INTRODUCTION

Views are fundamental. They may not appear to be, because we have gone so many years without them. But we have also gone without the applications that views make possible, and the clarity that they bring to programming.

Views are also complex. Their properties are poorly understood, their features rudimentary, and their documentation limited. But this was once true of DATA steps as well.

The aim of this paper is to encourage more sophistication in the development of views and techniques for their use.

CAVEATS

The answers provided in this paper should be considered to be approximate. The best sources of information are the SUGI Proceedings. Additional information was obtained through experimentation and discussions with SAS Technical Support.

SQL views were tested using release 6.06 and Beta 6.07 of SAS® software on VMS.

For DATA-step views, I used the Experimental release in 6.06, and the Beta release in Beta 6.07.

For SAS/ACCESS®, I used the interface to ORACLE in release 6.06. The description of release 6.07 is based upon production documentation alone.

One advantage of the lack of detailed official specifications is that these features can be improved with less concern for compatibility. Anything undocumented is subject to change.

WHAT IS A VIEW?

In Version 6, the term "SAS data set" incorporates both "SAS data files" (called tables in SQL terminology) and "SAS data views" (SAS Institute, 1990, page 200).

The data files supplying data to a view are termed "base tables". Views may reference other views, so there may be several layers of views above the base tables. When you use such a "chain" of views, you are limited by the weakest link in the chain.

TYPES OF VIEWS

The different types of views differ primarily in the languages used in their definition and the data which they can read. Additional considerations have received less attention, and these are the topic of this paper.

SAS release 6.07 supports three types of views:

SQL views

SQL is a standardized database query language supported by many vendors. In this paper, the term "SQL" refers to the SAS implementation.

SQL views are created using PROC SQL, introduced with base SAS release 6.06. SAS SQL views can access one or more SAS data sets (including any type of view).

Version 6.07 of SAS/ACCESS supports the "SQL Pass-Thru" facility for several database management systems, including ORACLE. A SAS SQL view can access ORACLE tables, using ORACLE SQL code embedded in the SAS view. These views have the properties of SQL views, providing an valuable alternative to SAS/ACCESS views. For example, SQL Pass-Thru views can reference multiple ORACLE tables, while SAS/ACCESS views are restricted to a single table.

Input DATA-step views

Input DATA-step views are a new feature in base SAS, release 6.07.301. The Experimental Release for SAS release 6.06 can be obtained through SAS Technical Support.

DATA-step views are created using the SAS DATA-step language, providing access to external files as well as multiple SAS data sets.

SAS/ACCESS views

Different SAS/ACCESS products interface with different data base management systems. My experience has been with SAS/ACCESS for ORACLE on VMS. Views for other interfaces may have different
SAS/ACCESS views are created using the Display Manager ACCESS window or (in release 6.07) batch PROC ACCESS. They can reference a single DBMS table.

**VIEWS = VIRTUAL DATA SETS**

The following discussion is organized by the parts of SAS data sets. It focuses on the differences between view types, rather than the differences between data files and views. Exceptions from general statements are described in italics. Software bugs are described in the endnotes.

**DATA SET HEADERS**

**Transporting**

SAS data files may be transported using the XPORT engine (creating transport files), PROC CPORT, or SAS/CONNECT™.

The only way to transport a SAS view is to transport the source code and compile it on the target machine.

**SAS/ACCESS views**: In release 6.06, there is no source code to transport. In release 6.07, PROC ACCESS code can be copied from one machine to another.

References to input data (at definition)

Program modules are simply text files, so they can be edited in any order.

Views that make use of the input data at view creation require that data files and views be created and modified in order of use, complicating development and maintenance.

**SQL views**: Only the syntax of the specifications is processed at view creation, allowing SQL views to be created or modified in any order.

**SAS/ACCESS views**: Creation of access descriptors (a step in the creation of views) requires access to the DBMS tables.

**DATA-step views**: Input SAS data sets and external files are referenced at view creation. If this is undesirable, an effective workaround is described by Squillace (1990, page 309).

References to input data (at execution)

Tables use their input data when they are created. There is then no need to maintain access to the input data.

Views (like programs) require access to the input data when executing. The names of data sets, libraries, and files referenced by views must not change between view creation and execution.

**SQL views**: Data set references without a libname are assumed to refer to the library in which the view is stored. This allows views to be moved from one library to another without modification (SAS Institute, 1989b, page 100).

**Member locking**

SAS data file headers cannot be modified while the data file is in use, because an exclusive lock must be obtained.

When a SQL or DATA-step view is in use, the base tables are protected from header modification, but there are no locks on the views themselves. If a view is modified during use, any new opens of the view will use the new definition.

**SAS/ACCESS views**: Views are locked during use, so all simultaneous opens of a view use the same definition.

**Privileges**

SAS release 6.06 does not support password-protection of data sets. If the operating system permits the SAS session to read and write a given file, then SAS will allow the user to do the same.

Release 6.07 restores and extends the password capabilities of version 5. Passwords can be hidden within the view definition, so users can be permitted to read a view but not the underlying data set (Klenz, 1991, page 295).

