

SAS/ACCESS® Software: Something Old, Something New

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

SAS/ACCESS software came into existence in Version 5 as a method of extracting data from the databases of some of the major database vendors into SAS® data sets. In Release 6.06 these capabilities were extended to include direct access to some of the databases using the multiple engine architecture. Now that Release 6.07 is being released for the MVS, CMS, and VMS operating systems, these interfaces have undergone changes and new interfaces have been added. This paper will describe the enhancements to the "old" Release 6.06 SAS/ACCESS products, introduce the "new" Release 6.07 SAS/ACCESS products, and briefly mention some of the interfaces under development for later releases.

The SAS/ACCESS interfaces available under Release 6.06 are DB2 on MVS, SQL/DS® on CMS, ORACLE® on VMS, AOS/V5, and Primos®, and Rdb/VMS® on VMS. These products contain the ACCESS procedure, which creates descriptors, a database engine that provides transparent access to database data from SAS procedures and the DATA step, and the DBLOAD procedure, which creates and loads a database table based on SAS data set input.

In Release 6.07 all of these interfaces have been significantly enhanced. The most frequently requested enhancement in the database products has been the ability to create descriptors in batch. In Release 6.07 PROC ACCESS has been modified to allow creation of access and view descriptors in batch and interactive line mode for all four interface products. A statement-style syntax allows you to supply table and database names, provide subsetting information, select individual columns, and modify SAS names and formats. Here is a simple example of the statements you would use.

```
PROC ACCESS DBMS=DB2;
CREATE WORK.CUST.ACCESS;
TABLE=SASDEMO.CUSTOMERS;
CREATE WORK.CUST.VIEW;
SELECT ALL;
RUN;
```

This example creates an access descriptor for the SASDEMO.CUSTOMERS table on the default DB2 system. It also creates a view descriptor containing all columns in the access descriptor.

The following example shows the syntax for creating an access descriptor and two view descriptors in the same execution of PROC ACCESS. The view descriptors use the SELECT statements to subset the columns in the table and the SUBSET statement to subset the rows. It also renames one column and changes the format of another column.

```
PROC ACCESS DBMS=DB2;
CREATE WORK.CUST.ACCESS;
TABLE=SASDEMO.CUSTOMERS;
CREATE WORK.MCCUST.VIEW;
LIST ALL;
SELECT 2 3 4 5;
RENAME 8 STREET;
FORMAT ZIPCODE 5.;
SUBSET WHERE STATE='NC';
CREATE WORK.ILCUST.VIEW;
SELECT 2 3 4 5;
RENAME 8 STREET;
FORMAT ZIPCODE 5.;
SUBSET WHERE STATE='IL';
RUN;
```

You can create many different access and view descriptors in the same execution of PROC ACCESS. The access descriptors can refer to different tables, and the view descriptors can select only the

columns and rows that they need. You can store both the access and view descriptors in either permanent or temporary SAS data libraries.

Another new feature added to the "old" SAS/ACCESS products is the passthru feature of the SQL procedure. This new feature is available for the interfaces to DB2, SQL/DS, ORACLE, and Rdb/VMS. The passthru feature allows you to supply statements to PROC SQL that are sent directly to the database for execution. This allows you to optimize your PROC SQL queries by letting the database handle the portions that it can optimize such as joining two tables from the same database system and by letting PROC SQL handle the portions that the database cannot such as joining two tables from different databases.

This is a simple example of using the passthru feature of PROC SQL to send a simple query to DB2.

```
PROC SQL;
SELECT * FROM CONNECTION TO DB2 (SELECT * FROM SASDEMO.CUSTOMERS);
QUIT;
```

This example selects all data from a DB2 table called SASDEMO.CUSTOMERS. It does an implicit connection to the DB2 system identified by the DB2SSID= system option. If you wanted to explicitly connect to a specific DB2 system, you would use the CONNECT statement.

```
PROC SQL;
CONNECT TO DB2(SSID=DB2T);
SELECT * FROM CONNECTION TO DB2 (SELECT * FROM SASDEMO.CUSTOMERS);
DISCONNECT FROM DB2;
QUIT;
```

This example connects to the DB2 subsystem called DB2T and performs the same query as the previous example. You can omit the DISCONNECT statement, and DB2 will be disconnected at the termination of PROC SQL.

