EXPERIENCES USING SAS/ACCESS® IN A COMPLEX RELATIONAL DATABASE APPLICATION

Murty Arisetty, Howmedica, Div. of Pfizer Hospital Products Group, Inc.

ABSTRACT

The SAS System's Release 6.06 version of SAS/ACCESS software offers enhanced Relational Database to SAS System interface. Rdb/VMS® (RDB) was selected for the application as a corporate relational database management system (DBMS). After extensive evaluation of commercially available Fourth Generation Languages (4GL's), SAS/ACCESS®, SAS/Ap® and SAS/ASSIST® software together were selected as the 4GL. The combination of the three SAS software products offered a long-term strategic solution. This paper discusses experiences using SAS/ACCESS software and presents different tips, strategies, and techniques for developing applications using SAS/ACCESS software and RDB.

Topics will include:

- design concerns
- user interface techniques
- managing access and view descriptors
- handling date fields
- improving efficiency and optimizing performance
- limitations: how to get around
- tidbits
- recommendations for enhancing SAS/ACCESS software.

INTRODUCTION

SAS/ACCESS is a powerful software product from SAS Institute. This access feature provides both read and write capabilities to DBMS. It is transparent to SAS programmers/users. Our work with SAS/ACCESS 6.06 and Rdb/VMS (RDB) along with some creative and innovative strategies developed in-house resulted in a state-of-the-art production systems environment. System development was accomplished using SAS/ACCESS 6.06. A preliminary evaluation from beta test of the new release 6.07 is also included in this paper.

NATURE OF THE APPLICATION

Clinical research in the orthopaedic device industry involves collection of large amounts of data on patients at different visits. Complex data involving clinical and x-ray evaluations are collected on patients at each scheduled visit. Data collected from visit to visit varies. The duration of data collection activities on patients may cover a number of years. Also, the elapsed time between patient visits is usually long. Data for a patient typically arrives at the sponsoring company a page or two at a time. Data is processed, entered, validated and stored in the database shortly after it arrives.

Summaries, tabulations, frequency counts, listings and statistical analyses are generated from the database for submission to the Food & Drug Administration (FDA).

RDB AS DATABASE MANAGEMENT SYSTEM

RDB is based on the relational model. SAS system is a data analysis system with some database management capabilities. The RDB database serves as centralized medical data repository.
It provides better portability to other relational DBMS's on other platforms. RDB was cost effective as it was in use on the company VAX<sup>®</sup> cluster. RDB has the features of data independence and consistency, data integrity and security. Data independence refers to the independence of data and computer application programs written to access data. This implies that application programs do not depend on storage structure and access method.

Storing data in a relational database such as RDB offers a number of advantages over "file" management systems such as RMS<sup>™</sup>. It offers transaction journaling, rollback, audit, backup, restore and recovery capabilities. Data is easy to manage.

The DBMS offers a high level of security. Only the Database Administrator (DBA) is allowed to edit (add, change, delete) data from the database. Referential integrity and domain integrity constraints ensure validity of data in the database. The ability to maintain the database integrity in case of user errors is provided by DBMS integrity functions. Data is normalized to ensure no data inconsistencies occur.

The relational database is more flexible than other types of databases and provides a logical structure that is easy to understand for both users and programmers. DBMS lets a user combine (i.e., join/merge) and compare data in a wide variety of ways without complex programming.

SAS/ACCESS, SAS/AF AND SAS/ASSIST

SAS/ACCESS was selected to provide SAS access to RDB data. It provides both read/write access to SAS/AF, SAS/ASSIST and other SAS software applications. SAS/ASSIST users (primarily medical personnel and Medical Data Coordinators) use RDB data for ad hoc browsing, tabulations, frequency counts and reporting.

SAS/AF was utilized for such subsystems as RDB Data Edit and customized Data Browse. Base SAS was used to develop a number of utilities.

RDB alone cannot meet all our system requirements. SAS software products do not fill all our needs. The two together along with some 3GL ("c") development offer a complete solution.

ACCESS/VIEW DESCRIPTORS

SAS/ACCESS 6.06 allows creation of access and view descriptors in interactive Display Manager Session (DMS) environment only. Access descriptors are first created using create function in PROC ACCESS. An access descriptor establishes an interface between an RDB table/view and SAS. Only SAS names and formats may be altered. The access descriptor is then used to create and later edit view descriptor(s).

