ABSTRACT

The transparent access of data from external databases is now a reality under Version 6 of the SAS System. Multiple engine concepts in the Multiple Vendor Architecture (MVA) bring increased flexibility and power to SAS users across many of the SAS platforms. Under the VMS operating system, two database engines have been developed for Release 6.06 of the SAS System. The SAS/ACCESS Interface to ORACLE and the SAS/ACCESS Interface to RdbIVMS enable you to combine the power of the SAS System with your data stored in these database management systems (DBMS).

INTRODUCTION

Release 5.18 of the SAS System under the VMS operating system introduces database interfaces in the form of SAS procedures. The ORACEXT procedure allows you to extract data from ORACLE tables and place it in SAS data sets for future use. Similarly, you can use the RDBEXT procedure to convert RdbIVMS tables into SAS data sets.

Database engines available with Version 6 SAS/ACCESS interfaces eliminate the conversion of the external database into a SAS data set. You can now use the DATA step or PROC steps to directly execute against the DBMS without the intermediate step of bringing the data into a SAS data set. Consequently, the data used by the SAS System will always be current since it is being accessed directly from the database.

Other new features of the SAS/ACCESS products include the ability to modify the database from within the SAS System and the ability to create an ORACLE table or RdbIVMS relation from a SAS data set. All of these enhancements to the SAS/ACCESS family of interfaces provide you with a reliable data management environment.

PROC ORACEXT UNDER VERSION 5

The ORACEXT procedure, available to VMS users under Release 5.18 of the SAS System, provides a means of extracting data from ORACLE tables and placing the data in temporary or permanent SAS data sets. Once a data set has been created from one or more ORACLE tables, the data set can be further processed by the DATA step or PROC steps.

Syntactically, PROC ORACEXT is very simple, yet restrictive. The PROC ORACEXT statement contains a few options to indicate the ORACLE user name and password, and the name of the SAS data set being created. The only other valid statement available with this procedure is the SELECT statement.

A limitation of 1000 characters is placed on the SELECT statement, which restricts the complexity or quantity of variables of the selection. After you submit the PROC ORACEXT code, the SELECT statement is parsed by the SAS System and transferred to ORACLE's Structured Query Language (SQL) for processing. If ORACLE SQL does not support a certain SQL construction or syntax in the SELECT statement, then the SAS SELECT statement does not accept it either.

Version 5 Syntax for the SAS/ACCESS Interface to ORACLE

Below is an example of what PROC ORACEXT may look like in a SAS program and the output from a PRINT procedure:

```
PROC ORACEXT OUT=supply NAME='pele' PWD='soccer';
SELECT brandname, item, material, size_of_item, color, price from inventory
WHERE material='cotton' OR material='leather'
ORDER BY item, brandname, color;
PROC PRINT DATA=supply LABEL NOOBS;
TITLE 'Inventory of cotton uniforms and accessories';
FORMAT price DOLLAR7.2;
RUN;
```

```
<table>
<thead>
<tr>
<th>BRANDNAME</th>
<th>ITEM</th>
<th>MATERIAL</th>
<th>SIZE_OF_ITEM</th>
<th>COLOR</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adidas</td>
<td>ball</td>
<td>leather</td>
<td></td>
<td>white</td>
<td>$20.00</td>
</tr>
<tr>
<td>Adidas</td>
<td>ball</td>
<td>leather</td>
<td></td>
<td>white</td>
<td>$29.00</td>
</tr>
<tr>
<td>Adidas</td>
<td>shirt</td>
<td>cotton</td>
<td></td>
<td>green</td>
<td>$15.99</td>
</tr>
<tr>
<td>Umbro</td>
<td>shirt</td>
<td>cotton</td>
<td></td>
<td>red</td>
<td>$15.60</td>
</tr>
<tr>
<td>Adidas</td>
<td>shoes</td>
<td>leather</td>
<td></td>
<td>black</td>
<td>$29.99</td>
</tr>
<tr>
<td>Adidas</td>
<td>shoes</td>
<td>leather</td>
<td></td>
<td>black</td>
<td>$30.99</td>
</tr>
<tr>
<td>Nike</td>
<td>shoes</td>
<td>leather</td>
<td></td>
<td>black</td>
<td>$76.99</td>
</tr>
<tr>
<td>Patrick</td>
<td>shoes</td>
<td>leather</td>
<td></td>
<td>black</td>
<td>$83.99</td>
</tr>
<tr>
<td>Lotto</td>
<td>shorts</td>
<td>cotton</td>
<td></td>
<td>black</td>
<td>$71.99</td>
</tr>
<tr>
<td>Lotto</td>
<td>shorts</td>
<td>cotton</td>
<td></td>
<td>red</td>
<td>$74.99</td>
</tr>
</tbody>
</table>
```