The passthru feature provides an efficient way of joining two or more tables belonging to the same database system. In the next example our database PERSONNEL has three tables, EMPLOYEES, JOB_HISTORY, and JOBS, which contain employee and job information. To print a list of names and job titles, you must join the three tables. The following statements use PROC SQL passthru to let Rdb/VMS join the three tables.

```
PROC SQL;
CONNECT TO RDB (DATABASE=PERSONNEL);
SELECT * FROM CONNECTION TO RDB (SELECT FIRST_NAME, LAST_NAME,
JOB_TITLE FROM EMPLOYEES A, JOB_HISTORY B, JOBS C WHERE
A.EMPLOYEE_ID = B.EMPLOYEE_ID AND B.JOB_CODE = C.JOB_CODE);
QUIT;
```

You can use the passthru feature to retrieve information from a database table and join it with data from a SAS data set. The following example creates a SAS data set containing employees who have not turned in their tax withholding forms. It joins the ORACLE table EMP containing employee names and numbers with the SAS data set TAXINFO containing employee numbers and withholding information.

```
PROC SQL;
CONNECT TO ORACLE (USER=SCOTT PW=TIGER);
CREATE TABLE TAXFORMS AS SELECT * FROM CONNECTION TO ORACLE
(SELECT ENUM, ENAME FROM EMP) AS E, TAXINFO WHERE
E.ENUM = TAXINFO.ENUM AND TAXINFO.TAXCODE IS MISSING;
QUIT;
```

You can also issue non-SELECT statements using the passthru feature of PROC SQL. This allows you to grant or revoke privileges for tables, create or drop indexes, or use other types of dynamically executable statements. As with all other features of our database interfaces, the passthru feature totally honors database security and will not allow you to access or modify any tables for which you do not have proper authority.

The following example uses the passthru feature to grant read access to a table that you have previously created and to create a unique index on one of the columns of the table.

```
PROC SQL;
CONNECT TO SQLDS;
EXECUTE (CREATE UNIQUE INDEX SASBNO.BOOKNUM ON SASBNO.BOOKS
COLUMN BOOKS) BY SQLDS;
EXECUTE (GRANT SELECT ON SASBNO.BOOKS TO PUBLIC) BY SQLDS;
QUIT;
```

The SELECT and non-SELECT statements can be mixed in one execution of PROC SQL.

Those are the two major enhancements we have added for all four of the interface products that were available under Release 6.06. Another Release 6.07 enhancement for DB2 and SQL/DS users is the addition of support for the SAS/DB2™ and SAS/SQL-DS™ procedures of Version 5. Applications developed in Version 5 that use the DB2EXT, DB2LOAD, and DB2UTIL procedures of SAS/DB2 and the SQLEXT, SQLLOAD, and SQLUTIL procedures of SAS/SQL-DS can now be run under Release 6.07. These procedures have been rewritten to use the tools available under Release 6.07 and are provided for compatibility purposes while you are converting your applications to use the Version 6 tools. Support for updating DB2 and SQL/DS tables that have referential constraints has also been added.

For our ORACLE users, we have added the ability to access remote ORACLE tables using the SQL/Net™ product provided by Oracle Corporation. In your access and view descriptors, you specify the driver and database name to use for the connection to a remote ORACLE system. We have also changed the ORACLE engine to read multiple database rows in one FETCH so that we are reducing the amount of buffered I/Os needed to read an ORACLE table. This change can result in a performance improvement when you execute SAS procedures against large ORACLE tables.

That is an overview of the Release 6.07 enhancements to our "old" products. Now we will take a look at the "new" Release 6.07 products, which are SAS/ACCESS interfaces to ADABAS; CA-DATACOM/DB; and IMS/VS, all under MVS, and INGRES™ under VMS.

Because INGRES is a relational database system, the INGRES interface is very similar to the ORACLE and Rdb/VMS interfaces under Release 6.06. The product contains the full-screen interface to PROC ACCESS, a database engine that reads and updates the INGRES data, and PROC DBLOAD, which creates and loads INGRES tables based on SAS data set input. The distributed database system products INGRES/STAR, INGRES/NET, and INGRES Gateways are also supported to permit SAS procedures to access data in INGRES distributed and remote databases.

We have totally rewritten the SAS/ACCESS interface to ADABAS for Release 6.07. The Version 5 interface consisted of the ADBEXT procedure that extracted data from ADABAS files or NATURAL™ DDMs. In the Version 6 product, both types of files are still supported, and now there are direct read and update capabilities from SAS procedures and the DATA step. PROC SQL can also be used to join several ADABAS files or NATURAL DDMs. All field types except hyperdescriptors can be referenced in views, and all fields except superfields, superdescriptors, subfields, subdescriptors, and phonetics can be updated. A new window named the Occurs

Window expands repeating groups and multiple-value fields so that each occurrence can be selected and named individually. NATURAL SECURITY is also supported. There is no DBLOAD procedure for the ADABAS product.

