Using SAS/ACCESS® Views to Retrieve DBMS Tables

Tischa A. Agnessi, Mayo Clinic, Rochester, MN

Abstract

A tutorial approach will be used to illustrate how useful SAS/ACCESS views can be in an environment with many types of frequently accessed data sources. Views can be considered to be virtual data sets. They require minimal storage and can be used to obtain up-to-date results as well as to create permanent SAS data sets.

Prior to accessing an existing Database Management System (DBMS) table, two descriptor files must be created to define a SAS/ACCESS view of the DBMS table. Special emphasis will be placed on building the Access Descriptor and the View Descriptor using the ACCESS windows of the Display Manager. By using the ACCESS windows as a building block, it requires no knowledge of the SQL language syntax, and requires minimal typing. We will also examine the use of the WHERE clause and how it increases efficiency and lowers cost.

Introduction

SAS® views can be a powerful tool when frequently accessing different DBMS. A view can be considered a virtual data set. In order to use this virtual data set an Access Descriptor and a View Descriptor must be constructed. The goal of this paper is to:

I. Inform the user of the construction of SAS views.

II. Show how SAS views can be an efficient way of data management.

I. Construction of SAS views

SAS/ACCESS can retrieve information from various DBMS such as Oracle, Ingres, Sybase, and DB2. For these DBMS the Access Descriptor window will be accessed slightly different. All that is needed to be known is where the database is located, and what the table name is called. The rest of the screens should be the same.

The following SAS view will be constructed using SAS/ACCESS on a UNIX platform. The data will be accessed from an Ingres DBMS.

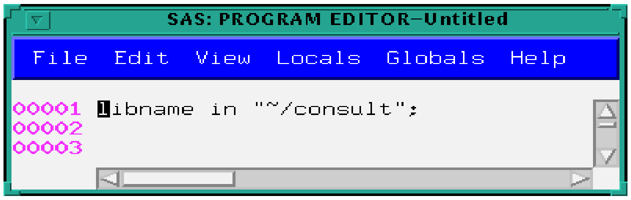
Please note:

*The SAS "*end*" command will enable the user to back out of virtually any window or screen.

*Submit the *libname* statement prior to going into the SAS/ACCESS windows.

*Be consistent with the *libref* and *libname* statements.

Submit the *libname* statement to define the targeted data file and/or disk:



```
libname in "~/consult";
```

Connecting to SAS/ACCESS

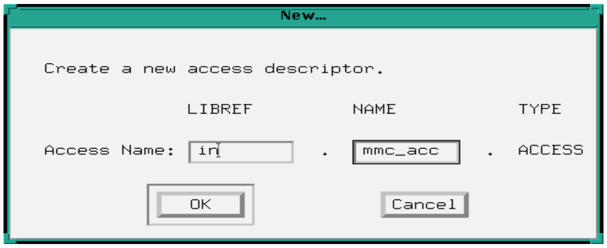
In order to connect to SAS/ACCESS use the pmenus in the Program Editor window:

Globals-->
  Access-->
    Access database files

Now use these pmenus in the DBMS Access window to name your Access Descriptor:
File-->New

Naming the Access Descriptor

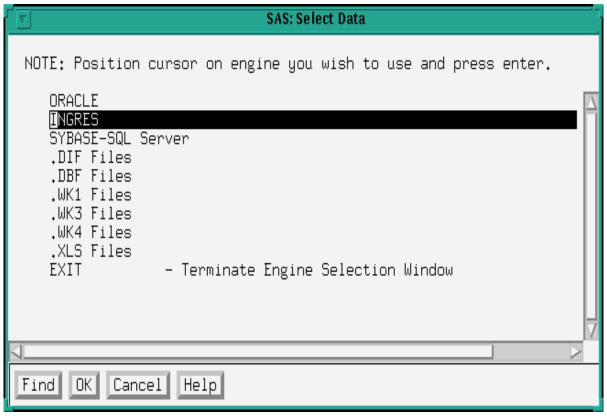
This window allows the Access Descriptor to be named *MMC_ACC.*



```
Create a new access descriptor.

              LIBREF          NAME           TYPE

Access Name:  in          .   mmc_acc    .   ACCESS
```

Choosing the Database Engine

The SAS: Select Data window provides a list of the available DBMS to access. For this example information from Ingres will be accessed. Therefore, the word INGRES will be clicked.