**SAS/ACCESS views**: The DBMS password is hidden within the view, even in release 6.06. Users can use the view without privileges to use the DBMS directly. Release 6.07 passwords provide security controlled from the SAS side.

**Re-creation of views**

Prior to SAS version 6, SAS data sets included a history section containing the code used to create the data set. This could be used to re-create the data set or verify that it was created correctly.

**SAS/ACCESS views**: The specifications defining a view can be reviewed using the Edit Descriptor selection-list command in the Access window (as long as the view still matches the base table). I would like to be able to re-create the PROC ACCESS code, so I could clone views for use with other tables.
SOL views: Code to re-create the view can be generated using the DESCRIBE statement of PROC SQL.

DATA-step views: There is no way to re-create the code defining a DATA-step view. It must be saved separately (SAS Institute, 1991b, page 151).

Member Labels

Explanatory labels may be attached to SAS data files using the LABEL= data set option.

Views cannot be labeled.

DATA-step views: A view may be labeled when it is created, but there is no way to modify the label later.

VARIABLES

Dynamic data vectors

Ordinary SAS DATA-steps are compiled at execution, using the current data set header information. This mechanism makes it possible to modify the behavior of all programs that use a particular data set simply by modifying the header of that data set, easing maintenance.

SAS/ACCESS views: All variable attributes are fixed at view creation.

SOL views: All variable attributes are determined at view execution.

DATA-step views: The length and type of explicitly referenced variables is fixed at when the view is created. The remainder of the data vector is constructed upon executing the view. (Stored DATA steps have the same properties).

Filters

DATA steps can manipulate or add variables while allowing other variables on the input data sets to pass through transparently. These generalized “filters” do not require modification when the input data structure is changed.

SAS/ACCESS views: At view creation, the output variables must be defined (SAS Institute, 1989a, page 40). You cannot define any transformed or computed variables.

SOL views: The SELECT * construct can be used to delay variable specification until execution of the view. The DROP, KEEP, and RENAME data set options cannot be used when re-creating a SQL view from within SQL (SAS Institute, 1991b, page 92), limiting the construction of filters.

Push-down KEEP lists

When several layers of views are defined, increased efficiency results from only processing those variables that are needed to produce the final result.


SOL views: Push-down KEEP is apparently supported only from a SQL SELECT statement.

DATA-step views: This feature is not supported.

Modification of attributes

The attributes of individual SAS data file variables can be modified by PROC DATASETS, the SQL ALTER statement, or the VAR or CONTENTS windows of Display Manager.

Views must be completely redefined to modify the attributes of any single variable.

SAS/ACCESS views: SAS/ACCESS views can be modified manually, through Display Manager windows. The batch ACCESS procedure of release 6.07 cannot be used to modify existing views (SAS Institute, 1991a, page 4).

Variable Labels

SAS data files and all types of views except SAS/ACCESS views support the definition of variable labels.

SAS/ACCESS views: The ORACLE column names are used for variable labels. There is no way to remove or override these labels.

OBSERVATIONS

Number of observations

SAS data files can efficiently report the number of observations, which is used in PROC CONTENTS, the CONTENTS window, and the DIR window (with DETAILS on).

Determining the number of observations that will be generated by a view usually requires materializing the entire view. For efficiency reasons, observations are not counted in SQL views and DATA-step views. A dummy value is used instead.

SAS/ACCESS views: The DBMS counts the
number of observations, at any cost.

Observation locking (for update)

SAS data files lock data observations prior to updating them, insuring that no one else can update the observation at the same time.

The CNTLLEV= data set option can be used to control whether locks are held on the record level or member level.

Only SAS/ACCESS views support updating of their base tables, so this issue does not arise for the other types of views.

**SAS/ACCESS views:** According to Plemons (1990, page 394), updates via PROC FSEDIT should request an update intent lock from ORACLE. The CNTLLEV data set option has no effect because all locking is controlled by ORACLE.

Input sequence

Views based on SQL (ACCESS views as well as SQL views) do not require sorted input.

**DATA-step views:** Like DATA-steps, DATA-step views work sequentially and often require pre-sorted input.

Output sequence

SAS data files may be read in physical order, or in order of any index, simply by specifying a BY statement.

**SAS/ACCESS views:** Observations can be read in any desired sequence, whether an index exists in the DBMS or not (SAS Institute, 1989a, page 62). The DBMS determines the optimal method of providing the sequence required by the BY statement (Klenz, 1991, page 290).

**SQL views:** Observations are always read in the sequence specified by the ORDER BY clause. If no ORDER BY clause is present, no particular sequence is guaranteed.

**DATA-step views:** Observations are always read in the order that the observations are generated by the DATA step.

Direct Access

SQL and DATA-step views of compressed data files can be accessed using SET POINT=, even though this method of access is unsupported by the base table. Sequential retrieval is used the first time an observation is read.

**SAS/ACCESS views:** Attempting to use SET POINT= gives an error message.

DATA VALUES

Data generation

When you read a SAS data file, you always get current data.

When you read a SAS view, you may get obsolete data for:
- observations that have been previously read by the view and subsequently updated, and
- observations added to the base table after the view has read the last observation in the table.

