will be
created with an SQL statement which looks something like:

```
SELECT col1, col2, col3, ..., coln
FROM MDSS.VW_SALES,
SEC.VW_COUNTRY
WHERE USER = SEC.USERID
AND MDSS.COUNTRY = SEC.COUNTRY
```

A security suffix is appended to the view name to identify the level of
security associated with it. So, in the above example, the resulting view
would be named MDSS.VW_SALES_COUNTRY.

The security tables are maintained by designated custodians through a
SAS/AF application. This is desirable because a database analyst is not
involved in security administration and the true "owners" of the data are
responsible for granting authority to people to access the data.

SAS APPLICATION DEVELOPMENT ENVIRONMENT

SAS was selected as the preferred tool for developing decision support
applications at Eastman. So in addition to the SOLDS data warehouse
environment, we also needed to establish a SAS application
development environment to go with it.

User Interface Guidelines

Members of the SLODS Coordination Committee were the first to agree
that all decision support applications should have a common "look and
feel". It was also agreed that a "point and click" interface was desirable.

Rather than starting from scratch, the group decided to adopt the IBM™
text subset of the graphical model as the basis for a user interface
standard. This model is documented in the IBM publication Systems
Application Architecture™, Common User Access™: Basic Interface
Design Guide, SC26-4583. Even though a traditional interface was
available, interpretation of the documentation was rather difficult. In an
effort to provide a clearer interpretation of the IBM standard, a subset
group of the coordination committee adapted the IBM standard to exploit
features of SAS/AF whenever possible and to suggest alternative
methods when SAS could not easily conform to an IBM feature.
Achieving consistency from one application to the next was an
evolutionary process and became easier with each new application.

The improved guidelines have been embraced by application developers
and have already proved to shorten the design and programming phases
of applications. We have been able to achieve our primary goal of
providing a consistent interface across applications with the added
benefit of moving toward an object-oriented interface.

SAS/AF Tips and Techniques

In addition to the user interface guidelines mentioned above, a document
outlining good programming practices for SAS application developers is
also available. The primary goals of this document are to promote
consistency and portability when programming an application.

Examples of the types of information contained in this document include
suggestions for case conventions and indentation, coding all arguments to
a function instead of relying on defaults, using Screen Control Language
statements and functions instead of submitted SAS statements
whenever possible, and using data-driven techniques.

SAS/AF Common Modules

To facilitate application development even more, a library of common
SAS/AF programs has been created. This library is available to any
SAS application running under CMS or MVS. The common routines
conform to all of the established standards and guidelines. In addition to
providing useful functions which eliminate work for application
developers, these common routines serve as good examples when
developers need to create something similar.

Since some applications are used widely throughout Eastman and may
be used on even non-xenial type of terminal or terminal emulator,
common routines which have display windows are developed in two
formats. One format assumes the worst possible emulation available
and conforms strictly to text subset guidelines. This means choice
groups which would normally display "radio buttons" and "check boxes"
must be programmed to emulate those features through the use of text
characters. A graphical version of the same screen is developed for use
by applications which are delivered to a limited number of users who are
known to have appropriate terminal emulators. Some applications
choose to query the terminal device to see if it supports graphics and
call the appropriate version of a common routine. Another option which
is available for common routines which have display windows is to
display or not display a legend of function key definitions at the bottom of
the window.

Examples of some of the routines available are common dialog pop-up
windows to prompt for "are you sure?", "save and exit versus exit
without saving", and "file print/view/save"; a set of several programs
which can be used by an application to send OfficeVision/VIM notes; a
program which updates a specified SQL/DS table with data contained in
a SAS data set; and a program which evaluates an SQL/DS return code.

Design Reviews

During the process of refining the user interface standards, we also
identified a need for general design reviews. The goal of these reviews
is not necessarily to force adherence to standards but rather to assist
application developers who may be new to the SOLDS and/or SAS
application development environments. Design reviews include a review
of an application's security implementation as well as any potential
performance problems. Since design assistance is provided through the
early stages of an application by data analysts, database analysts and
SAS support personnel, an actual design review goes rather smoothly.