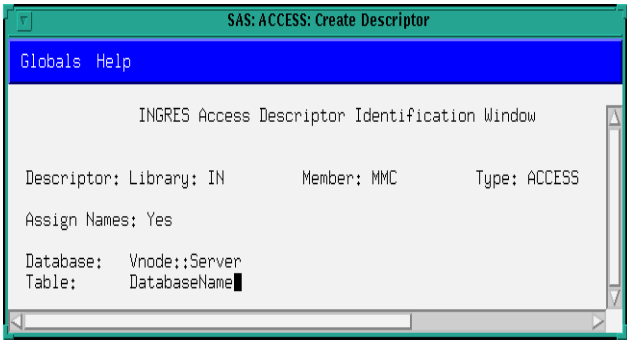


```
NOTE: Position cursor on engine you wish to use and press enter.

   ORACLE
   INGRES
   SYBASE-SQL Server
   .DIF Files
   .DBF Files
   .WK1 Files
   .WK3 Files
   .WK4 Files
   .XLS Files
   EXIT          - Terminate Engine Selection Window
```

Defining the Location of the Database

By filling in the Ingres vnode, server, and database name, the system knows where to look for the data. For other DBMS, such as Oracle, the user would need to know the driver, node, and database name.

To get the SAS formats and variable names to come across, be sure to type *YES* in the *Assign Names* area.



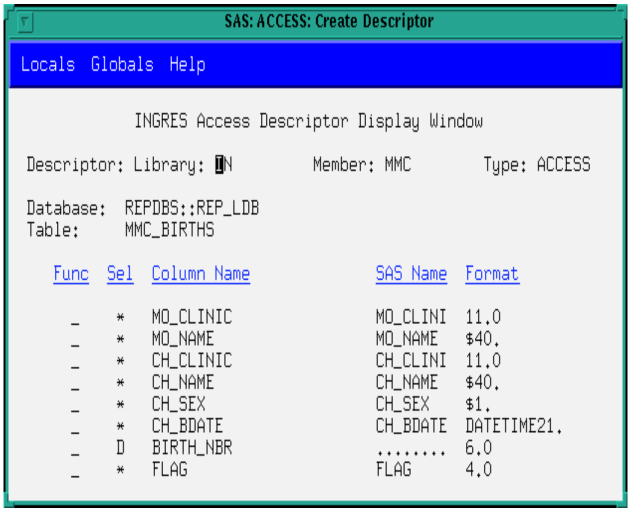
```
          INGRES Access Descriptor Identification Window

Descriptor: Library: IN        Member: MMC        Type: ACCESS

Assign Names: Yes

Database:    Vnode::Server
Table:       DatabaseName
```

Manipulating SAS Variables

The Access Descriptor window allows the user to make changes to the variable names without changing them on the actual database, and select or deselect variables. If the variables have not been assigned SAS names, check the previous screen and make sure this is correct:

Assign Names: YES

To make changes to the variable names just type over the SAS name and hit <return>. In this example there will be no changes to any variable names, but BIRTH_NBR will be deselect from the variable list. To do so type a “d” under the Func column next to the variable. By hitting <return> the screen reappears and there will be a “D” in the Sel column. To keep all of the variables without any adjustments just hit <return>. The (*)’s represent the variables that have been selected to be sent across from the DBMS to the SAS system.

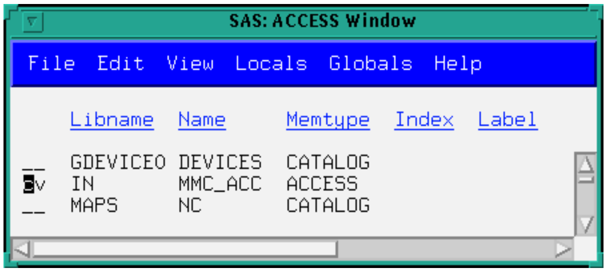


```
            INGRES Access Descriptor Display Window

Descriptor: Library: IN        Member: MMC        Type: ACCESS

Database:   REPDBS::REP_LDB
Table:      MMC_BIRTHS

   Func  Sel  Column Name            SAS Name  Format

    _     *   MO_CLINIC              MO_CLINI  11.0
    _     *   MO_NAME                MO_NAME   $40.
    _     *   CH_CLINIC              CH_CLINI  11.0
    _     *   CH_NAME                CH_NAME   $40.
    _     *   CH_SEX                 CH_SEX    $1.
    _     *   CH_BDATE               CH_BDATE  DATETIME21.
    _     D   BIRTH_NBR              ........  6.0
    _     *   FLAG                   FLAG      4.0
```

Back out of the SAS:ACCESS: Create Descriptor window by using the SAS “*end*” command until you get to the DBMS Access window.