Selected fields from an access descriptor can be included in the view descriptor. The view descriptor facilitates both subsetting of RDB data using WHERE clause and sorting data (ORDER BY). It does not refer to the access descriptor once it is created except when it needs editing. Hence, a view descriptor(s) can function even if no associated access descriptor exists. In this paper, the term descriptor is used to refer to either an access descriptor or view descriptor.

DESIGN CONCERNS

SAS variable (field) and descriptor name length can be a maximum of eight (8) bytes long which limits the use of meaningful names. RDB allows name lengths up to 31 characters. Use of two different names (RDB and SAS) creates confusion, particularly when a large number of tables/fields is involved. The same names can be used in both RDB and SAS for small databases to make it convenient for all users and programmers.
SAS Institute announced (at SUGI16) their intention to increase the field/dataset name sizes in Release "7". When this happens, the above difficulty will be eliminated in the long run. The label associated with variable name in the descriptors is the RDB field name which is useful to the end users in interpreting the eight byte SAS names. If meaningful names are important, usage of dual names is recommended. However, when the RDB field name is eight (8) bytes or less, both RDB and SAS names should be the same.

When creating fields in RDB, assigning appropriate sizes will be helpful in the overall system optimization. TINYINT should be used for numeric fields with values less than 128. SMALLINT should be used for numeric fields that store values less than 32,768. Assigning proper sizes to character fields will be more efficient.

Good RDB database design should take into consideration SAS software based applications. Using two products (RDB and SAS) from two vendors creates the potential for difficulties. The complexity of the design increases. Also, linkage overhead is associated with this arrangement.

Standards play an important role in smoothing out complexities and were found to be extremely useful. Good standards should be followed both on the RDB and SAS/ACCESS sides.

SAS/ACCESS allows creation of access and view descriptors based on a single RDB database, hence applications design should take this into consideration.

SAS/ACCESS eliminates the need for storing data in two locations. At times, snapshots of data (temporary SAS datasets) need to be generated for statistical "program" development. Good data processing procedures should be implemented to administer these datasets.

USER INTERFACE TECHNIQUES

- Creating or storing view descriptors only for the RDB tables and columns (fields) required for a given application/user will help simplify user search and reduce disk storage. For example, an RDB database contains 100 tables, and a specific application involves data from 40 tables. In this case, storing view descriptors for only these 40 tables will be very efficient.

- It is effective and efficient to select only the required fields for inclusion in descriptors.

- When only one (1) view descriptor is created for an access descriptor on a directory, using the same name for both access and view descriptor will simplify maintenance.

- View descriptor should be treated as a SAS dataset for all practical purposes.

MANAGING ACCESS AND VIEW DESCRIPTORS

A large number of descriptors need to be created, maintained and managed for a database with a large number of tables. A DBA should be charged with this responsibility. Since SAS 6.06 allows for creation of these descriptors only in time consuming DMS mode, a Data Processing Assistant should be assigned to assist the DBA. Sets of views corresponding to protocols should be stored using directories.

SAS/ACCESS is capable of displaying access and view descriptor names from multiple libraries; but it works with only one library at a time. As an example, two libraries are opened; the first one contained access descriptors and the second contained the associated view descriptors. An attempt to edit a view descriptor would fail because the associated access descriptor exists on
Whenever modifications are made to an RDB table/view, all associated descriptors should be edited/recreated to ensure concurrency. Another solution is to use the "B" (Browse) function on the view descriptors to check out the status. SAS/ACCESS displays an error message if any discrepancies exist between a view descriptor and the associated RDB table/view. The edit (E) function should not be used to check out view descriptors as a precaution against inadvertent changes to data.

HANDLING DATE FIELDS

In the clinical research environment, several RDB fields have date values. Most date fields contain month, day, year format (MMDDYY6.). For example, Date of Birth (DOB) is collected and "stored" in MMDDYY6. format. Handling dates between RDB and SAS should be done with caution. Internal storage structures for dates in RDB and SAS are different. RDB uses November 17, 1858 as base (start). The number of seconds from the base is stored internally to calculate dates. It is stored as a 64-bit VAX standard absolute date structured as "DD-MMM-YYYY HH:MM:SS.CC" (similar to SAS Format DATETIME21.2). RDB provides no flexibility/choice in storing other date formats such as "MMDDYY6."

SAS uses January 1, 1960 as base. SAS offers versatile date and time functions. Whereas RDB stores date and time combined as one field, SAS can store as either two different fields for date and time or as one combined field.