This problem occurs because observations generated by the view are saved in a "spill file" for future use (SAS Institute, 1991b, page 155). Spill files are used in SAS/FSP procedures and multi-pass procedures such as PROC PRINT with a BY statement (Jacobs, 1990, page 405).

**SAS/ACCESS views:** All data are frozen whenever the view is opened, the WHERE clause is changed, or a BACKWARD command is executed. Base table updates are not incorporated into the generated result until one of these events occurs.

Updating data through views

Only tables have data, so only tables can be updated.

Updates must be passed through views to the base table.

**SAS/ACCESS views:** Data can be updated.

**SQL views:** Data cannot be updated in release 6.07. Simple SQL views will be updatable in the future (SAS Institute, 1989b, page 43). This presumably will include SQL Pass-Thru views.

**DATA-step views:** Data cannot be updated.

INDEXES

Push-down WHERE clauses

The V606 engine uses indexes to navigate directly to the desired observations. These indexes are accessed using the WHERE clause or statement.

Only data files can be indexed, not views. Indexes can only be used by a view if the view can push the WHERE clause down to the base table.

Even if the base table indexes cannot be
reached, the ability to push down WHERE clauses as far as possible is still a crucial factor in view efficiency (Flemmons, 1990, page 395).

**SAS/ACCESS views:** WHERE clauses pushed into SAS/ACCESS views are pushed down into the DBMS and processed according to the rules of the DBMS. SAS/ACCESS does not adjust the WHERE clause for differing rules. For example, MISSING < 0 is TRUE in SAS but not in ORACLE.

**SQL views:** WHERE clauses are only pushed into SQL views from PROC SQL or SQL views.

**DATA-step views:** WHERE clauses cannot be pushed into DATA-step views.

**Surfacing of index information**

The SAS SQL optimizer selects a strategy for executing a given request based upon information about the input data sets (Kent, 1991, page 568), including the availability of indexes.

Even though a view allows WHERE clauses to be pushed-down until they reach an index, the SQL optimizer may select a less-than-optimal strategy if it cannot recognize the availability of an index.

**SAS/ACCESS views:** Kent (1991, page 571) states "the SAS/ACCESS Interface does not surface index information."

**SQL views:** The optimizer appears to be able to detect index availability.

**DATA-step views:** Indexes are never accessible through DATA-step views.

**CONCLUSION**

No one type of view is best for every purpose.

When the data is stored in a DBMS, there are two choices (in release 6.07): SAS/ACCESS views or SQL Pass-Thru views. SQL Pass-thru views are much more flexible, but cannot be used to update data. They also cannot assume a variety of sort orders.

SQL views are usually the best choice for views of SAS data sets, because of the sophistication of the SQL optimizer. SQL views are especially valuable when push-down of WHERE clauses is supported.

DATA-step views provide the flexibility needed for LAG operations, extensive variable transformation, and reading external files. Efficiency can be limited by the inability to accept pushed-down WHERE clauses.

Some of the differences discussed above are due to intrinsic limitations, some due to pursuit of different goals, and some are simply mistakes or unfinished work. In any case, they complicate the selection of which type of view to use.

**ENDNOTES**

1. Due to a bug in release 6.06 (SAS usage note V6-SQL-2766), the description of the WHERE clause may be incorrect. This is fixed in release 6.07.

2. Due to a bug in release 6.06, if the ORACLE table has been modified so that a column that used to be numeric is now character, an empty data set is produced rather than an error message. This is fixed in release 6.07.

3. Due to a bug in release 6.06, if any variable attribute is defined by a SQL view, the remaining attributes will not be inherited from the base table. This is fixed in 6.07.

4. Due to a bug in both releases 6.06 and 6.07 (SAS usage note V6-ENGINE-3517), SAS/ACCESS to ORACLE does not lock data rows prior to update. Therefore, data updates may get lost. For example,
   A. User 1 reads data observation.
   B. User 2 reads the same data observation.
   C. User 1 updates his copy of the observation and saves it.
   D. User 2 updates his copy of the observation and saves it, thereby overwriting the updates made by User 1.
   E. Neither user has any indication that anything went wrong.

5. Due to a bug in release 6.06 (SAS usage note V6-FSVIEW-3470), if FSVIEW is invoked in BROWSE mode, and then changed to MODIFY mode, updates through an ACCESS view are not applied to the ORACLE table. This is fixed in release 6.07.

**ACKNOWLEDGEMENTS**

Thanks to Mark Moorman, Stephen Vincent, Maureen Hayes, and Paul Kent for their help and good humor.

Thanks to Grant Blank for reviewing an earlier draft of the paper, and to LOREX Pharmaceuticals for providing an up-to-date computing environment.
REFERENCES


NOTES

An earlier version of this paper was delivered at the 1991 Midwest SAS Users Group conference (Norton, 1991).

SAS, SAS/ACCESS, SAS/FRSP, SAS/CONNECT, and SAS/DB2 are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

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

The author may be contacted at: Andrew A. Norton Trilogy Consulting Corporation 1616 1/2 Forest Place Evanston, IL 60201 (708) 982-8466