**Definition of SAS Views**

A **SAS view** is a type of SAS data set that retrieves data values from other files. A SAS view contains only descriptor information such as the data types and lengths of the variables (columns), plus information that is required for retrieving data values from other SAS data sets or from files that are stored in other software vendors' file formats. SAS views are of member type VIEW. In most cases, you can use a SAS view as if it were a SAS data file.

There are two general types of SAS views:

**native view**

is a SAS view that is created either with a DATA step or with PROC SQL.

**interface view**

is a SAS view that is created with SAS/ACCESS software. An interface view can read data from or write data to a database management system (DBMS) such as DB2 or ORACLE. Interface views are also referred to as **SAS/ACCESS views**. In order to use SAS/ACCESS views, you must have a license for SAS/ACCESS software.

**Note:**   You can create native views that access certain DBMS data by using a SAS/ACCESS dynamic LIBNAME engine. See [SAS/ACCESS Views](http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/a001278892.htm), or the SAS/ACCESS documentation for your DBMS for more information.

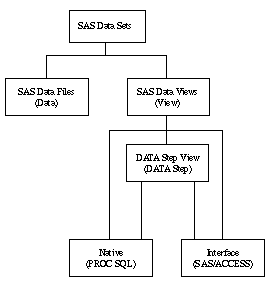
**Benefits of Using SAS Views**

SAS views provide the following benefits:

* Instead of using multiple DATA steps to merge SAS data sets by common variables, you can construct a SAS view that performs a multi-table join.
* You can save disk space by storing a SAS view definition, which stores only the instructions for where to find the data and how it is formatted, not the actual data.
* SAS views can ensure that the input data sets are always current because data is derived from SAS views at execution time.
* Since SAS views can select data from many sources, once a SAS view is created, it can provide prepackaged information to the information community without the need for additional programming.
* SAS views can reduce the impact of data design changes on users. For example, you can change a query that is stored in a SAS view without changing the characteristics of the view's result.
* With SAS/CONNECT software, a SAS view can join SAS data sets that reside on different host computers, presenting you with an integrated view of distributed company data.

The following figure shows native and interface SAS views and their relationship to SAS data files:

***Native and Interface SAS Views***



You can use SAS views in the following ways:

* as input to other DATA steps or PROC steps
* to migrate data to SAS data files or to database management systems that are supported by SAS
* in combination with other data sources using PROC SQL
* as pre-assembled sets of data for users of SAS/ASSIST software, enabling them to perform data management, analysis, and reporting tasks regardless of how the data is stored

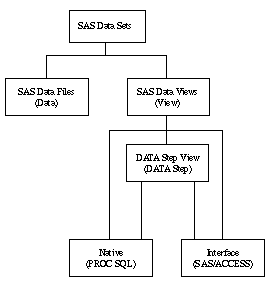
**Benefits of Using SAS Views**

SAS views provide the following benefits:

* Instead of using multiple DATA steps to merge SAS data sets by common variables, you can construct a SAS view that performs a multi-table join.
* You can save disk space by storing a SAS view definition, which stores only the instructions for where to find the data and how it is formatted, not the actual data.
* SAS views can ensure that the input data sets are always current because data is derived from SAS views at execution time.
* Since SAS views can select data from many sources, once a SAS view is created, it can provide prepackaged information to the information community without the need for additional programming.
* SAS views can reduce the impact of data design changes on users. For example, you can change a query that is stored in a SAS view without changing the characteristics of the view's result.
* With SAS/CONNECT software, a SAS view can join SAS data sets that reside on different host computers, presenting you with an integrated view of distributed company data.