Calculations involving two date fields will have to be done using the following:

\[
\text{No. of seconds} = (\text{Date}_2 - \text{Date}_1)
\]

(Note that RDB provides internal date in seconds. FLOOR function in base SAS to obtain largest integer.)

\[
\begin{align*}
\text{No. of Years} = \frac{(\text{Date}_2 - \text{Date}_1)}{365.25 \times 24 \times 60 \times 60} = \frac{(\text{Date}_2 - \text{Date}_1)}{31,557,600}
\end{align*}
\]

Given today's date and Date of Birth (DOB) to calculate age:

\[
\text{Age} = \frac{(\text{Today} - \text{DOB})}{(365.25 \times 24 \times 60 \times 60)} \text{ in years}
\]

When comparing dates, DATETIME7. format (DDMMDYY) is efficient:

If DOB > "01JAN60"D is valid

(If DOB > 010160 is not valid)

To obtain just date, use DATEPART function in SAS:

\[\text{HOSPDT} = \text{DATEPART} (\text{HOSP_ADM});\]

IMPROVING EFFICIENCY AND OPTIMIZING PERFORMANCE

- When the same view descriptor is used in more than one SAS step or PROC, it is generally more efficient to create a work dataset in DATA step and use the work data set for subsequent operations. This method improves CPU usage and also reduces elapsed time as SAS software does not have to wait for RDB to return same data more than once.
If a subset is needed from a complex merge involving view descriptors, it is possible to improve performance by first creating work datasets from view descriptors with subset clauses and then merging the subsetted work datasets. The WHERE clause is generally more efficient than IF.

If a view is used only once in a session/job, then it should be used directly without creating a work dataset.

When a majority of the SAS applications require sorted data, using the ORDER BY clause in the "Selection Criteria Entry Window" (namely SUBSET) of view descriptor is very efficient. Ordering data with the RDB primary key is more efficient than using non-key fields. For applications requiring sorted data occasionally, it is more efficient to sort data when needed.

LIMITATIONS: HOW TO GET AROUND

The linkage between access descriptors and RDB is of snapshot type and any metadata changes made to an RDB table/view subsequently are not known to the access descriptor, and consequently the associated view descriptor may be disabled. Any attempts to use the view descriptor might result in an error message. Whenever modifications are made to an RDB table/view, all associated access and view descriptors should be edited/recreated to ensure concurrency. Once an access descriptor is created, limited editing is allowed. Another solution is to use the "B" (Browse) function on the views to check out this concurrency.

The linkage between view descriptor and access descriptor is also of snapshot type. Any changes made to the access descriptor subsequently are not automatically incorporated in the view descriptors. Therefore, views should be edited/recreated to insure concurrency. This can be very tedious when a large number of descriptors is involved. SAS/ACCESS 6.06 is not of industrial strength in some ways. The 6.07 upgrade alleviates this difficulty with batch/noninteractive modes. A program file can be used to maintain access and view descriptors. Our experience showed that this is not a major issue once the application is in production mode as minor changes are made to the RDB metadata.

RDB index (primary key) is unknown to SAS/ACCESS. However, a view containing Indexes for all RDB tables can be created from RDB system table, RDB$INDEX_SEGMENT.

Contents of an access descriptor cannot be printed using PROC CONTENTS. PROC CONTENTS with MEMTYPE = ACCESS does not work and base SAS attempts to give contents of the same name view descriptor instead. Contents of an access descriptor can be printed using a Command-line Command when it is displayed on screen. The contents of a view descriptor can be printed using PROC CONTENTS. The output does not include text from SUBSET. The number of observations is shown as '.' (i.e., missing) and does not accurately reflect the number of rows in the RDB database. The values for the number of indexes and observation length are also incorrectly shown as '0'. The answer to whether data is compressed is printed as 'No', which is not accurate as RDB compresses character data by default. The informats shown are the same as format values. This information does not offer any
value to the user. However, it would be helpful if informat values stored the RDB data type and size.

- Limitation of 200 bytes in SAS could cause problems where character fields such as COMMENTS noted by an investigator are recorded. An RDB table can be designed to alleviate this difficulty.

- If an RDB view is based on multiple tables or other views, SAS cannot update the view's source tables. However, SAS view descriptors can be created to correspond to these source tables.

