Paper 10-26

Database Access Using the SAS System

Frederick Pratter, Computer Science Department, University of Montana Missoula

INTRODUCTION: CLIENT/SERVER DATABASE MANAGEMENT SYSTEMS

The SAS® System currently provides many of the features of a database management system, including database views and an extended superset of ANSI SQL. However, it is often impractical or just plain impossible to convert desktop or legacy databases into SAS. Consequently, the SAS System® provides several procedures for access to relational databases. This paper will review how to use the various SAS/Access® products to link networked workstations to remote servers. Most of the examples will use the SAS/Access® Interface to Oracle, but the principles described apply equally to local databases in Microsoft Access 2000, as well as other client/server systems such as DB2.

In order to understand the various SAS/Access® options, it is important to recognize that this product was written specifically to run on client/server database systems, in which a separate database engine supplies data to the local application. In this paper, all of the examples use SAS as the client and a relational DBMS as the server. Since the SAS System® was originally written to run only on the mainframe platform, it did not initially provide a separate server engine. [1]

In contrast, a relational DBMS (Database management System) such as Oracle runs as a network-centric application that supports efficient database access. This efficiency results because it is not necessary to copy the entire database each time a set of records is selected. The server engine selects the desired records and only these are sent over the network to the client. The database server is generally not on same platform as the application although in principle it is possible to have the server run locally on the same machine.

ACCESS TO RELATIONAL DATABASE MANAGEMENT SYSTEMS (RDBMS)

The complexity of the SAS/Access® product results from the necessity of sending database commands from the client to be executed by the server. The product is available in three different forms, depending on the platform. As the following table illustrates, there are more options available for Windows than for systems running Unix operating systems.

[1] The SAS/Share® product was added to allow this functionality for reading SAS data libraries.

Table 1. SAS/Access Interfaces by Platform

Client	Server	Available Interfaces
Unix	Unix	SAS/Access Interface to Relational Databases
Windows	Unix	SAS/Access Interface to Relational Databases SAS/Access Interface to ODBC SAS/Access Interface to OLE DB
Windows	Windows	SAS/Access Interface to Relational Databases SAS/Access Interface to ODBC SAS/Access Interface to OLE DB SAS/Access to PC File Formats

It is important to note that there are DBMS specific as well as operating system differences from system to system. For example, to use SAS/Access for Oracle on a Unix system, it is necessary to set the environment variable SASORA to V7, V8, or V8i, depending on the ORACLE version. When using this product on a Windows platform, it is necessary to install and properly configure Oracle SQLPlus on the client workstation in order to access remote databases via TCP/IP. The best advice is to consult the documentation that comes with the specific version of the SAS/Access® product.

The only option available for a user at a Unix client workstation is to use one of the relational database specific products. Currently a number of these are available, the most widely used of which are probably the SAS/Access Interface to Oracle and the SAS/Access Interface to DB2.[2]

Each of these interfaces includes three procedures:

- PROC ACCESS – Used to download DBMS data into SAS
- PROC DBLOAD – Load SAS data into DBMS
- PROC SQL Pass-Through – Execute native SQL (Structured Query Language)

[2] Other supported products include SQL Server, Sybase, CA-OpenIngres, Informix., and Adabas

The Access procedure (as distinct from the SAS/Access® product) is extremely cumbersome to use. I once described it as "probably the worst designed SAS product of the decade" (Pratter, NESUG '94). This was because in Version 6 of the SAS System® it was necessary first to create an Access descriptor to describe the data in a single DBMS table and then create a second View descriptor to define a subset of the DBMS data described by the Access descriptor.

PROC DBLOAD works in the opposite direction from PROC ACCESS. It is used to copy data into a DBMS from a SAS dataset. This PROC is useful for bulk loads, e.g., copying entire SAS datasets into Oracle, but *caveat programmer*: there are two important "features" that must be noted:

- The default load limit is 5000 records; in order to load larger tables, specify *limit=0*
- The PROC will abend if table exists; it can only be used to create new tables.

One of the important changes in Version 7 and higher, is that it is no longer necessary to go through this cumbersome process-- one no longer needs to create access and view descriptors. As the following code illustrates, by using a "dynamic libname engine" SAS can treat the remote database as if it were a SAS dataset.

```
libname oralib oracle
    user=scott password=tiger path=sample;
```

Using the interactive feature in the latest versions of the SAS System, this process is even simpler. Selecting **File, New** from the libname window brings up the window shown in Figure 1. The user is prompted to enter a libname, which can be any valid SAS library name. In the example, the engine selected is Oracle and the user and path information is from the batch example above.