We have also rewritten the CA-DATACOM/DB interface for Release 6.07. Like the ADABAS interface, it allows direct read and update of CA-DATACOM/DB tables. All field types such as compound fields, repeating fields, nested compound structures, and repeating structures are supported. The Occurs Window is also available to individually select multiple-value fields. The CA-DATACOM/DB interface also does not have a DBLOAD procedure.

The IMS/VS interface is probably the most challenging product we have developed thus far. The IMS interface enables you to read and update IMS data from SAS applications without having to understand IMS programming concepts and terminology.

Since IMS databases do not contain descriptive information about their record layouts, users must supply this information. To facilitate the definition process, a new window named the Descriptor Definition Window and statement-style syntax have been added to PROC ACCESS. The Descriptor Definition Window can be used to aid you in the process of creating an IMS access descriptor because it uses fill-in-the-blank fields to show you the information that is needed. The items that must be supplied are the database name as defined in the IMS Database Description (DBD), SAS format notation for the way the data are stored on disk, names of segment and search fields as defined in the DBD, key fields, and the number of times that multiple-value fields repeat.

For users who prefer to create their descriptors in batch, a statement-style syntax is available. The following example creates an access descriptor for the wine customers database using the statement-style syntax.

```
PROC ACCESS DBMS=IMS;
CREATE WORK.WINECUST.ACCESS;
DBD=WINECUST;
RECORD=CUSTOMER SG=CUSTOMER SL=130;
ITEM='CUSTOMER NUMBER' LV=3 DBF=PD0.0 SE=CUSTNUM KEY=0 SN=CUSTNU
ITEM=NAME LV=3 DBF=$30. SE=NAME;
ITEM=STREET LV=3 DBF=$30. SE=STREET;
ITEM=CITY LV=3 DBF=$20. SE=CITY;
ITEM=STATE LV=3 DBF=$2. SE=STATE;
ITEM=COUNTRY LV=3 DBF=$20. SE=COUNTRY;
ITEM='WINE PURCHASES' LV=3 DBF=$.2 SE=TOTPURCH SN=WINEPURC;
CREATE WORK.WINECUST.VIEW PSB=WCUSVDT;
SELECT ALL;
RUN;
```

The DBD statement specifies the name of the database, and there is only one DBD statement per access descriptor. There is one RECORD statement for each segment in the database. The SL field specifies the segment length as defined in the DBD statement. There is one item record for each field in the database. The item name is a 32-character name that describes the field. The LV field specifies the numeric level of the field and can indicate items that are part of repeating groups. The DBF field specifies how the field is stored in the database and reads the IMS data from disk. The SE field supplies the name of any search fields defined in the DBD and the IMS engine to construct Segment Search Arguments (SSAs). The KEY field indicates the items that are key or sequence fields in the database. The SN field specifies SAS names. If SAS names are not specified for a particular item, PROC ACCESS uses the first eight characters of the item name as the SAS name. Once the access descriptor is created, the view descriptor is created to select all items in the database. Notice that the CREATE statement for the view descriptor contains the name of the Program Specification Block (PSB) that the IMS engine uses to access the data.

A subset clause can be added to an IMS view descriptor. When the IMS engine uses a view descriptor containing a subset clause, it

examines the clause to see if it can be turned into an SSA to be passed to IMS. Performance is improved when the subset clause can be turned into an SSA because IMS returns only the records that meet the subsetting conditions. If the subset clause cannot be turned into an SSA, IMS returns all records in the database, and the SAS supervisor does the filtering.

That is an overview of the SAS/ACCESS products that will be available for Release 6.07 for MVS, CMS, and VMS. There is one additional access product that I have not mentioned that is currently available. It is the SAS/ACCESS interface to OS/2[®] Databases, which contains interface engines for dBASE[®] files, dIF[®] files, and the OS/2 Database Manager. It is available for Release 6.06 under OS/2.

The interfaces that we will be working on in the next year include ORACLE, INGRES, and SYBASE[®] on some of the UNIX[®] platforms. We will also be working on an engine that executes under OS/2 and accesses data on an AS/400[®] machine using the IBM AS/400 PC Support product. We will be providing engines for dBase and dIF files under Microsoft[®] Windows[®] in Release 6.07 under Windows 3.0. Other products that we are investigating are interfaces to Microsoft SQL/Server, CA-IDMS[®], and a VSAM and flat file engine similar to our other interface engines.