The following figure shows native and interface SAS views and their relationship to SAS data files:

***Native and Interface SAS Views***



You can use SAS views in the following ways:

* as input to other DATA steps or PROC steps
* to migrate data to SAS data files or to database management systems that are supported by SAS
* in combination with other data sources using PROC SQL
* as pre-assembled sets of data for users of SAS/ASSIST software, enabling them to perform data management, analysis, and reporting tasks regardless of how the data is stored

**DATA Step Views**

|  |
| --- |
|  |
| **Definition of a DATA Step View** |

A **DATA step view** is a native view that has the broadest scope of any SAS view. It contains a stored DATA step program that can read data from a variety of sources, including:

* raw data files
* SAS data files
* PROC SQL views
* SAS/ACCESS views
* DB2, ORACLE, or other DBMS data

|  |
| --- |
|  |
| **Creating DATA Step Views** |

In order to create a DATA step view, specify the VIEW= option after the final data set name in the DATA statement. The VIEW= option tells SAS to compile, but not to execute, the source program and to store the compiled code in the input DATA step view that is named in the option.

For example, the following statements create a DATA step view named DEPT.A:

libname dept *'SAS-library';*

data dept.a / view=dept.a;

*... more SAS statements ...*

run;

Note that if the SAS view exists in a SAS library, and if you use the same member name to create a new view definition, then the old SAS view is overwritten.

Beginning with Version 8, DATA step views retain source statements. You can retrieve these statements using the DESCRIBE statement. The following example uses the DESCRIBE statement in a DATA step view in order to write a copy of the source code to the SAS log:

data view=inventory;

describe;

run;

For more information on how to create SAS views and use the DESCRIBE statement, see the DATA statement in **SAS Language Reference: Dictionary**.

|  |
| --- |
|  |
| **What Can You Do with a DATA Step View?** |

Using a DATA step view, you can do the following:

* directly process any file that can be read with an INPUT statement
* read other SAS data sets
* generate data without using any external data sources and without creating an intermediate SAS data file.

Because DATA step views are generated by the DATA step, they can manipulate and manage input data from a variety of sources including data from external files and data from existing SAS data sets. The scope of what you can do with a DATA step view, therefore, is much broader than that of other types of SAS views.

|  |
| --- |
|  |
| **Differences between DATA Step Views and Stored Compiled DATA Step Programs** |

DATA step views and stored compiled DATA step programs differ in the following ways:

* a DATA step view is implicitly executed when it is referenced as an input data set by another DATA or PROC step. Its main purpose is to provide data, one record at a time, to the invoking procedure or DATA step.
* a stored compiled DATA step program is explicitly executed when it is specified by the PGM= option on a DATA statement. Its purpose is usually a more specific task, such as creating SAS data files, or originating a report.

For more information on stored compiled DATA step programs, see [Stored Compiled DATA Step Programs](http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/a000988945.htm).

|  |
| --- |
|  |
| **Restrictions and Requirements** |

Global statements do not to apply to a DATA step view. Global statements such as the FILENAME, FOOTNOTE, LIBNAME, OPTIONS, and TITLE statements, even if included in the DATA step that created the SAS view, have no effect on the SAS view. If you do include global statements in your source program statements, SAS stores the DATA step view but not the global statements. When the view is referenced, actual execution can differ from the intended execution.

When a view is created, the labels for the variable that it returns are also created. If a DATA step view reads a data set that contains variable labels and a label is changed after the view is created, any procedure output will show the original labels. The view must be recompiled in order for the procedure output to reflect the new variable labels.

If a view uses filerefs or librefs, the fileref or libref that is used is the one that is defined at the time that the view is compiled. This means that if you change the file that is referenced in a fileref that the view uses, the new file is ignored by the view and the file that is referred to by the fileref at the time the view was compiled continues to be used.

|  |
| --- |
|  |
| **Performance Considerations** |

* DATA step code executes each time that you use a DATA step view, which might add considerable system overhead. In addition, you run the risk of having your data change between steps. However, this also means that you get the most recent data available--that is, data when the view is executed as compared to data when the view was compiled.
* Depending on how many reads or passes on the data are required, processing overhead increases.
  + When one sequential pass is requested, no data set is created. Compared to traditional methods of processing, making one pass improves performance by decreasing the number of input/output operations and elapsed time.
  + When random access or multiple passes are requested, the SAS view must build a spill file that contains all generated observations so that subsequent passes can read the same data that was read by previous passes. In some instances, the view SPILL= data set option can reduce the size of a spill file.

|  |
| --- |
|  |
| **Example 1: Merging Data to Produce Reports** |

If you want to merge data from multiple files but you do not need to create a file that contains the combined data, you can create a DATA step view of the combination for use in subsequent applications.

