Database Interfaces under the Version 6 Engine Architecture

Emily P. Wallace, SAS Institute Inc., Cary, NC

For the last several SUGI conferences, you have been hearing about the Multiple Vendor Architecture (MVA) and the multiple engine concepts of Version 6 of the SAS® System that make data access transparent. Thus far, we have implemented four database engines and development is underway on other database engines, so you can see that the engine architecture has become a reality. These new interfaces provide much more functionality and flexibility than their counterparts in Version 5 of the SAS System.

The Version 5 interfaces are generally procedure-style interfaces with each database having its own set of procedures that bring the data from the Database Management System (DBMS) into a SAS data set. You can run the SAS procedures against this copy of the DBMS data to perform your analyses or produce your reports. If there are procedures that update the database, they generally use SAS data sets as input for the updates. Both the extract and update procedures are very flexible in allowing you to work with a subset of your data. However, the only way to keep the SAS data you have extracted from your database current is to periodically extract the DBMS data. For many people, this is fine, but there are others who want to access the DBMS data directly from SAS programs without making a copy of the data. This is one of the primary reasons for the development of the multiple engine architecture.

The primary goal of the multiple engine architecture is to provide transparent access to data that are not stored in SAS data sets. What this means is that you can use the same SAS programming statements on data regardless of where the data are stored. With the multiple engine architecture, you can execute DATA and PROC steps directly against the DBMS data instead of executing against a copy of the data. Now you use the same tool for analysis, graphics, and reporting needs. A long term benefit of this new architecture is that as the SAS System adds new features and additional capabilities, these new features will be immediately available to the DBMS engine users as well.

The database engine makes the direct access possible. The job of the database engine is to interpret any request from the SAS supervisor, translate it into the appropriate DBMS request, and return data or status information to the SAS supervisor in the correct SAS format. For example, if a procedure asks the SAS supervisor to read a record, the database engine will build a DBMS read request, pass it to the database system, and receive the data that was read. Then the engine must translate the DBMS data into the format that the SAS System expects and pass the translated data to the SAS supervisor who passes it on to the procedure.

This description of a database engine might make you believe that it is a relatively simple task to bring DBMS data into the SAS environment but that is not entirely true. There are some problems that must be handled to make the process work smoothly.

One of the most visible problems is the difference in the rules for SAS names and the rules for DBMS names. In many cases, the DBMS names are longer than the SAS names and contain characters that would not be valid in SAS variable names. Also, there is no syntax on the procedure statements to specify the name of the database that contains the data you want to use. Many DBMSs also have options that must be set by the calling program before accessing the database management system. In many cases, the databases contain large amounts of data and you want to work with a small portion of the data so you need a way to subset it. Another significant problem is the security of the data and the impact on the performance for other users of the database system.

In order to solve these problems, we have created the ACCESS procedure. Its job is to capture all of the information needed to access a database. This information can include such things as the name of the database, the password and the other options needed to uniquely identify the database. It also allows you to specify the elements of the database you want to work with and give them SAS names that will be meaningful to you. You can specify a WHERE clause to subset the records retrieved so that you work with only the portion of the database that you need.

Before we look at PROC ACCESS in more detail, we need to define two new terms, master descriptor and subdescriptor. A master descriptor is a member of a SAS data library that contains a layout of the database and all DBMS information needed to access its data. A master descriptor can point to both tables and views in a relational database system. A subdescriptor or view is based on a master descriptor and contains a list of elements in the database that will be used by the SAS program, the SAS names and formats associated with the elements, and any subsetting or ordering information that is valid for the DBMS. The subdescriptor or view is used by the procedures or DATA step to access the DBMS data.

You must use PROC ACCESS to create master descriptors and subdescriptors. If you want to keep the descriptors you build, you must supply a LIBNAME statement pointing to a SAS data library where you want the descriptors to be written. If you do not supply a LIBNAME statement, the descriptors will be written to the WORK library and will be temporary. To invoke the procedure, type

PROC ACCESS;

In the program edit window and submit the statement. You see the ACCESS panel that displays the contents of all SAS data libraries you have defined to your SAS session. Master descriptors are identified by their member type (Mmtype) of ACCESS and subdescriptors are identified by their type of VIEW. Views created by the SQL procedure are also identified by the type of VIEW and will also be displayed on the ACCESS PANEL.
To create a new master descriptor, type the following on the command line of the ACCESS PANEL:

```
CREATE libname.memname.ACCESS
```

where `memname` is the member you want to create and `libname` points to the library where you want the member stored. If you have more than one database engine installed on your CPU, you will see a list of the engines you have and you should choose the one needed for the master descriptor you are creating. If you have only one database engine installed, you will bypass this panel. The next step is to identify the name of the database or table to be described by this master descriptor. Once this information is supplied, you see the MASTER DESCRIPTOR DISPLAY PANEL, which lists the names of the elements of the database. If you want to enforce the use of standard names for all subdescriptors created from this master descriptor, type in the SAS names you want to use. If you do not supply SAS names, then subdescriptors can use SAS names of their choosing. If you do not want an element to be available for use in any views, type an `N` in the selection field and the element will be marked as non-display and cannot participate in any views. When you enter the END command, the new master descriptor is created and can be used to create subdescriptors.

