SAS Data Views: A Virtual View of Data

John C. Boling, SAS Institute Inc., Cary, NC

ABSTRACT

The concept of a SAS data set has been extended or broadened in Version 6 of the SAS System. Two SAS file structures now function as SAS data sets: SAS data files and SAS data views. Both file structures can be processed as SAS data sets in DATA steps or PROC or procedure steps. This paper introduces the concept of a SAS data view, presents its advantages and disadvantages, discusses the three types of SAS data views, and compares its structure with that of a SAS data file.

Defining a SAS Data View

The SAS data file is already familiar to experienced SAS users because it represents what, historically, has been called a SAS data set. For example, 1982 Version SAS data sets and Version 5 SAS data sets are technically referred to now as SAS data files. This is essentially a change in terminology. The new file structure and definition in Version 6 is the SAS data view.

Unlike SAS data files, SAS data views do not contain actual data. Rather, SAS data views describe data stored in other file structures. Examples of other file structures are

- DB2, SQL/DS, and ORACLE tables
- sequential files and VSAM files
- other SAS data sets.

The data the view describes are obtained when the view is processed as a SAS data set in a DATA step or PROC step. It is appropriate to refer to SAS data views as virtual SAS data sets since they represent virtual views of data. The data are obtained at run time.

For example, this program prints the SAS data set SASDATA.YEAR93:

```
proc print data=sasdata.year93;
run;
```

If the data set is a SAS data file, the PRINT procedure processes the data stored in the SAS data file. If the data set is a SAS data view, the instructions in the view are used to retrieve the data and pass them to the calling task, in this case PROC PRINT, for processing. Thus PROC PRINT can directly print data from a DB2 table, a sequential file, a VSAM file, a Lotus 1-2-3 file, or even multiple files.

Additionally, the instructions in the view definition can specify how to subset the rows (observations), the columns (variables), and can even create new columns (variables) not stored in the original file.

SAS Data View and Engine Processing

When a SAS data set is processed, a request for data is made to the SAS supervisor. The supervisor recognizes the SAS data set is a SAS data view and loads the appropriate engine to obtain the data.

An engine is a set of routines or instructions executing on a host that interfaces to a specific file format. Engines allow these file formats to be processed as SAS data files. The responsibility of the engine is to obtain the data, and pass them to the calling task for processing. In other words, the engine materializes the data for appropriate processing by the DATA step or PROC step.

Advantages of SAS Data Views

There are a number of advantages to using SAS data views. In particular, they

- allow definition of multiple perspectives on data
- minimize data replication
- minimize maintenance
- always process current data
- allow processing of proprietary file structures
- allow source code to be hidden from users.

Defining Multiple Perspectives on Data

Since the data reside in other file structures, SAS data views can represent multiple views or perspectives on the same file. For example, a corporate office maintains financial data on the Chicago, New York, Kansas City, and Seattle offices in a common DB2 table. However, when the Chicago office processes the data, the Chicago office only sees the data relevant to their office. Likewise, when the New York office processes the data, the office only processes the New York data. A different view is defined for each regional office to use against the same table of data.

Minimizing Data Replication

Disk storage is premium. Unfortunately, in many organizations data replication occurs frequently and this valuable storage space is wasted.

By defining multiple views to a common file, data replication is avoided. In the previous example, it is not necessary for each regional office to have its own copy of the data since they are stored in a common file.

Since no data are stored in the SAS data view, the view requires little storage space.

Data Are Always Current

The view does not obtain the data until the view is processed. Thus, as the data values in the file change, the view definition always reflects the current data.

Processing Proprietary Files

Views permit the SAS System to process other proprietary file structures directly without first extracting the data and reading them into a SAS data file. SAS data views use engines, which are instructions that tell the SAS System how to logically model and interpret other proprietary files directly as SAS data sets.

Hiding Source Code

Source code can be shielded from users and stored in a SAS data view definition. The user simply processes the view as a SAS data set and the view handles the processing.

Disadvantages of SAS Data Views

One major disadvantage to using SAS data views is the additional overhead required to process data defined by a SAS data view as compared to processing a SAS data file.