Browsing the Access Descriptor

At the DBMS Access window look for the newly created Access Descriptor. In this example the Access Descriptor is called IN.MMC_ACC.



```
    Libname  Name     Memtype  Index  Label

__  GDEVICE0 DEVICES  CATALOG
cv  IN       MMC_ACC  ACCESS
__  MAPS     NC       CATALOG
```

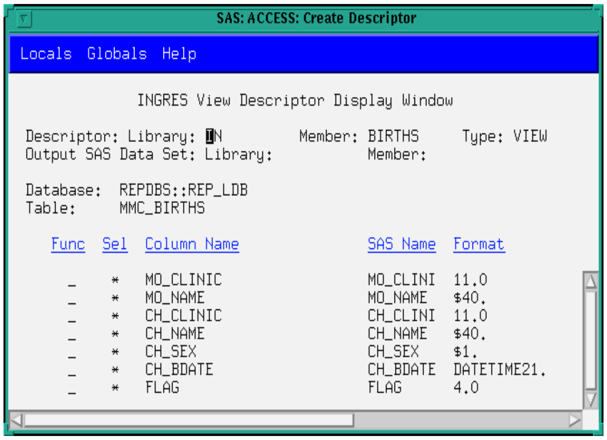
To see the list of commands to enter on the line type a “?”.

Examples:
B Browse (not valid for ACCESS)
BD Browse descriptor
(for ACCESS and view only)
CV Create view (for ACCESS only)

Creating a View Descriptor

To create a View Descriptor a “cv” needs to be typed on the line before the Access Descriptor. This brings up another Create Descriptor window. The View Descriptor window allows the user to create a permanent SAS data set or view data set. To get an up-to-date report each time the SAS program is run, a view data set needs to be created.

In this example a **view** will be created to obtain the latest information available. Therefore, fill in the LIBRARY and MEMBER on the **first** line of the Create Descriptor window.



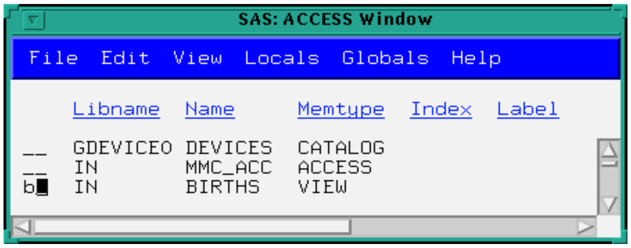
```
                INGRES View Descriptor Display Window

Descriptor: Library: IN         Member: BIRTHS     Type: VIEW
Output SAS Data Set: Library:           Member:

Database:  REPDBS::REP_LDB
Table:     MMC_BIRTHS

   Func  Sel  Column Name               SAS Name  Format

    _     *   MO_CLINIC                 MO_CLINI  11.0
    _     *   MO_NAME                   MO_NAME   $40.
    _     *   CH_CLINIC                 CH_CLINI  11.0
    _     *   CH_NAME                   CH_NAME   $40.
    _     *   CH_SEX                    CH_SEX    $1.
    _     *   CH_BDATE                  CH_BDATE  DATETIME21.
    _     *   FLAG                      FLAG      4.0
```

To create a **permanent** SAS data set fill in the LIBRARY and MEMBER on the **second** line in the window.

Type an “s” next to each variable under the Func column to select it from the database. Then hit <return>. There should now be (*)’s under the Sel column to show that those variables have been selected.

Back out of these windows until the DBMS Access window is visible.

To look for the newly created view data set, type a “b” in the blank before the new view member.



```
    Libname  Name     Memtype  Index  Label

__  GDEVICE0 DEVICES  CATALOG
__  IN       MMC_ACC  ACCESS
b   IN       BIRTHS   VIEW
```

For example, this is what the BIRTHS data looks like in Ingres through the SAS: FSBROWSE window:

MO_CLINI:	1234567
MO_NAME:	Mrs. John Smith
CH_CLINI:	7654321
CH_NAME:	Jill Marie Smith
CH_SEX:	F
CH_BDATE:	01JAN1980:00:00:00
FLAG:	0

You can move forward and backward in the FSBROWSE window. To exit this screen use the SAS "*end*" command.