- The filename extension (SASEB$VIEW) is used for both view descriptors created in SAS/ACCESS and views created in PROC SQL. This might cause confusion when both these "types" of views are stored on the same directory. Even though both types are considered views, view descriptor and PROC SQL view are not identical entities. Views created from PROC SQL are also included in the Access Window Display. Some commands work on these views, PROC SQL views may not be edited using ED (Edit Descriptor). Storing PROC SQL views on a separate directory would alleviate the difficulty.

- On the ACCESS window, a 2 character long response is allowed. Response should start at the first position. An error message is given by SAS/ACCESS if "?" is entered at the second position.

- If a user without RDB write access (i.e., update, insert) attempts to edit data from a SAS software product such as PROC FSEDIT, the terminal hangs in limbo. No error message is given to the user. The process will have to be stopped to resume activity on the terminal.

- When using "SUBSET" for a view descriptor, the WHERE and ORDER BY clauses require the use of only RDB field names. This is required as SAS basically sends the SQL script as is to RDB. If these two clauses contain any SAS variable names (different from RDB column names) then, the view becomes disabled.

- RDB does not know the existence of any of the SAS/ACCESS side activity. This is not a problem by itself. But, the interface could be improved on the RDB side if Digital Equipment Corporation "works" with SAS Institute.

- SAS/ACCESS does not have complete knowledge of the RDB metadata. SAS/ACCESS does not recognize RDB field and table constraints, triggers, domains, comments on domains, security information, SNP data, segmented strings, security levels, and indices.

- RDB field constraints are not available to SAS. When PROC FSEDIT is used to edit data and the edited value violates an RDB constraint, SAS displays an error message sent by RDB. If a SAS/AF application is written for (RDB) data entry and editing and a need exists to better control data entry/data edit, a SAS dataset containing valid constraint values can be created and used. This solution has its own disadvantage of maintaining two sets of constraints.

- If an SCL application is used to edit the RDB database and an RDB error condition is encountered, SAS/ACCESS does not provide an error code to SCL to control execution of your application. When an error condition occurs, the SAS job is aborted.
SAS/ACCESS does not handle null values for RDB character fields correctly. RDB and SAS use different values internally to represent null values. RDB gives a value of "NULL" for null character fields; SAS uses " " (blank) to represent missing/null value. When RDB character data is edited using PROC FSEDIT or PROC FSVIEW, SAS sends " " (blank) for null values and RDB stores " " (blank). A blank is thus stored in RDB when the objective is to store NULL. A possible solution is to execute a post-edit RDB SQL update script that reads all rows in all tables in the RDB database and sets character field values from blank to NULLs. This process adds processing time. Also, this setting of blank character field values may not be relevant/appropriate for some fields in the application. If the character field has an RDB field constraint, the problem becomes more complex. The only way around this is to include " " as a valid constraint value and later use a trigger to set it to NULL.

TIDBITS

- When creating descriptors, only required fields should be selected to minimize data transfer from RDB to SAS, which in turn improves system performance.

- For efficient execution of SUBSET of view descriptor, attempt should be made to use primary keys in the WHERE and ORDER BY clauses.

- When creating an access descriptor, selecting "YES" for SAS/ACCESS prompt "ASSIGN NAMES:" results in unique SAS names. SAS names and formats may be edited subsequently.

- SAS/ACCESS generates default formats when creating an access descriptor. These default formats need to be changed, particularly for numeric (INTEGER, SMALLINT and TINYINT) fields. For date fields, DATETIME7. is recommended.

- Data type VARCHAR (namely, variable length character) is more efficient from a storage point of view. A field such as COMMENT from an investigator contains a variable length textual string. Currently, SAS format accounts for the maximum length specified in VARCHAR.

- For organizational efficiency, all descriptors should be set up using directories by protocol. Only the DBA should be allowed to edit access and view descriptors.

- By default, SAS 6.06 allocates 65 blocks (65*512 = 33,280 bytes) to store descriptors. Using CACHESIZ=0 option on LIBNAME statement reduces the storage requirement to 17 blocks (17*512 = 8,704 bytes) which is adequate for storing most descriptors, thereby saving 48 blocks (48*512 = 24,576 bytes). When a large number of descriptors is involved, the storage savings can be substantial.

- Using logicals to refer to the RDB database location while creating access and view descriptors is highly recommended. Simply reassigning the value of the logical will reference a different database location. If the database location is hardcoded into the access descriptor, it may not be edited later using "ED" function; the access descriptor needs to be deleted and recreated with the new database location. Also, using logicals will eliminate the need to create separate access descriptors and view descriptors for production and test environments.
Also in the LIBNAME statement of a SAS program, use of the logical will make the program more portable. The logical can easily be reassigned to refer to a different directory containing view descriptors without having to change SAS programs.

