

# CHAPTER 29

## SAS Data Views

<i>Definitions</i>	455
<i>DATA Step Views</i>	456
<i>Definition</i>	456
<i>Creating DATA Step Views</i>	456
<i>Recent Enhancements to Views</i>	457
<i>Examples</i>	457
<i>What Can You Do with a Data Step View?</i>	457
<i>Differences between DATA Step Views and Stored Compiled DATA Step Programs</i>	457
<i>Restrictions and Requirements</i>	458
<i>Performance Considerations</i>	458
<i>Example 1: Merging Data to Produce Reports</i>	458
<i>Example 2: Producing Additional Output Files</i>	459
<i>PROC SQL Views</i>	460
<i>SAS/ACCESS Views</i>	461
<i>Benefits of Using Data Views</i>	461
<i>When to Use Views</i>	462
<i>Comparing DATA Step and PROC SQL Views</i>	462

## Definitions

### SAS data view

is a SAS data set that uses descriptor information and data from other files. SAS data views allow you to dynamically combine data from various sources without using disk space to create a new data set. While a SAS data file actually contains data values, a SAS data view contains only references to data stored elsewhere. SAS data views are of member type VIEW. In most cases, you can use a SAS data view as though it were a SAS data file. There are two general types of SAS data views: *native* and *interface*.

### native view

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

### interface view

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

*Note:* Beginning in Version 7, you might be able to create native views that access DBMS data by using a SAS/ACCESS dynamic LIBNAME engine. See “SAS/ACCESS Views” on page 461, Chapter 33, “Accessing Data in a DBMS,” on page 487 or the SAS/ACCESS documentation for your DBMS for more information. △

## DATA Step Views

### Definition

DATA step view

is a native view that has the broadest scope of any SAS data view. It contains stored DATA step programs 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

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 SAS source program and to store the compiled code in the input DATA step view that is named in the option.

```
DATA view-name <data-set-name-1 <(data-set-options-1)>>
      <...data-set-name-n <(data-set-options-n)>> /
      VIEW=view-name <(<password-option><SOURCE=source-option>)>;
```

where

*view-name*

names a view that the DATA step uses to store the input DATA step view.

*data-set-name*

specifies a valid SAS name for the output data set created by the source program. The name can be a one-level name or a two-level name. You can specify more than one data set name in the DATA statement.

*data-set-options*

specifies optional arguments that the DATA step applies when it writes observations to the output data set.

*view-name*

names a view that the DATA step uses to store the input DATA step view.

*password-option*

assigns a password to a stored compiled DATA step program or a DATA step view.

*source-option*

specifies whether to save or encrypt the source code.

If the SAS data view already exists in a SAS data library and you use the same member name to create a new view definition using the same member name, then the old data view is overwritten.

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

---

## Recent Enhancements to Views

- SAS Version 8 has the capability to read views created by previous versions.
- Data views created by SAS Version 8 retain source statements. You can retrieve these statements using the DESCRIBE statement. See the following examples.

---

## Examples

- The following statements create a DATA step view named DEPT.A:

```
libname dept 'SAS---data---library';
data dept.a / view=dept.a;
    ... more SAS statements ...
run;
```

- The following statements create a DATA step view named BUDGET\_JAN:

```
data budget_jan / view=budget_jan;
    ... more SAS statements ...
run;
```

- The following example uses the DESCRIBE statement in a DATA step view to write a copy of the source code to the SAS log:

```
data viewname view=inventory;
    describe;
run;
```

For information about the DESCRIBE statement, see the *SAS Language Reference: Dictionary*.

---

## What Can You Do with a Data Step View?

You can:

- process directly 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 data views.

---

## Differences between DATA Step Views and Stored Compiled DATA Step Programs

DATA step views and SAS programs created using the Stored Program Facility 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 the Stored Program Facility, see Chapter 30, “Creating and Executing Stored Compiled DATA Step Programs,” on page 465.

---

## Restrictions and Requirements

*Do not expect global statements 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 data view, have no effect on the data 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 may differ from the intended execution.

---

## Performance Considerations

- *DATA step code executes each time that you use a view.* This may add considerable system overhead. In addition, you run the risk of having your data change between steps.
- *Depending on how many reads or passes on the data are required, processing overhead increases.*
  - When one 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 multiple passes are requested, the 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.

---

## 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 “MYV8LIB”, which merges the sales figures in the data file V8LR.CLOTHES with the sales figures in the data file V8LR.EQUIP. The data files are merged by date, and the value of the variable TOTAL is computed for each date.

```
libname myv8lib 'SAS-data-library';

data myv8lib.qtr1 / view=myv8lib.qtr1;
    merge v8lrclothes.clothes myv8lr.equip;
    by date;
    total = cl_v8lr + eq_v8lr;
run;
```

The following PROC print statement executes the view:

```
proc print data = myv8lib.qtr1;
run;
```

## Example 2: Producing Additional Output Files

In this example, the DATA step reads an external file named STUDENT, which contains student data, then writes observations that contain known problems to MYV8LIB.PROBLEMS. The DATA step also defines the DATA step view MYV8LIB.CLASS. The DATA step does *not* create a SAS data file named MYV8LIB.CLASS.