Output 1 ORACLE Data Stored in the SAS Data Set SUPPLY

For large ORACLE tables, you should run the SAS job in batch mode for more productive use of resources.

SAS/ACCESS INTERFACE TO ORACLE UNDER VERSION 6

Version 6 of the SAS System brings many enhancements to the SAS/ACCESS interface products. The interface comprises three main parts: the ACCESS procedure, the interface view engine, and the DBLOAD procedure. Of the three, the interface view engine is the most transparent and does not require you to learn any new syntax. The two procedures do require some study, but each provides interactive windows used in guiding you through the procedure.

ACCESS procedure

PROC ACCESS is the front end to all of the SAS/ACCESS products under Release 6.06. This procedure enables you to describe the database data to the SAS System. The first step involves the creation of an access descriptor that is used as a definition of the database table. You must do this interactively using the screens provided. First, you must identify the library in which the descriptor
files will be placed. The running example in this paper uses the following LIBNAME statement:

LIBNAME desclib 'disk:sas.descriptors';

Invoke the ACCESS procedure using one of two methods:

PROC ACCESS;
RUN;
or

PROC ACCESS ACCDESC=desclib.inv FUNCTION=C;
RUN;

The first method brings up the ACCESS window where you create an access descriptor by using the CREATE statement. The second statement enables you to bypass the ACCESS window and brings you directly to the Access Descriptor Identification window. If more than one SAS/ACCESS product is available, the primary goal is to provide a transparent method of accessing data stored in another vendor’s DBMS or data stored in SAS data files. Interface view engines retrieve data directly from files formatted by the DBMS products through the use of SAS/ACCESS views or view descriptors. The view descriptor does not actually contain data values. Instead, it contains a definition or a description of data stored in the DBMS table.

Interface View Engine

The second part of the SAS/ACCESS Interface product is the interface view engine, which allows you to use the ORACLE data in SAS programs in much the same way SAS data files are used. Engines are a new feature of Version 6 of the SAS System. The primary goal of the Multiple Engine Architecture is to provide a transparent method of accessing data stored in another vendor’s DBMS or data stored in SAS data files. Interface view engines retrieve data directly from files formatted by the DBMS products through the use of SAS/ACCESS views or view descriptors. The view descriptor does not actually contain data values. Instead, it contains a definition or a description of data stored in the DBMS table.

Interface view engines enable you to use SAS procedures and program statements to process data values stored in these files without the cost of converting them into SAS data files. This saves storage space and maintenance costs because changes made to the source data are automatically reflected by the view results. Be aware that alterations to the table (dropping or adding columns) will not be reflected by the view and may cause the view descriptor to be out-of-date or invalid.

At this point, a view descriptor can be created based on the access descriptor similar to the access descriptor, the view descriptor is created with the interactive use of PROC ACCESS. The first method involved in creating a view descriptor follows:

PROC ACCESS;
RUN;

This brings up the ACCESS window. Place the cursor next to the access descriptor on which the view descriptor will be based. Enter CV and press the ENTER key. The ORACLE View Descriptor Display window appears next. To bypass the ACCESS window, specify the following SAS statements:

PROC ACCESS ACCDESC=desclib.inv;
RUN;

If more than one SAS/ACCESS product is available, the Engine Selection window appears before the ORACLE View Descriptor Display window. From the View Descriptor window, you need to select several features: the name of the view, the library where the view will be stored, an optional SAS data set name to be used in extracting ORACLE data to create a SAS data set, the columns to be included in the view, their corresponding SAS variable names and formats, and optional selection criteria determined by a WHERE clause. The WHERE clause can be specified for your view descriptor by using the SUBSET command to enter the Selection Criteria Entry window.