WHERE Clause

The View Descriptor window allows a WHERE clause to be used. Using WHERE conditions in the View Descriptor increase efficiency by decreasing CPU time. The WHERE statement allows only information from the DBMS within the WHERE criteria to pass through to SAS. To build the WHERE conditions use the pmenus:

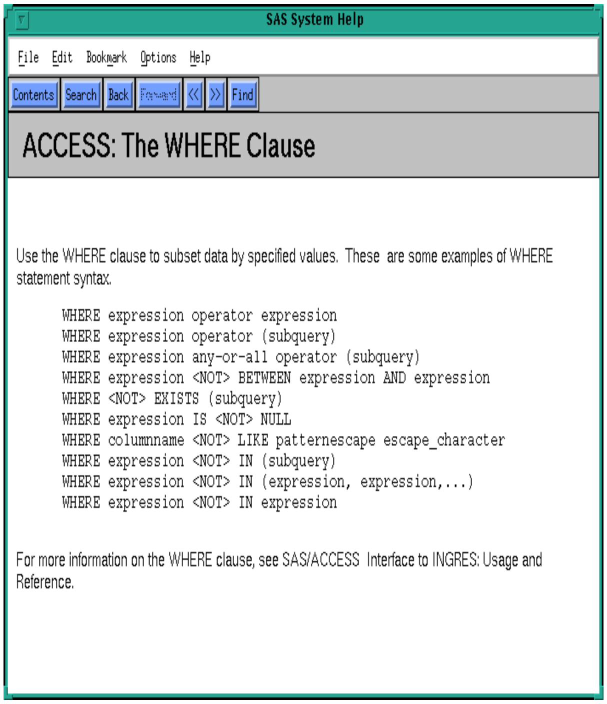
```
  Local -->
    Subset
```

A mini-program editor appears for the WHERE criteria to be submitted. For example, a WHERE clause would look like this:

WHERE FLAG=0

*note: there is no semicolon used

Here is a clip from SAS on-line help about the WHERE clause:


SAS System Help
File Edit Bookmark Options Help
Contents Search Back Forward << >> Find

ACCESS: The WHERE Clause

Use the WHERE clause to subset data by specified values. These are some examples of WHERE statement syntax.

```
WHERE expression operator expression
WHERE expression operator (subquery)
WHERE expression any-or-all operator (subquery)
WHERE expression <NOT> BETWEEN expression AND expression
WHERE <NOT> EXISTS (subquery)
WHERE expression IS <NOT> NULL
WHERE columnname <NOT> LIKE patternescape escape_character
WHERE expression <NOT> IN (subquery)
WHERE expression <NOT> IN (expression, expression,...)
WHERE expression <NOT> IN expression
```

For more information on the WHERE clause, see SAS/ACCESS Interface to INGRES: Usage and Reference.

Views as Virtual Data Sets

A view allows an ACTIVE database to be accessed and used as a virtual data set. To store a FROZEN copy of the view data, create a SAS data set from the view. Manage the view as if it were like any other SAS data set.

For example:

```
  DATA NEWDATA;
    SET IN.BIRTHS;
```

MMC_BIRTHS on Ingres is being viewed by our View Descriptor IN.BIRTHS. The data set NEWDATA acts like a normal SAS data set.

II. View Advantages

Here are a few ways views can help work areas:

✔ Ability to handle multiple types of data sources.
- SAS Data sets
- DB2 Tables
- Sequential Files

✔ User is able to examine contents of data source.
- Variables and attributes
- Attach formats and labels

✔ Specifies extract criteria and generates code to extract data.
- Allow specification of WHERE conditions

✔ Does not suffer from out-of-date data.

✔ No knowledge of SQL needed.

✔ Reduces the amount of typing for longer requests.

Conclusion

By constructing views the data becomes more visible for others to observe. Views are basically a set of directions for the DBMS to follow. They can be designed to look like customized databases without redundancy or the concern of storage space.

SAS can be a primary application used for data warehousing but not all databases are SAS data sets. Views are able to reduce or eliminate the amount of knowledge needed about the table or the table's underlying structure. Thus, SAS/ACCESS in an effective and efficient way for users to manage databases.

Acknowledgments

I would like to thank Priscilla VanGrevenhof for taking the time to show me how to access data using the SAS Access and View windows via point and click methods.

References

Wilson, Steven A. (1996), "Using SAS Views for Data Source Visualization", Proceedings of the Twenty-first SAS Users Group International Conference, pps. 484-493.

Lafler, Kirk Paul (1996), "Frame Your View of Data with SQL Procedure", Proceedings of the Twenty-first SAS Users Group International Conference, pps. 494-497.

Author's Address

Tischa A. Agnessi
Mayo Clinic
200 1st Street SW
Rochester, MN 55905-0001
(507) 284-5309

e-mail: agnessi@mayo.edu

Any comments, questions, or details of your experiences are welcome.