The overhead is a function of engine processing and the layering involved since it is possible to chain multiple SAS data view definitions together. If the same data will be analyzed in multiple steps in the same program, it is advisable to first create a SAS data file representing the data and then process that file.

A secondary disadvantage is that the processing instructions comprising a type of SAS data view (know as a DATA step view) cannot be examined directly. Although this protects the DATA step view definition from other users, the source code is not retrievable from the view definition. The source code must be explicitly saved prior to storing it in the view if the source code needs to be examined.

Types of SAS DATA Views

There are three types of SAS data views:

- SAS/ACCESS® views
- SQL views
- DATA Step views

SAS/ACCESS Views

SAS/ACCESS views are created using SAS/ACCESS software and allow access to other vendors' proprietary file structures. The specific interface engines supported include:

- DB2
- SQL/DS
- IMS-DI/I
- IDMS
- DB2/2
- SQL/400
- ORACLE
- Rdb/VMS
- CA-DATACOM/DB
- INGRES
- INFORMIX
- SYBASE's SQL Server
- Microsoft's SQL Server
- ODBC
- ADABAS
- System 2000 Data Management Software
- PC file formats

SAS/ACCESS views support both read and write functionality.

SQL Views

SQL views of SQL tables (SAS data sets are SQL tables) are created by PROC SQL and contain SQL statements. SQL views provide read functionality only.

DATA Step Views

DATA step views are created in the DATA step and contain DATA step statements. DATA step views provide read functionality only.

Structure of SAS Data Files

SAS data files have MEMTYPE=DATA and contain both a descriptor component and a data component. Information from the descriptor component can be examined using PROC CONTENTS and the data component using PROC PRINT. Thus, when the descriptor component and data component are physically stored together in the same file structure, the SAS data set is referred to as a SAS data file. Figure 1 illustrates the structure of a SAS data file and shows that information can be examined and processed with DATA and PROC steps.

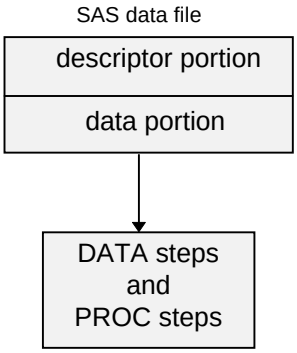

Figure 1

As Figure 2 illustrates, a SAS data view represents a separation of the descriptor component and the data component. A SAS data view stores a descriptor component, but no data component. The data are stored externally to the SAS data view. SAS data views have MEMTYPE=VIEW. When a SAS data view is processed by a DATA step or PROC step, the view obtains the data and passes them to the DATA step or PROC step for processing.

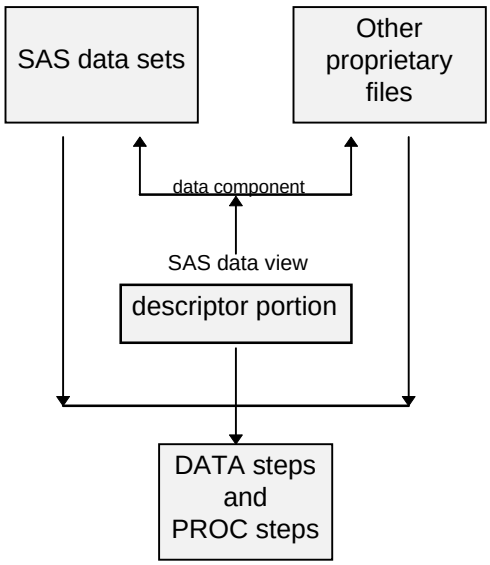

Figure 2

Identifying SAS Data Files and SAS Data Views

Since both SAS data files and data views are classified as SAS data sets, a SAS data file cannot have the same name as a SAS data view within the same physical or logical SAS library.

The CONTENTS procedure can be use to verify the SAS data set MEMTYPE as DATA (a SAS data file) or VIEW (a SAS data view). The engine definition identifies whether the view is a SAS/ACCESS view (the appropriate proprietary engine is reported), an SQL view (SASESQL engine), or a DATA step view (SASDSV engine). Since the view contains no data, the number of observations reported by PROC CONTENTS is missing.