Note the "Enable at startup" check box. If this is selected, the library name will be available in this user's profile. The next time SAS is started, the Oracle data will be available automatically, along with the standard SAS data libraries.

The Access and DBLoad procedures are still available. Programs written using Version 6.06 and later should still work correctly. Using the new libname engines feature, however, allows the SAS user to create relational database tables in the DATA step, greatly simplifying the process. It is even possible to update and delete Oracle tables directly from SAS, using PROC DATASETS or the LIBNAME window, just as if they were native SAS datasets.

SQL PASS-THROUGH

The least complicated way to manage remote database tables in Version 6 was with PROC SQL. This alternative still offers a powerful and relatively straightforward interface for experienced SQL users. Except for bulk loads, where it is still probably better to use DBLoad, PROC SQL has the advantage of simplicity and familiarity.

The following example illustrates how to use the PROC SQL connect to as an alternative to PROC ACCESS:

```
proc sql;
connect to oracle
    (user=scott orapw=tiger path=sample);
create table EMP as
    select * from connection to oracle
            (select * from emp);
disconnect from oracle;
quit;
```

Three SQL statements are necessary. "Connect" and "disconnect" attach to the database and detach respectively. The "select" statement has two parts: the parenthesized expression `select * from EMP` is the pass-through SQL. This code is sent to the Oracle database server to return the data from table "emp". The outer `select * from connection to oracle` returns the result to SAS. Finally, the "create table" clause cause the results to be saved as the temporary dataset work.EMP. If this clause were omitted, PROC SQL would simply display the table in the output window (the default behavior for a SAS select).

The advantages of using SQL pass through can be substantial. For example, in the above illustration the parenthesized expression `select * from emp` could include a *where* clause, limiting the records to be selected. Alternatively, the outer select statement from the oracle connection could have a SAS SQL *where* clause, which would have the same effect of limiting the records displayed. The difference, of course, is that in the first case only the selected records would be sent to the client, while using a SAS *where* clause would result in the entire employee table being sent to the client, at which point the client would discard the unneeded records.

UNIX SERVER/WINDOWS CLIENT

It is still possible to use the SAS/Access® Interface to DBMS if the database is on a Unix server and SAS is running on a Windows client. In general, however, it is usually easier to use SAS/Access® to ODBC. The main advantage of the latter, as opposed to the DBMS specific products, is that it allows a Windows client to access a wide variety of database engines, not just the one it is licensed for.

The main drawback is that before using this product it is necessary to set up an ODBC data source that points to the database. The user needs to install the correct driver, for example, Microsoft ODBC for Oracle, and then go to the Windows Control Panel and click on "ODBC Data Sources (32 bit)" to set up an ODBC data source name, e.g. "Oracle". Figure 2 illustrates how this is done in Windows 98; Windows 2000 is virtually identical

This process only needs to be done once for each client workstation, after which the database will be accessible using the following SQL code:

```
proc sql;
connect to odbc
          (dsn=oracle uid=scott pwd=tiger);
create table EMP as
   select * from connection to odbc
          (select * from emp);
disconnect from odbc;
quit;
```

As the comparison to the previous example clearly illustrates, SQL pass-through works the same way in the DBMS specific products and in SAS/Access® to ODBC. The only difference is that the "connect" statement references the "DSN" (data source name) created using the ODBC administrator.

It is also important to note that in Version 7 and higher, one can use the dynamic libname engine to access Oracle tables and views via ODBC as if they were SAS datasets:

```
libname save odbc
          dsn=oracle uid=scott pwd=tiger;
```

The procedure for doing this is identical to the Oracle example shown in Figure 1, except that the options for ODBC are slightly different (see Figure 3).

There is one more database access option available in SAS Version 8 for client workstations using Windows. The SAS/Access product for OLE DB can support both native Oracle and ODBC access. The library window looks like Figure 3. It is only necessary to supply a value in the name field and select the OLE DB engine. Clicking on OK brings up the Data Link window shown in Figure 4 below.