The SAS/ACCESS interfaces provide a tool for transparently accessing your databases and building SAS applications to analyze your DBMS data. The new features in Release 6.07 make your applications even more powerful and flexible. Let's take a look at several different applications that calculate the current value of wine inventory stored in a DB2 table called SASDEMO.INVENTORY. The table contains columns CASES, which contains the number of cases of each wine in our inventory and PRICE, which contains the price per case. The sum of the price times the number of cases for each wine represents the value of our inventory.

The following code shows an example of creating an access and a view descriptor using the statement-style syntax and then using the view to calculate the value of the goods in stock in a DB2 table named SASDEMO.INVENTORY.

```
PROC ACCESS DBMS=DB2;
CREATE WORK.INV.ACCESS;
TABLE=SASDEMO.INVENTORY;
CREATE WORK.INV.VIEW;
SELECT PRICE CASES;
RUN;
DATA _NULL_;
SET INV END = EOF;
RETAIN TOTAL 0;
TOTAL = TOTAL + (PRICE * CASES);
IF EOF THEN PUT 'Total Inventory = ' TOTAL DOLLAR15.2;
RUN;
```

The same query could have been done using the passthru feature of PROC SQL. In this case the calculations are done by DB2 instead of by the SAS system.

```
PROC SQL;
SELECT EXPRESN LABEL = 'Total Inventory' FORMAT = DOLLAR15.2
FROM CONNECTION TO DB2
(SELECT SUM (PRICE * CASES) FROM SASDEMO.INVENTORY);
QUIT;
```

The default SAS name assigned to a calculated field is EXPRESN. The LABEL= option assigns a SAS label to the variable to be used when PROC SQL displays the results of the query. The FORMAT= option assigns a format to the results. In many cases it is more efficient for the database to do the calculations because it can optimize the query based on the table structure and available indexes.

If you use a workstation and have licensed SAS/CONNECT[®] software for the workstation and the mainframe running DB2, you can perform the SQL query on a remote MVS TSO session. You add

SAS code to do the connection and remote submit of the query from the workstation, and the query executes on the mainframe. The following example shows the code that you submit on an OS/2 machine to start a remote session with an MVS TSO session and execute the SQL query.

```
FILENAME RLINK 'CONNECT\TSO.SCR';
OPTIONS COMAND=SHLLAP1 REMOTE=MVS;
SIGNON;
RSUBMIT MVS;
PROC SQL;
SELECT EXPRESN LABEL = 'Total Inventory' FORMAT = DOLLAR15.2
FROM CONNECTION TO DB2
(SELECT SUM (PRICE * CASES) FROM SASDEMO.INVENTORY);
QUIT;
ENDRSUBMIT;
FILENAME RLINK 'CONNECT\TSO.SCR';
SIGNOFF MVS;
```

The SIGNON command executes the script file TSO.SCR defined by the RLINK LIBNAME statement. It establishes a remote SAS session and connection between the local and remote SAS sessions using the information provided by the OPTIONS statement. Once the connection is established, the statements bracketed by the RSUBMIT and ENDRSUBMIT statements are sent to the MVS session for execution. The SIGNOFF command terminates the connection between the local and remote SAS sessions using the TSO.SCR script file.

If you have SAS/AF[®] or SAS/FSP[®] software and you want to build an application that calculates the current value of the inventory, you can write an SCL program that can be called from a full-screen application to display the current value of the inventory.

```
INIT;
TOTAL = 0;
SUBMIT CONTINUE;
PROC ACCESS DBMS=DB2;
CREATE WORK.INV.ACCESS;
TABLE=SASDEMO.INVENTORY;
CREATE WORK.INV.VIEW;
SELECT PRICE CASES;
RUN;
ENDSUBMIT;
INV = OPEN('WORK.INV');
DO WHILE (FETCH(INV) NE -1);
P = GETVARN(INV, 1);
C = GETVARN(INV, 2);
TOTAL = TOTAL + (P * C);
END;
RETURN;
MAIN;
RETURN;
TERM;
RETURN;
```

The submit block uses the statement-style syntax of PROC ACCESS to create temporary access and view descriptors for the SASDEMO.INVENTORY table. Once the descriptors are created, the program opens the view, reads the rows, and calculates the inventory value into the variable TOTAL. If TOTAL is defined as a window variable, its value is displayed on an application window.

As you can see, there are a lot of new features and functionality in our SAS/ACCESS products in Release 6.07, and there are many more enhancements coming your way in the future. We are committed to providing you transparent access to your database data so that the SAS System can be used for all of your applications.

SAS, SAS/ACCESS, SAS/AF, SAS/FSP, SAS/CONNECT, SAS/DB2, and SAS/SQL-DS are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. * indicates USA registration. IBM, AS/400, DB2, and SQL/DS are registered trademarks or trademarks of International Business Machines.

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