If changes need to be made to the view descriptor, place the ED command next to the view descriptor in the ACCESS window to bring up the View Descriptor window for modification. Table integrity can be achieved by keeping the access descriptor and the view descriptor up to date with the ORACLE table. Unlike access descriptors, there may be several view descriptors per ORACLE table. The subsetting of the ORACLE table, depending on the application, lends itself to the creation of multiple views.

The next example shows the use of a view descriptor in a PROC PRINT statement. Output 2 shows the output that is created for the inventory of soccer shoes in the running example.

PROC PRINT data=desclib.shoes NOOBS LABEL;
VAR brand size price quantity;
FORMAT price DOLLAR7.2;
TITLE 'Soccer Shoes Inventory';
RUN;

Output 2 Data Described by the View DESCLlB.SHOES

<table>
<thead>
<tr>
<th>BRAND</th>
<th>SIZE</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adidas</td>
<td>7</td>
<td>29.99</td>
</tr>
<tr>
<td>Adidas</td>
<td>6</td>
<td>29.99</td>
</tr>
<tr>
<td>Adidas</td>
<td>5</td>
<td>29.99</td>
</tr>
<tr>
<td>Nike</td>
<td>10</td>
<td>79.99</td>
</tr>
<tr>
<td>Nike</td>
<td>9</td>
<td>85.99</td>
</tr>
</tbody>
</table>

DBLOAD Procedure

The final part of the SAS/ACCESS interface products is the DBLOAD procedure. This procedure enables you to create and load ORACLE tables using data from SAS data sets. PROC DBLOAD can also be used for sending SQL statements to ORACLE for processing without leaving the SAS session. The DBLOAD procedure constructs SQL statements to create and load an ORACLE table using data from a SAS data file, from a view created with the SQL procedure, or from ORACLE or another DBMS. This procedure can be run interactively or in batch mode. When run interactively, windows guide you through the process, prompting you for the necessary information and providing HELP screens when needed. Batch processing provides a computing alternative that eliminates the burden of online resources. The interactive method begins with submitting the following SAS statements:

PROC DBLOAD;
RUN;

When the ORACLE Load Identification window appears, identify the location of the input data and specify the name of the ORACLE table to be created. Other required fields to be completed in this window include the User Name and Password fields. Optional fields enable you to include the access descriptor name to be created from the
The next window that appears is the ORACLE Load Display window, which is used in specifying the ORACLE column names and data types associated with each SAS variable in the input data. After you have made these choices, a prompt appears explaining how many possible inserts could be made into the table. If the LOAD command is issued at this time, the ORACLE table is loaded with the data from the SAS data file. Large numbers of inserts should be done in batch mode for more efficient use of resources. Below is an example of what PROC DBlOAD may look like in a batch job:

PROC DBlOAD DBMS=ORACLE DATA=mydata.orders;
SSID=soccer;
PASSWORD=soccer;
TABLE=oraclesoccer;
ACCESS=descrn.lib.;
WHERE material='cotton' OR material='leather';
ORDER BY item, brandname, color;
RUN;

To submit ORACLE SQL statements to ORACLE while running the DBlOAD procedure, invoke the Query Entry window from the command line. Any valid ORACLE SQL statement can be entered at this time with the exception of the SELECT statement or any statement that refers to the table created within the current running DBlOAD procedure. This can be very useful in creating views, browsing or modifying other tables, or executing stored ORACLE SQL statements without having to leaving the SAS System. Note that these statements are not parsed by the SAS System and are sent directly to ORACLE.