Note that, depending on what software is available on the workstation, there are providers available for a variety of data sources including Oracle and ODBC. It is Microsoft's plan that the OLE DB engine should replace these native access methods, and SAS offers transparent connectivity to databases via this source.

ACCESS TO PC FILE FORMATS

In contrast to the other two products described, SAS/Access® to PC File Formats is not used to access client/server databases. Instead, it can be used on the Windows platform to read and write database or spreadsheet files. Licensing this product in SAS 6.12 includes PROC DBF and PROC DIFF, which were available in Version 6.04 as part of Base SAS but were subsequently unbundled. Alternately, one can set up Access and View descriptors to read Dbase and FoxPro tables (but not Visual FoxPro) and Lotus and Excel (but not Excel 97 or Excel 2000). In addition, the File menu Import/Export commands will run AF wizards that can be used to read and write PC files.

In PC File Formats for SAS version 7 and higher there is new support for MS Access and Excel 97. Also, PROC IMPORT and PROC EXPORT are available as standalone procedures, not just as menu items.

CONCLUSION

The SAS System® offers a variety of choices for RDBMS access, depending on client/server platform. The dynamic libname engines for Oracle, ODBC, and DB2 available in Version 8 are a great improvement over the clumsy access and view descriptors of the earlier releases. It is to be hoped that in future versions of SAS the situation will continue to improve and SAS will continue to expand libname support for external data sources of all sorts.

REFERENCES

SAS Online Documentation for Version 8. SAS/ACCESS® Software for Relational Databases: Reference. SAS Institute, Cary NC.

Frederick Pratter, "Desktop Database Management Using the SAS System®". Proceedings of the Sixth Annual Regional Conference, Northeast SAS Users Group, November 1993.

Andrew T. Kuligowski, "Advanced Methods to Introduce External Data into the SAS System®". Proceedings of the Twenty-Fourth Annual SAS Users Group International Conference. SAS Institute. Cary, NC.

TS329: SAS/ACCESS to ODBC Setup and Use with SAS. SAS Institute. Cary, NC.

TS-501: Data Acquisition and Exportation in PC SAS. SAS Institute. Cary, NC.

TS-518D: SAS/ACCESS® Guidelines for Connecting to ORACLE® Databases in the UNIX® Environment. SAS Institute. Cary, NC.

Jeff Polzin and Cheryl Garner, "Cross Platform Access to SAS DATA Files". SAS Institute. Cary, NC.

TS-609: Minimum Requirements for using SAS/ACCESS® Software for Relational Databases in the UNIX® Environment. SAS Institute. Cary, NC.

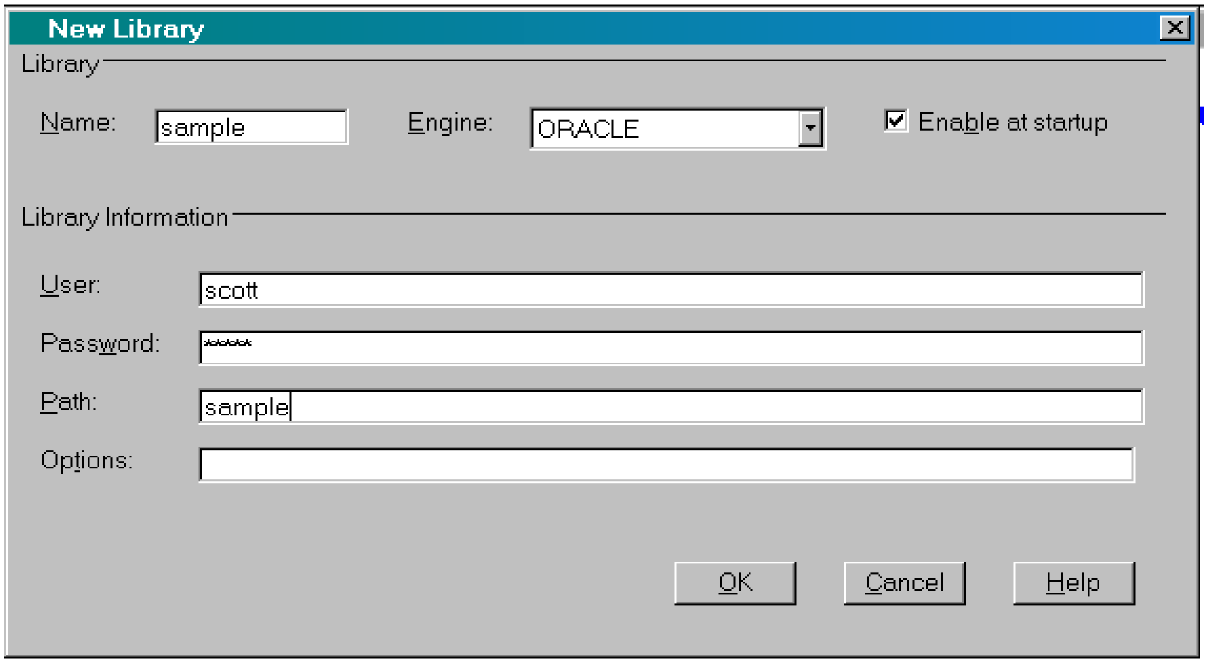
TS-624: Connecting to Oracle from SAS on WINNT or WIN95. SAS Institute. Cary, NC.