DECISION SUPPORT APPLICATION EXAMPLES

Three different applications will be reviewed in the following sections.
The first of these, the Marketing Decision Support System (MDSS), is
the direct result of the Sales and Marketing pilot project described
earlier. Although it is more of a reporting application than a true decision
support application, it is certainly worthy of mention because of the
features it offers. The second, Sales and Earnings Analysis System
(SEAS) and third, Estimated Comparative Earnings System (ECES) do
possess the features of a true decision support application. All three
applications are "data-driven" as much as possible. This is probably the
single-most important feature of any application since it is what
guarantees a virtually maintenance-free future. The evolutionary nature
of the common user interface is evident in these three applications.
They are presented in the order in which they were developed with the
ECES application having the benefit of earlier experience.

Marketing Decision Support System (MDSS)

The Marketing Decision Support System is based upon the historical
sales data now available via the SOLDS data warehouse. MDSS
serves as a replacement for the previously mentioned RAMIS
application. The design for MDSS was done by SAS Consulting®
Services personnel working closely with Eastman people. The
requirements included conformance to the common user interface
standards, the use of the PROC SQL pass-through facility to submit
One of the best features of MOSS is one which allows the users to "manage" the queries they construct. A sample list of queries is shown in Figure 1. Users may save, copy, and share queries with each other.

Another feature, related to performance, includes the ability to estimate the complexity of a query. This is done by returning a relative number based on a value returned by the SQL "prep" command. A relative number of 1 or 2 usually means a query should run relatively quickly with few resources. Anything over 4 or 5 is a candidate for overnight processing. Figure 2 shows a complete list of tasks available for a query.

When the "EDIT QUERY" option is selected, the user is then able to modify any of the four major components of a query. These components are shown in Figure 3.

When the "ROWS SELECTION SCREEN" option is selected, the user is presented with a list of all of the columns in the database which may be used for subsetting and sorting data in the report. This screen is shown in Figure 4. The list of columns displayed and information about those columns is stored in an application parameter file.

From any of the four major component screens for a query, a user may use the "Query Screens" pull-down to jump to another component screen. The pull-down options are shown in Figure 5.

Figure 6 shows the "SUMMARY COLUMN SELECTION" screen. This screen is very similar to the row selection screen. However, in this case, the columns displayed are numeric values which will be summarized in the query. On both the row and column selection screens, the user can manipulate the list by finding specific column names or by sorting the list.

The fourth component screen for a query is shown in Figure 7. From this screen, a user may select columns for which variance computations can be performed.
should be performed. Or, ranking selections may be specified. The "File" pull-down options are used to run or save a query.

**Figure 7: Output specifications for a query.**

The MDSS application has a cost-saving feature which allows a user to specify that a query should be run during non-prime-time at reduced billing rates. This information is provided when a query is run. The run method selection screen is shown in Figure 8. If the batch option is specified, the user is then asked to provide a starting date and time.

**Figure 8: Selecting when to run a query.**

Various output options available to a user are illustrated in Figure 9 and Figure 10.

**Figure 9: Selecting the output format for a query.**

The results of a query are displayed in the FSUST window as shown in Figure 12. This window has been customized to restrict the tasks which may be performed by a user.

**Figure 12: FSUST used to display report output.**

Sales and Earnings Analysis System (SEAS)

A Sales and Earnings Analysis System targeting the needs of business managers was next to go into production. It is a true decision support system which provides "drill-down" capabilities to business managers and financial analysts. Data may be displayed in both tabular and graphical forms. Users of this system can construct an "earnings statement" using any combination of customer, product, geographic, or market information. Users can further drill down to get detailed cost
elements for a given earnings statement. This application gives business managers a means for tracking costs at a lower level than ever before possible. Using SEAS, they are able to isolate areas of both high and low profitability, initiate actions to control costs, and effectively manage costs for their business organizations.