Creating and Using SAS/ACCESS Views

SAS/ACCESS views are created using the appropriate SAS/ACCESS software. The following program uses a SAS access file named SASDATA.FRQFLYR to a DB2 table to create a view named SASDATA.N93. The view selects the columns FFID, MBRTYPE, NAME, and MILETRAV and subsets those rows that satisfy the WHERE expressions.

```
proc access dbms=db2 access=sasdata.frqflyr;
  create sasdata.ny93.view;
  select ffid mbrtype name miletrav;
  subset where state='NY' and
     miletrav>90000;
run;
```

When the TABULATE procedure processes the data set SASDATA.NY93, TABULATE is actually processing the selected columns and rows directly from the DB2 table.

```
proc tabulate daya=sasdata.ny93;
   var miletrav;
   class mbrtype;
   tables mbrtype,miletrav*mean;
run;
```

If write authority to the DB2 table is granted, the FSEDIT procedure can be used to edit the data directly in the DB2 table.

Creating and Using SQL Views

SQL views are created using the CREATE VIEW statement in PROC SQL. Views can be defined to process one or more SAS data sets. Thus, views can be defined to process other SAS data files or other SAS data views.

The following SQL example creates a view named SASDATA.PILOTV. The SQL view extracts columns IDNUM, JOBCODE, GENDER, and SALARY from the SAS data set SASDATA.PERSONL for pilots who earn less than $70,000. The view also creates a computed column named BONUS which is defined as eight percent of the employee's salary. It then sorts the data by SALARY.

```
proc sql;
  create view sasdata.pilotv as
  select idnum, jobcode, gender, salary,
     salary*.08 as bonus
     from sasdata.personl
     where jobcode contains 'PILOT'
       and salary<70000
     order by salary;
```

The SQL statements are compiled and stored in the SAS data view SASDATA.PILOTV which defines a view to the data stored in SAS data set SASDATA.PERSONL.

The data the view represents are not retrieved until the view is processed. When the SAS/GRAPH® procedure GCHART processes the SAS data view SASDATA.PILOTV, the columns are extracted from the SAS data set SASDATA.PERSONL, the BONUS column is computed, the rows are subsetted and ordered, and the bar chart is produced.

```
proc gchart data=sasdata.pilotv;
   vbar jobcode/sumvar=bonus ctext=cyan
      patternid=midpoint;
   title 'Bonus Information';
run;
```

The DESCRIBE statement is PROC SQL is used to review the instructions stored in an SQL view.

```
proc sql;
  describe view sasdata.pilotv;
```

The SQL instructions appear in the SAS log:

```
NOTE:   SQL view IN.PILOTV is defined as:

 select IDNUM, JOBCODE, GENDER, SALARY,
    SALARY*0.08 as BONUS
    from IN.PERSONL
 where JOBCODE contains 'PILOT' and
   (SALARY<70000)
 order by SALARY asc;
```

Views can also be created of other SAS data views; thus, views can be chained together.

The following program creates a view named SASDATA.MPILOT that extracts IDNUM, SALARY, and BONUS columns for male pilots from the SAS data set SASDATA.PILOTV.

```
proc sql;
  create view sasdata.mpilot as
  select idnum, salary, bonus
     from sasdata.pilotv
     where gender='M';
```

When the view SASDATA.MPILOT is processed, the view processes the SAS data set SASDATA.MPILOTV, which is a view definition that obtains the data from the SAS data file SASDATA.PERSONL.

If a view in the chain is not found, the view definition does not execute.

Multiple SAS data sets can be incorporated into a SQL view definition. In the following example, specific columns are selected from three SAS data sets: SASDATA.PRICES, SADATA.ORDERS, and SASDATA.CUSTOMER. Specific rows are joined (merged) using different keys. The view creates a new column COST and orders the rows by two sort keys. The three SAS data sets could represent any SAS data file or SAS data view, that is, joining three sequential files; a DB2 table, a sequential file, and a SAS data file; or a DB2 table, an IMS table, and a DATACOM/DB table.

```
proc sql;
  create view sasdata.company as
    select company, city, state,
      orders.product,
      quantity, unitcost*quantity as cost
      from sasdata.prices, sasdata.orders,
        sasdata.customer
      where prices.product=orders.product and
        orders.custid=customer.custid
      order by orders.custid, orders.product;
run;
```

Creating and Using DATA Step Views

DATA step views are created using the DATA step. Just as SQL views contain compiled SQL statements, DATA step views contain compiled DATA step statements.