Roger E. Sanders, "Accessing Data from Your PC Using Version 7 of the SAS System". SAS Institute. Cary, NC.

CONTACT INFORMATION

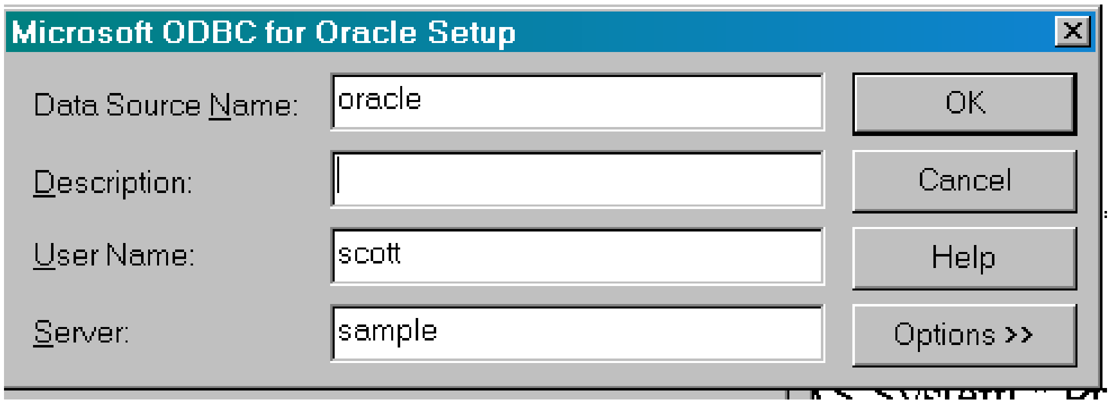
Frederick Pratter
Computer Science Department
University of Montana
Missoula, MT 59812
pratter@cs.umt.edu

Figure 1. New Library Window: Oracle Engine



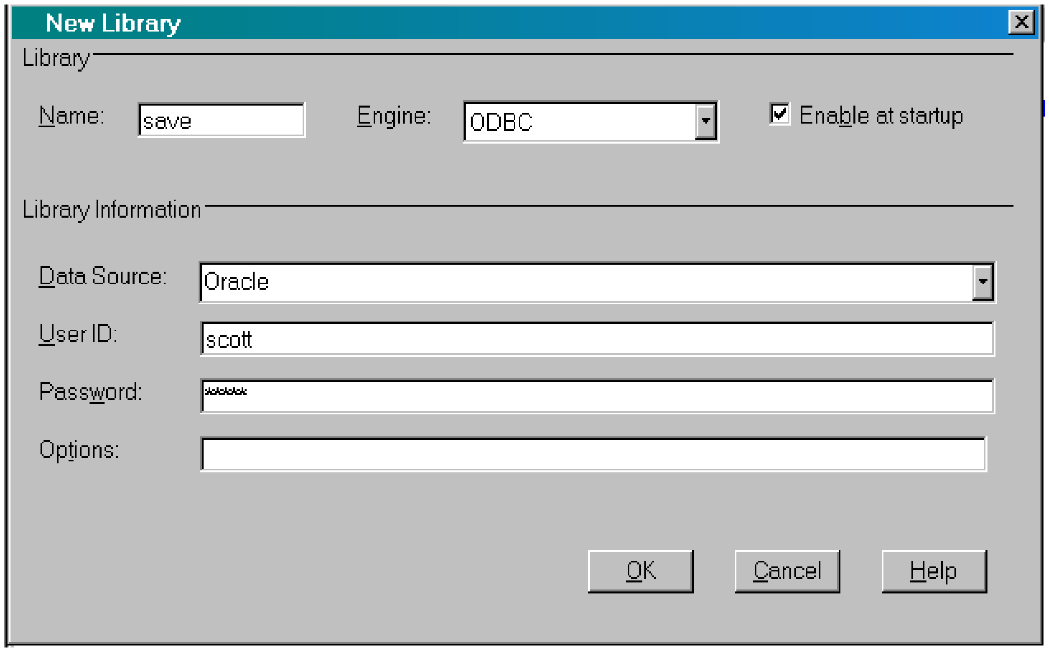
Name:	sample
Engine:	ORACLE
Enable at startup	✓
User:	scott
Password:	*****
Path:	sample
Options:	