To create a subdescriptor from the master descriptor you just created, type a `C` beside the name of the master descriptor on the ACCESS PANEL. You see the SUBDESCRIPTOR DISPLAY PANEL, which lists all of the database elements that are eligible for inclusion into the view. If the master descriptor has assigned SAS names for the elements in the database, the SAS names are displayed and cannot be changed. To select elements, enter an `S` in the FUNC field for each element you want. If the SAS name field is blank, you may fill in a name to use or you can let the procedure assign the names.

If you want to define subsetting or ordering information, type `WHERE` on the command line. The SELECTION CRITERIA ENTRY PANEL is displayed and you can type in where clauses or ordering clauses that are valid for the DBMS you are accessing. When you exit the SUBDESCRIPTOR DISPLAY PANEL, the new subdescriptor is written and its name is displayed on the ACCESS panel. Now your new view is ready for use by SAS procedures and the DATA step. If you want to validate the view before leaving the ACCESS PANEL, you can invoke the FSEDIT, FSBATCH, or FSVIEW procedures to see the data from the DBMS.

Now that your view is built, you can use it in SAS programming statements that will access data in your database. The program-
mlng statements are the same statements that you would use to access SAS data sets; the only difference is that your DATA= option points to a subprocess instead of a physical data library. If your database descriptors are in a permanent SAS data set, you must use a LIBNAME statement to point to the library. If they are in a temporary library, no LIBNAME statement is required.

```
LIBNAME DB "BASENAME.DESCRIB.LIBRARY";
PROC PRINT DATA=DB.INPUT;
RUN;
```

PROC ACCESS and the descriptors are powerful tools because they allow you to work with the DBMS data without a detailed knowledge of database concepts and terminology. Once the descriptors are set up, they remain valid until the database structure changes. Therefore, there is very little maintenance required. The descriptors can be used as a security mechanism to control both the elements and the records that you can see. The master descriptors and the views can be kept in separate data libraries so the database administration group can control the building of master descriptors and subprocesses and provide access to only the information in the databases that you need.

There are probably some of you who are wondering why we changed the way we interface to databases. For a variety of reasons, people preferred to extract data from the DBMS into a SAS data set and work with the copy of the data. In Version $8$ you can still do the extracts, and in some cases this approach is better than doing direct access from procedures and the DATA step. If you are going to use the same data in many procedures in the same SAS session, it would probably be advantageous to extract the data into a SAS data set. SAS data sets are organized in a manner that provides optimum performance for SAS procedures and the DATA step so procedures using SAS data set input will frequently take less CPU time than procedures that use views to access DBMS information.

Another situation where you might prefer to do an extract of the DBMS data is when your database is large and you want to read a large portion of the records. If the database, being shared by other users on an online system, your program might have a negative impact on the system performance and cause the other users to experience slow response time. The Database Administration group might suggest that you do extracts of the DBMS data into SAS data sets for security reasons also. You should carefully evaluate the types of SAS programs you will use to access the DBMS data and decide whether to use direct access or database extracts based on the programming tasks and your company's guidelines.

PROC ACCESS solves many problems for SAS users but there are some situations that it still does not handle. If your data are in a database or file system that does not contain descriptive information as a part of the database system, such as IMS, VSAM, tape files, or other external files, you might want to use PROC ACCESS and an engine to work with your data. One of the things we will be working on in the future is an enhancement to PROC ACCESS that will let you define the file layouts so that we can use that information to build the master descriptors. This feature will also enable you to create new databases using a database layout you define. For example, you might want to create a new ORACLE table using the NEW= option in PROC FSEDIT. Currently this is not supported, but in the future you will be able to define a new ORACLE table using PROC ACCESS to build a master descriptor and then create the table using PROC FSEDIT.

Another enhancement planned for a future release is the ability to build a database view containing information from more than one database of the same DBMS type when the database system can do the joins efficiently. Since PROC SQL already creates views that join two of our database views, we will let PROC SQL do the joins of database views when the database cannot do the joins itself.

The database engines that we currently have implemented are for DB2 under MVS, SQL/DS under CMS, ORACLE and Rob/VMS under VMS. These products are all part of the SAS/ACCESS family of interface products and they will be available with Release 8.08 of the SAS System.

Future engines will include interfaces to all currently supported database systems (IMS, DL/I, SYSTEM 2000, Data Management Software, IDMS/R, DATACOM/DB$ and ADABAS$) plus other databases you request. You will also see the existing engines ported to other platforms when there is adequate interest in them.

As you can see, the multiple engine architecture provides you with a powerful and flexible interface to your database system. You can write SAS programs to access your DBMS data without knowing specific DBMS terminology or DBMS programming concepts. You use the same programming statements whether your data are in SAS data sets or database systems. Also you can choose between direct access and extracts based on your programming tasks.

SAS, SYSTEM 2000 and SAS/ACCESS are registered trademarks of SAS Institute Inc., Cary, NC, USA

ORACLE is a registered trademark of Oracle Corporation.

Rob/VMS and VMS are trademarks of Digital Equipment Corporation.

DATACOM/DB is a registered trademark of Computer Associates International.

ADABAS is a registered trademark of Software AG.

DB2 and SQL/DS are trademarks of International Business Machines Corporation.

IDMS/R is a registered trademark of Cullinet Software, Inc.