Views can be copied from one directory to another. These new views can then be edited using ED (Edit Descriptor) to modify the WHERE and ORDER BY clauses. This method facilitates fast and efficient setting up of views which have the same contents but different subsetting/sort criteria.

If only one view descriptor is created on an access descriptor, keeping the view descriptor name the same as access descriptor would simplify operations.

PROC SQL is a high powered SQL feature of base SAS. In cases involving complex joins, PROC SQL is a much more efficient tool to use than RDB SQL.

When temporary snapshots of data are needed, using PROC ACCESS to create a SAS output dataset is more efficient than using DATA step.

A SAS/AF application can be written to keep an on-line audit trail within the RDB database.

RECOMMENDATIONS FOR ENHANCING SAS/ACCESS SOFTWARE

- Capability to specify any valid SAS date format such as "MMDDYY6." for a descriptor would be beneficial. SAS/ACCESS should perform internal date conversion prior to making the date available for SAS use. It should also make the conversion implicitly prior to passing data for storage in RDB. An SCL program needs to be written currently to convert RDB dates into MMDDYY6. format for SAS/AF and SAS FSEDIT based updates. This enhancement would eliminate the SCL code involving date conversions.

- When creating access descriptors, the default format for numeric fields is a rather large number (e.g., 12). A user should be able to specify a default size for all fields via a global option statement such as "FORMAT_INTEGER = 6."

- Ability to store several views in a catalog type file would be extremely useful in organizing descriptors. Currently, each descriptor occupies a file. The number of files increases with the creation of new view descriptors.

- Keeping track of all view descriptors created from an access descriptor is currently a manual task. If SAS/ACCESS can be enhanced to track information on view descriptors based on each access descriptor, the need for manual tracking would be eliminated.

- Higher level editing of access descriptors would simplify the DBA’s work. The new editing features include ability to change an RDB column name and add a new RDB column name. Currently, view descriptors may not be edited if the associated access descriptor was created using "YES" to ASSIGN NAMES prompt. This restriction should be eliminated so that the view descriptors can be edited.

It would be useful to add two (2) new commands in ACCESS WINDOW:

CD - Compare and print differences between access descriptor and DBMS table/view.

301
SAS/ACCESS 6.07

SAS 6.07 alleviates some of the limitations discussed in this paper. This upgrade allows creation of access and view descriptors in both DMS and non-interactive or batch modes. The batch mode is an extremely useful feature for industrial strength applications. Password protection similar to SAS datasets is provided to descriptors. Access and view descriptors created with 6.07 may not be used in SAS 6.06; but upward compatibility exists. Another limitation is that the filename for an access/view descriptor cannot be used to determine whether it is created using 6.06 or 6.07. Intermixing 6.06 and 6.07 views on the same directory can be troublesome.

CONCLUSION

SAS/ACCESS is a powerful feature of SAS software. Our experience showed that the marriage between RDB and SAS software is not as "smooth" as we would have liked it to be. But, with some creative and innovative solutions and strategies developed in-house and excellent technical support from SAS Institute, Inc., a state-of-the-art production environment was created.

Technical support personnel from SAS Institute, Inc., particularly those in the VAX area and marketing, provided an excellent level of support during the development and implementation phases. It is very simple to set up access and view descriptors. Views offer adequate security. The functionality is good at a reasonable cost. The effort required to set up an application is reasonable. Even though some minor "bugs" exist in the product, the software works with no material effect on the functionality. Disk storage space taken up by access and view descriptors is modest. SAS/ACCESS 6.07 features may make it a potential industrial strength software product.

SAS/ACCESS eliminates the need to store data in two different systems and thus is justified. It is hoped that the guidelines and solutions discussed in this paper would be useful to developers of SAS/ACCESS based applications.

AUTHOR CONTACT

Murty Arisetty
Howmedica, Division of Pfizer Hospital Products Group, Inc.
359 Veterans Boulevard
Rutherford, NJ 07070
(201) 507-7801

REFERENCES


SAS, SAS/ACCESS, SAS/AF, SAS/ASSIST and SAS/FSP are registered trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

RMS/VMS, Rdb/VMS, VAX and VMS are trademarks of Digital Equipment Corporation.