The decision support nature of SEAS allows users to navigate freely through the system and to drill down in any order desired. It also allows the user to maintain an application profile of various control limits which highlight out of control variance figures. Other features include a "drill-depth" window which shows current selection criteria and a means to change the comparison preference and time frame for the data being selected. Many of these features are illustrated in the figures below.

Upon entering the SEAS application, a user sees a detailed earnings statement based on the "start-from" criteria in his user profile. This is the screen shown in Figure 13.

The very first time a user invokes the SEAS application, he is prompted to complete his profile. However, at any time, a user may select the "Profile" pull-down and then select the "Start-From" option to get to the data warehouse on a quarterly basis. A comparison preference may also be specified. These options are available from the "View" pull-down as shown in Figure 17.

Another area of flexibility allows a user to modify the time frame for which data is being reviewed. Sales and earnings data are downloaded to the data warehouse on a quarterly basis. A comparison preference may also be specified. These options are available from the "View" pull-down as shown in Figure 17.

Adjusting the year and quarter for which data are being selected is as simple as clicking on a plus or minus push-button. This feature is shown in Figure 18.

Figure 15: Control limit portion of user profile.

Selecting the "Graph" pull-down option, as illustrated in Figure 16, indicates that graphs may be displayed for dollar amounts, unit values and sales percentages. This data may be displayed in either a bar or pie chart format. If the user has a graphics terminal or graphics emulation, the graph will appear in the SAS Graph window.

Figure 16: Options for graphs from detail screen.

Another area of flexibility allows a user to modify the time frame for which data is being reviewed. Sales and earnings data are downloaded to the data warehouse on a quarterly basis. A comparison preference may also be specified. These options are available from the "View" pull-down as shown in Figure 17.

Another feature of a user profile is the ability to specify control limits for variance values. These values may be modified at any time by selecting the "Profile" pull-down. The elements for which control limits may be specified are shown in Figure 15.

Figure 17: Changing or verifying the view of the data.

Adjusting the year and quarter for which data are being selected is as simple as clicking on a plus or minus push-button. This feature is shown in Figure 18.
The comparison preferences determine whether the quarters being compared represent cumulative or actual data. Also, a quarter may be compared to the corresponding quarter of a previous year, or the previous quarter of the same year. The preferences are displayed in Figure 19.

The next screen to appear is a listing screen showing the geographic areas of the world. At this point, a manager can quickly compare the performance of one area to another. Asterisks, as shown in Figure 20, indicate out of control variances. A user can select one row on the listing screen to see a detailed earnings statement for additional information.

The "Drill" option may also be selected from a listing screen. However, a row on the screen must first be selected. Assuming "Area Two" is the selected row, a user might now drill down again but this time by "Product". The "Drill" pull-down appears again in Figure 21.

The "Drill" option may also be selected from a listing screen. However, a row on the screen must first be selected. Assuming "Area Two" is the selected row, a user might now drill down again but this time by "Product". The "Drill" pull-down appears again in Figure 21.

The comparison preferences determine whether the quarters being compared represent cumulative or actual data. Also, a quarter may be compared to the corresponding quarter of a previous year, or the previous quarter of the same year. The preferences are displayed in Figure 19.

If, for example, "Geography" is selected, another pop-up screen appears to prompt for the type of geographic area desired. Area has been selected.

The "Drill" option may also be selected from a listing screen. However, a row on the screen must first be selected. Assuming "Area Two" is the selected row, a user might now drill down again but this time by "Product". The "Drill" pull-down appears again in Figure 21.
There are several ways in which product information may be presented. Figure 25 shows that "Sales Product Code" has been selected.

The listing screen in now populated with specific product information as shown in Figure 26. Graphs of the data may also be produced at this point.

Another look at the "Drill-depth" screen in Figure 27 shows the up to date selection criteria. The data have been subset further by geographic area and the listing screen now shows product-related information.