Any DATA step program can essentially be stored in a view definition. Certain statements such as global statements or host specific options, if present in the DATA step, are not stored in the view definition.

When the view is processed as a SAS data set, the DATA step program obtains the data and returns them to the calling task for processing.

Prior to Version 6, data in an external file had to be stored in a SAS data set before it could be processed by SAS procedures.

```
data sasdata.current;
  infile mydata;
  input @20 trandate mmddyy8. @;
  if trandate=today();
  input @1 invoice $char4. supplier $char15.
    @28 itemno $char4. amount comma8.
    clerk $char6. location $2. state $2.
    billcode $3. priority $1. quantity comma6.
    payflag $1.;
  if billcode='120' then due=trandate+45;
    else due=trandate+30;
  format trandate due date7.;
run;
```

To create a DATA step view to the external file, add the VIEW=option to the DATA statement. When the DATA step executes, a compiled version of the program is created and stored in SASDATA.CURRENT. Note that the view name must correspond to the SAS data set name.

```
data sasdata.current/view=sasdata.current;
  infile mydata;
  input @20 trandate mmddyy8. @;
  if trandate=today();
  input @1 invoice $char4. supplier $char15.
    @28 itemno $char4. amount comma8.
    clerk $char6. location $2. state $2.
    billcode $3. priority $1. quantity comma6.
    payflag $1.;
  if billcode='120' then due=trandate+45;
    else due=trandate+30;
  format trandate due date7.;
run;
```

When the PRINT procedure processes the SAS data view, the sequential file is read, subsetted, a new column DUE created, and the resulting data passed to the PRINT procedure.

No intermediate SAS data file is created. The PRINT procedure is printing the selected records and fields from the sequential file as defined by the view.

```
proc print data=sasdata.current;
   title “Transactions for &sysdate”;
run;
```

The original DATA step source statements cannot be retrieved from a stored DATA step view definition. It is recommended that the source code be saved so that it can later be retrieved.

The following DATA step interleaves four SAS data sets together and creates a new SAS data file. If the date for QTR1, QTR2, QTR3, or QTR4 changes, the SAS data file YEAR must be recreated.

```
data sasdata.year;
  set sasdata.qtr1 sasdata.qtr2 sasdata.qtr3
    sasdata.qtr4;
  by account;
run;
```

The interleave operation could be stored in a view definition. If the data sets being interleaved are updated, the view definition does not require any changes.

```
data sasdata.year;
  set sasdata.qtr1 sasdata.qtr2 sasdata.qtr3
    sasdata.qtr4;
  by account;
run;
```

The MEANS procedure generates descriptive statistics on the interleaved data return by the view definition.

```
proc means data=sasdata.year mean min max;
   title “Descriptive Statistics”;
run;
```

Summary

The concept of a SAS data set has changed significantly in Version 6. Two SAS file structures function as SAS data sets - SAS data files and SAS data views. A SAS data file contains data whereas a SAS data view only describes the data. The actual data are stored in other file structures external to the view.

SAS data views allow you to define one or more view perspectives on data residing in proprietary file structures; on SAS data sets (SAS data files and SAS data views); on SQL tables; and on any file structure that can be read with the DATA step.

Views can be created using SAS/ACCESS software, PROC SQL, and the DATA step. When the view is processed as a SAS data set, the view retrieves the data it represents and passes the data to the calling task. Since the view only stores a description of the data, not the data themselves, the view always processes the most current data. Views can be used to minimize data replication, minimize file maintenance, process proprietary file structures, and hide source code.

This extended SAS data set definition enhances the richness of the SAS System for processing a variety of different file structures.