For example, the following statements define DATA step view MYV9LIB.QTR1, which merges the sales figures in the data file V9LR.CLOTHES with the sales figures in the data file V9LR.EQUIP. The data files are merged by date, and the value of the variable **Total** is computed for each date.

libname myv9lib '*SAS-library*';

libname v9lr '*SAS-library*';

data myv9lib.qtr1 / view=myv9lib.qtr1;

merge v9lr.clothes v9lr.equip;

by date;

total = cl\_v9lr + eq\_v9lr;

run;

The following PRINT procedure executes the view:

proc print data=myv9lib.qtr1;

run;

|  |
| --- |
|  |
| **Example 2: Producing Additional Output Files** |

In this example, the DATA step reads an external file named STUDENT, which contains student data, and then writes observations that contain known problems to data set MYV9LIB.PROBLEMS. The DATA step also defines the DATA step view MYV9LIB.CLASS. The DATA step does **not** create a SAS data file named MYV9LIB.CLASS.

The FILENAME and the LIBNAME statements are both global statements and must exist outside of the code that defines the SAS view, because SAS views cannot contain global statements.

Here are the contents of the external file STUDENT:

dutterono MAT 3

lyndenall MAT

frisbee MAT 94

SCI 95

zymeco ART 96

dimette 94

mesipho SCI 55

merlbeest ART 97

scafernia 91

gilhoolie ART 303

misqualle ART 44

xylotone SCI 96

Here is the DATA step that produces the output files:

libname myv9lib '*SAS-library*';

filename student '*external-file-specification*'; **1**

data myv9lib.class(keep=name major credits)

myv9lib.problems(keep=code date) / view=myv9lib.class; **2**

infile student;

input name $ 1-10 major $ 12-14 credits 16-18; **3**

select;

when (name=' ' or major=' ' or credits=.)

do code=01;

date=datetime();

output myv9lib.problems;

end; **4**

when (0<credits<90)

do code=02;

date=datetime();

output myv9lib.problems;

end; **5**

otherwise

output myv9lib.class;

end;

run; **6**

The following example shows how to print the files created previously. The MYV9LIB.CLASS contains the observations from STUDENT that were processed without errors. The data file MYV9LIB.PROBLEMS contains the observations that contain errors.

If the data frequently changes in the source data file STUDENT, there would be different effects on the returned values in the SAS view and the SAS data file:

* New records, if error free, that are added to the source data file STUDENT between the time you run the DATA step in the previous example and the time you execute PROC PRINT in the following example, will appear in the SAS view MYV9LIB.CLASS.
* On the other hand, if any new records, failing the error tests, were added to STUDENT, the new records would not show up in the SAS data file MYV9LIB.PROBLEM, until you run the DATA step again.

A SAS view dynamically updates from its source files each time it is used. A SAS data file, each time it is used, remains the same, unless new data is written directly to the file.

filename student '*external-file-specification*';

libname myv9lib '*SAS-library*'; **7**

proc print data=myv9lib.class;

run; **8**

proc print data=myv9lib.problems;

format date datetime18.;

run; **9**

|  |  |
| --- | --- |
|  | [1]Reference a library called MYV9LIB. Tell SAS where a file that associated with the fileref STUDENT is stored. |
|  | [2]Create a data file called PROBLEMS and a SAS view called CLASS and specify the column names for both data sets. |
|  | [3]Select the file that is referenced by the fileref STUDENT and select the data in character format that resides in the specified positions in the file. Assign column names. |
|  | [4]When data in the column NAME, MAJOR, or CREDITS is blank or missing, assign a code of **01** to the observation where the missing value occurred. Also assign a SAS datetime code to the error and place the information in a file called PROBLEMS. |
|  | [5]When the amount of credits is greater than zero, but less than ninety, list the observations as code **02** in the file called PROBLEMS and assign a SAS datetime code to the observation. |
|  | [6]Place all other observations, which have none of the specified errors, in the SAS view called MYV9LIB.CLASS. |
|  | [7]The FILENAME statement assigns the fileref STUDENT to an external file. The LIBNAME statement assigns the libref MYV9LIB to a SAS library. |
|  | [8]The first PROC PRINT calls the SAS view MYV9LIB.CLASS. The SAS view extracts data on the fly from the file referenced as STUDENT. |
|  | [9]This PROC PRINT prints the contents of the data file MYV9LIB.PROBLEMS. |