Figure 2. Define ODBC Data Source



Data Source Name:	oracle
Description:	
User Name:	scott
Server:	sample

Figure 3. New Library Window: ODBC Engine



New Library

Library

Name: save

Engine: ODBC

☑ Enable at startup

Library Information

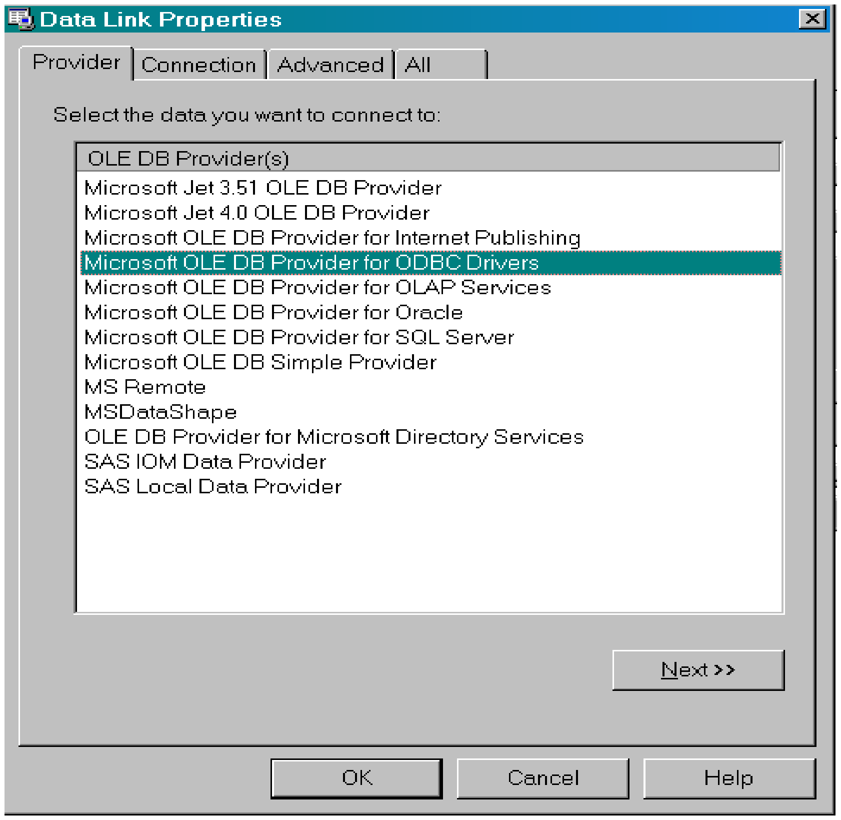
Data Source: Oracle

User ID: scott

Password: *****

Options:

Figure 4. OLE DB Data Link Properties



Data Link Properties

Select the data you want to connect to:

OLE DB Provider(s)

Microsoft Jet 3.51 OLE DB Provider
Microsoft Jet 4.0 OLE DB Provider
Microsoft OLE DB Provider for Internet Publishing
Microsoft OLE DB Provider for ODBC Drivers
Microsoft OLE DB Provider for OLAP Services
Microsoft OLE DB Provider for Oracle
Microsoft OLE DB Provider for SQL Server
Microsoft OLE DB Simple Provider
MS Remote
MSDataShape
OLE DB Provider for Microsoft Directory Services
SAS IOM Data Provider
SAS Local Data Provider