One important point needs to be stressed before ending this portion on the Version 6 SAS/ACCESS Interface to ORACLE. A shadow process is created as a route of communication between the SAS System and ORACLE each time you use the interface. At times this shadow process (a subprocess) may remain if the SAS System is abnormally terminated or if a problem occurs while using the interface. This subprocess needs to be terminated before the next execution of the interface from your process ID.

PROC RDBEXT UNDER VERSION 5

The RDBEXT procedure provides a means of extracting data from Rdb/VMS tables and placing the data in temporary or permanent SAS data sets. Release 5.18 of the SAS/ACCESS interface to Rdb/VMS under the VMS operating system consists of PROC RDBEXT and can be used to access and extract Rdb/VMS data interactively, noninteractively, or in batch mode. Once a data set has been created from one or more Rdb/VMS tables, the data set can be further processed by subsequent PROC or DATA steps.

PROC RDBEXT resembles PROC ORACTEX in its simplicity and syntax. The PROC RDBEXT statement options include the name of the Rdb database, whether date values or datetime values are generated from Rdb variables, the name of the SAS data set being created, and the SQL option indicating what type of SELECT statements can be used. Just as in the ORACTEX procedure, the only other valid statement is the SELECT statement.

Depending on the software installed at your site, there are two different access methods supported by PROC RDBEXT for extracting data from an Rdb database. The first and most restrictive is through the Digital Standard Relational Interface (DSRI). This type of access restricts the SELECT statement to selecting any number of variables from one table or view. The WHERE clause may be used with the SELECT statement to specify one condition used in selecting the data. The VAX SQL method allows you to use any valid SQL SELECT statement and is restricted to 4000 characters in length.

Version 5 Syntax for the SAS/ACCESS Interface to Rdb/VMS

Below is an example of what PROC RDBEXT may look like in a SAS program. Output 3 shows the output from a PROC PRINT.

PROC RDBEXT OUT=supply db=soccer supply.rdb SQL;
SELECT brandname, item, material, size_of_item, color, price from inventory
WHERE material='cotton'
ORDER BY item, brandname, color;
PROC PRINT DATA=supply LABEL;
TITLE 'Inventory of cotton uniforms and accessories';
FORMAT price DOLLAR2.2;
RUN;

<table>
<thead>
<tr>
<th>Brandname</th>
<th>Item</th>
<th>Material</th>
<th>Size_of_Item</th>
<th>Color</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASt ball</td>
<td>leather</td>
<td>white</td>
<td>125.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spaldilltj ball</td>
<td>leather</td>
<td>white</td>
<td>21.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IUk&lt;! $birt cotton</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.ke shoes</td>
<td>leather</td>
<td>black</td>
<td>139.99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patrick shoes</td>
<td>leather</td>
<td>black</td>
<td>139.99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nite shoes</td>
<td>leather</td>
<td>black</td>
<td>139.99</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Output 3. Rdb/VMS Data Stored in the Data Set SUPPLY

For large Rdb/VMS tables, you should execute your SAS job in batch mode for more productive use of resources.

SAS/ACCESS INTERFACE TO RDB/VMS UNDER VERSION 6

The SAS/ACCESS interface to Rdb/VMS consists of the same three parts common to all SAS/ACCESS interface products. The first part, the ACCESS procedure, is used to define the SAS/ACCESS descriptor files. The interface view engine is the next component, which enables you to use Rdb/VMS data in SAS programs in much the same way you use SAS data files. Lastly, the DBlOAD procedure provides you with the ability to create and load the Rdb/VMS tables using data from SAS data sets. PROC DBlOAD can also be used for sending VAX SQL statements to Rdb/VMS for processing.