**PROC SQL Views**

A PROC SQL view is a PROC SQL query-expression that is given a name and stored for later use. When you use a PROC SQL view in a SAS program, the view derives its data from the data sets (often referred to as tables) or SAS views listed in its FROM clause. The data that is accessed by the view is a subset or superset of the data in its underlying data sets or SAS viewspper.

A PROC SQL view can read or write data from:

* DATA step views
* SAS data files
* other PROC SQL views
* SAS/ACCESS views
* DB2, ORACLE, or other DBMS data.

For complete documentation on how to create and use PROC SQL views, see **Base SAS Procedures Guide**.

For information on using PROC SQL views created in an earlier release, see [SAS 9.2 Compatibility with SAS Files from Earlier Releases](http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/a001051126.htm)

**Comparing DATA Step and PROC SQL Views**

To help you decide between a DATA step view and a PROC SQL view, consider the characteristics of each type of SAS view:

* DATA step views
  + DATA step views are versatile because they use DATA step processing, including DO loops and IF-THEN/ELSE statements.
  + DATA step views do not have update capability; that is, they cannot directly change the data that they access.
  + There is no way to qualify the data in a DATA step view before using it. Therefore, even if you need only part of the data in your SAS view, you must load into memory the entire DATA step view and discard everything that you do not need.
* PROC SQL views
  + PROC SQL views can combine data from many different file formats.
  + PROC SQL views can both read and update the data that they reference.
  + PROC SQL supports more types of WHERE clauses than are available in DATA step processing and has a CONNECT TO component that enables you to easily send SQL statements and pass data to a DBMS by using the Pass-Through Facility.
  + You can also use the SQL language to subset your data before processing it. This capability saves memory when you have a large SAS view, but need to select only a small portion of the data contained in the view.
  + PROC SQL views do not use DATA step programming.
* **SAS/ACCESS Views**
* A SAS/ACCESS view is an interface view, also called a **view descriptor**, which accesses DBMS data that is defined in a corresponding **access descriptor**.
* Using SAS/ACCESS software, you can create an access descriptor and one or more view descriptors in order to define and access some or all of the data described by one DBMS table or DBMS view. You can also use view descriptors in order to update DBMS data, with certain restrictions.
* In addition, some SAS/ACCESS products provide a dynamic LIBNAME engine interface. If available, it is recommended that you use SAS/ACCESS LIBNAME statement to assign a SAS libref to your DBMS data because it is more efficient and easier to use than access descriptors and view descriptors. The SAS/ACCESS dynamic LIBNAME engine enables you to treat DBMS data as if it were SAS data by assigning a SAS libref to DBMS objects. Using a SAS/ACCESS dynamic LIBNAME engine means that you can use both native DATA step views and native PROC SQL views to access DBMS data instead of view descriptors.
* See [About SAS/ACCESS Software](http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/a001278864.htm) or the SAS/ACCESS documentation for your database for more information about SAS/ACCESS features.
* For information on using SAS/ACCESS view descriptors created in an earlier release, see [SAS 9.2 Compatibility with SAS Files from Earlier Releases](http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/a001051126.htm).
* **Note:**   Starting in SAS 9, PROC SQL views are the preferred way to access relational DBMS data. You can convert existing SAS/ACCESS view descriptors into PROC SQL views by using the CV2VIEW procedure, enabling you to use the LIBNAME statement to access your data. See the CV2VIEW Procedure in **SAS/ACCESS for Relational Databases: Reference**.  [cautionend]