If a new estimate is to be created, a screen like the one shown in Figure 29 will appear. From this screen, the user selects the specific business unit, geographic area, and product for which an estimate is to be done.
An estimate is created by selecting data from several different sources and then presenting it in an earnings statement format. An estimate is made up of several different components as displayed in Figure 30. The first screen to be displayed is the Earnings From Operations (EFO) screen. On this screen, a user may change any of the values which appear in the right hand column labeled "User Estimate." New values will be automatically calculated for the remaining elements.

Figure 30: Components of an estimate as shown from the EFO screen.

In Figure 31, the "Detail Cost Data" option has been selected from the "View" pull-down. Then, the "SAA" option is selected to show that detailed information is available for "Selling," "Advertising," and "Administrative" costs.

Figure 31: Drilling down to detailed cost elements.

Figure 32 shows the Data Sources portion of an estimate. The values appearing in the "User Estimate" column may be different from the "Base Case" if the user has specified his own default values or has made changes which resulted in calculated values.

Figure 32: Verification of data sources.

The components of a user profile are displayed in the pull-down in Figure 33.

Figure 33: Specifying user profile options.

Default values may be specified in either percentages of the sales price or in per unit absolute amounts for the general areas of cost. These areas appear in the User Estimate Defaults screen illustrated in Figure 34.

Figure 34: Specifying default values for estimate.

The other part of a user profile controls default print options. A printed copy of an estimate may contain the "Base Case," the "User Estimate," or both sets of values. These options appear in Figure 35.

106
When the "Assumptions" option is selected from the "View" pull-down, a text editor window appears. A user may enter whatever comments he wishes to clarify details of an estimate. The text editor window as it appears in Figure 36 is one of the common routines available to SAS/AF programmers at Eastman.

Another common routine is called when a user attempts to exit from the system without saving an estimate. This window appears in Figure 37.

CONCLUSION

During the past two years we have struggled with learning new technology, solving performance problems, and learning to think in new ways but great strides have been made toward breaking down the barriers to effective information delivery.

There is now much more data available than ever before. To date, the data warehouse contains data for several different functional areas amounting to approximately 850 tables and 11 million rows. Furthermore, data is much more accessible because of the central storage location, consistent naming conventions, and standardized data formats. These factors also contribute toward improved data integrity. Security concerns are now addressed by authorizations based on data values. Information is readily available through both ad-hoc queries and easy-to-use applications such as MDSS, SEAS, and ECES.

The SAS decision support application development environment enables developers to deliver applications faster due to interface design guidelines, standard programming methods, and a library of common routines. In addition, a design review process insures consistency from one application to another.

Future plans include a Human Resources Decision Support System (HRDSS) which will be based on employee information. A second phase for the Sales and Earnings Analysis System is also planned. These applications will reap the benefits of the framework already in place.

REFERENCES


ACKNOWLEDGMENTS

I would like to recognize the efforts of those people who have been involved in the construction of the SQL/DS data warehouse. The people who worked on the original sales and marketing pilot were Kathryn Booth, Toni Corwin, Betty Ellege, Brian Harmon, and Ray Olinger. Technical support throughout this effort has been provided by Brian Harmon, data analyst and leader of the SQL/DS Coordination Committee; Don Owens and Harold Sheffield, database analysts; and Dan Chase, systems programmer. Thanks go to Jeff Freley, responsible for the design and development of the SEAS and ECES applications.

SAS, SAS/AF, SAS/ACCESS, SAS/ASSIST, SAS/CONNECT, SAS/FSP, SAS/GRAPH, SAS/QC, SAS/STAT and SAS Consulting are registered trademarks or trademarks of SAS Institute, Inc. in the USA and other countries. Common User Access, DB2, IBM, OfficeVisionVM, OS/2, SQL/DS, and Systems Application Architecture are registered trademarks or trademarks of International Business Machines. © indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.