The FILENAME and the LIBNAME statements are both global statements and must exist outside of the code that defines the view, because 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
xyлотone   SCI   96
```

Here is the DATA step that produces the output files:

```
libname myv8lib 'SAS-data-library';
filename student 'external-file-specification'; ❶
data myv8lib.class(keep=name major credits)
    myv8lib.problems(keep=code date) / view=myv8lib.class; ❷
infile student;
    input name $ 1-10 major $ 12-14 credits 16-18; ❸
select;
when (name=' ' or major=' ' or credits=.)
    do code=01;
        date=datetime();
        output myv8lib.problems;
    end; ❹
when (0<credits<90)
    do code=02;
        date=datetime();
        output myv8lib.problems;
    end; ❺
otherwise
    output myv8lib.class;
end;
run; ❻
```

The following example shows how to print the files created previously. The data view MYV8LIB.CLASS contains the observations from STUDENT that were processed without errors. The data file MYV8LIB.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 the SAS data 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 data view MYV8LIB.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 MYV8LIB.PROBLEM, until you run the DATA step again.

A SAS data 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 myv8lib 'SAS--data--library'; ❶
proc print data=myv8lib.class;
run; ❷
proc print data=myv8lib.problems;
    format date datetime18.;
run; ❸
```

- ❶ Reference a library called MYV8LIB. Tell SAS where a file that associated with the fileref STUDENT is stored.
- ❷ Create a data file called PROBLEMS and a data view called CLASS and specify the column names for both data sets.
- ❸ 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.
- ❹ When data in the columns 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”.
- ❺ 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.
- ❻ Place all other observations, which have none of the specified errors, in the SAS data view called MYV8LIB.CLASS.
- ❼ The FILENAME statement assigns the fileref STUDENT to an external file. The LIBNAME statement assigns the libref MYV8LIB to a SAS data library.
- ❽ The first PROC PRINT step calls the data view MYV8LIB.CLASS. The data view extracts data on the fly from the file referenced as STUDENT.
- ❾ This PROC PRINT step prints the contents of the data file MYV8LIB.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 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 set(s) or view(s).

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 the *SAS Procedures Guide*.

---

## 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*.

Version 6 of SAS/ACCESS software enabled you to create an access descriptor and one or more view descriptors to define and access some or all of the data described by one DBMS table or DBMS view. You could also use view descriptors to update DBMS data, with certain restrictions. Descriptors continue to work in SAS software if they were available for your DBMS in Version 6.

Beginning in Version 7, some SAS/ACCESS products now 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. This means that you can use both native DATA step views and native PROC SQL views to access DBMS data instead of view descriptors.

SAS Institute continues to support SAS/ACCESS views that were created for previous versions of SAS.

See Chapter 33, “Accessing Data in a DBMS,” on page 487 or the SAS/ACCESS documentation for your database for more information about SAS/ACCESS features.

---

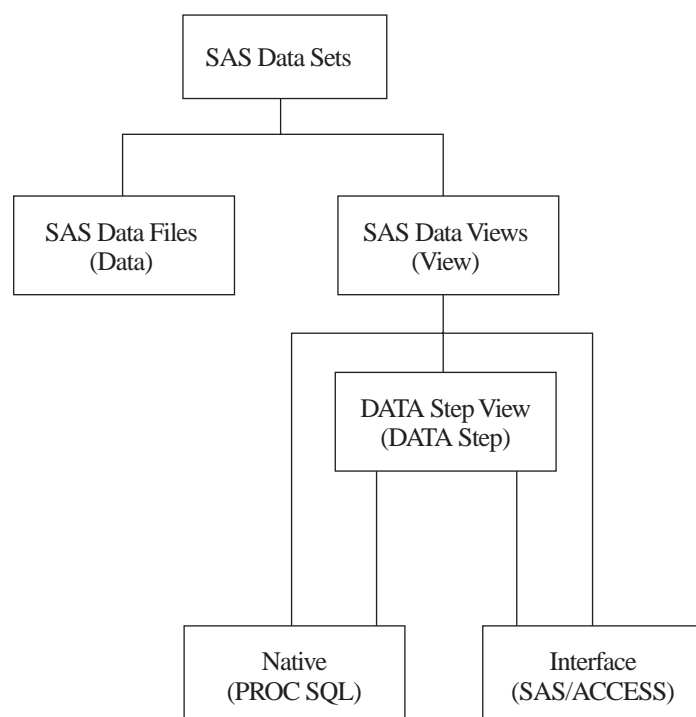
## Benefits of Using Data Views

Data views provide the following benefits:

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

**Figure 29.1** Native and Interface SAS Data Views



You can use views in the following ways:

- as input to other DATA steps or PROC steps
- to migrate data to SAS data sets 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.

---

## When to Use Views

The first question you'll probably ask is: "What is better for my purposes, a SAS data set, or a SAS data view?" Here are a few things to consider:

- Data sets use additional disk space; data views use additional processing time.
- Data set variables can be sorted and indexed prior to use; data views must process data in its existing form during execution.

---

## Comparing DATA Step and PROC SQL Views

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



- DATA step views
  - DATA step views are very versatile because they can harness the power of SAS DATA step processing, including DO loops and IF-THEN-ELSE statements.
  - DATA step views do not have write capability, that is, they cannot directly change the data that they can access.
  - There is no way to qualify the data in a DATA step view prior to using it therefore, even if you need only part of the data in your data view, you must load into memory the entire DATA step view and discard everything that you don't 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 allows you to easily send SQL statements and pass data to a DBMS by using the Pass-Through Facility.
  - You can also use the power of the SQL language to subset your data prior to processing it. This saves memory when you have a large 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.



The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS Language Reference: Concepts*, Cary, NC: SAS Institute Inc., 1999. 554 pages.

**SAS Language Reference: Concepts**

Copyright © 1999 SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-441-1

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, by any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute, Inc.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of the software by the government is subject to restrictions as set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, November 1999

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.® indicates USA registration.

IBM, ACF/VTAM, AIX, APPN, MVS/ESA, OS/2, OS/390, VM/ESA, and VTAM are registered trademarks or trademarks of International Business Machines Corporation. ® indicates USA registration.

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

The Institute is a private company devoted to the support and further development of its software and related services.