ACCESS Procedure

The invocation of PROC ACCESS is identical to what is explained in SAS/ACCESS INTERFACE TO ORACLE UNDER VERSION 6 earlier in this paper. The ACCESS procedure is used first in creating an access descriptor for the Rdb/VMS table in the user-specified database. This access descriptor contains information on the Rdb/VMS database and table name, the Rdb/VMS column names, and their default SAS formats. From the access descriptor, one or more view descriptors are created and are used to define a subset of the data. Remember that view descriptors do not contain the data, only a description of the Rdb/VMS table. Both the access descriptors and view descriptors are listed in the ACCESS window and can be browsed, modified, or deleted from this listing. If an application tends to use the same Rdb/VMS data multiple times without modification to the data, it is recommended that the data be extracted from the table in one of two ways. The first method uses the EXTRACT command from the ACCESS window.
Consider the following SAS statement:

```
PROC ACCESS VIEWDESC=desclib.uniforms OUT=mydata.uniform;
RUN;
```

Note that a view descriptor and a SAS data file must have unique names if they are stored in the same SAS data library (same VMS directory).

**Interface View Engine**

The second component of the SAS/ACCESS interface to Rdb/VMS is the interface view engine. As mentioned earlier, the interface view engine allows you to use the Rdb/VMS data in SAS programs in much the same way you use SAS data files. For example, if you want to modify the data contained in an Rdb table, use the FSEDIT procedure as shown below:

```
PROC FSEDIT DATA=desclib.uniforms;
RUN;
```

Another useful procedure to be used with SAS/ACCESS views is PROC SQL, a new procedure introduced with Version 6 of the SAS System. Since view descriptors can only act on one Rdb/VMS table, the SQL procedure provides another way to select and combine data from multiple SAS data files or SAS/ACCESS views or both. The next example is a combination of the uniform view descriptor and a team view descriptor to produce a PROC SQL view; the corresponding output is shown in Output 4.

```
LIBNAME sqllib 'disk:lsas.sqlviewsj';
PROC SQL;
CREATE VIEW sqllib.unl_team AS SELECT uniforms.brandname, uniforms.color, uniforms.material, team.name, team.manager FROM desclib.team, desclib.uniforms WHERE team.color=uniforms.color AND team.material=uniforms.material ORDER BY team.name;
TITLE 'Team Uniforms';
SELECT * FROM sqllib.unl_team;
```

![Output 4](image)

**DBLOAD Procedure**

Like the other SAS/ACCESS interface products, PROC DBLOAD enables you to create Rdb tables from SAS data files. This can be done using the interactive windows or by using SAS program statements. Required information for using this procedure includes the database and table name and the name of the source SAS data file. Other optional information includes the name of an access descriptor to be created from this information, a commit frequency, an input limit, and an error limit. Once again, it is stressed that large transfers of SAS data to Rdb/VMS be done in batch.

The Query Entry window, a part of PROC DBLOAD, is also available with the interface to the Rdb/VMS software. From this window, you can submit VAX SQL statements to Rdb/VMS while running PROC DBLOAD interactively or in batch mode. In batch mode, simply precede all VAX SQL statements with SQL and place the statement after the PROC DBLOAD statement. Remember the VAX SQL statements cannot access any new tables created by the running DBLOAD procedure.

**CONCLUSION**

We have seen the development of the SAS/ACCESS interface software from the simplistic and restrictive stage of SAS procedures, PROC RDTEXT and PROC ORACEXT, to the engine architecture system of Version 6. Through the use of the ACCESS procedure, database tables can be described to the SAS System with little knowledge needed about the SAS System or the DBMS. The interface view engine allows you to use your database data as if they existed in a SAS data set. PROC DBLOAD carries the interface software one step further in providing the capability to create new database tables from SAS data files or manipulate the database using VAX SQL statements from within the SAS System. New SAS/ACCESS interface products will follow the same structure as seen in the ORACLE and Rdb/VMS interfaces. Once you have mastered one SAS/ACCESS interface, you have the basic knowledge needed to understand and use other SAS/ACCESS interfaces.

**ACKNOWLEDGEMENTS**

I would like to thank Virginia Dineley and Todd Lloyd for their technical assistance and encouragement.

**REFERENCES**


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

ORACLE is a registered trademark of Oracle Corporation.

Rdb/VMS is a trademark of Digital Equipment Corporation.

SQL is a trademark of International Business Machines Corporation.

Adidas is a registered trademark of Adidas Sportschuhtabriken Adi Dassler KG Limited Partnership.