Paper TF-21

Access to Relational Databases Using SAS®

Frederick Pratter, Destiny Corp.

ABSTRACT

SAS® software 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 organizational databases into SAS. Consequently, SAS software provides several procedures for access to relational databases. This paper will review how to use the various SAS/ACCESS® products for linking networked workstations to remote servers. Most of the examples will use the SAS/ACCESS for Oracle, but the principles described apply equally to local databases in Microsoft Access, as well as other client/server systems such as DB2 and MySQL.

INTRODUCTION: CLIENT/SERVER DATABASE MANAGEMENT SYSTEMS

In order to understand the various SAS/ACCESS options, it is important to recognize that it 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 software as the client and a relational DBMS as the server. A relational DBMS (Database Management System) such as Oracle runs as a network application that provides 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 as SAS.

Before you can use SAS/ACCESS database software, in addition to Base SAS software and the SAS/ACCESS Interface software it is necessary to install a version of the client software. For Oracle, for example, the following is indicated:

- The minimum required Oracle client libraries release is Oracle, Release 10g.
- If you are using the 32-bit version SAS, you must have the 32-bit version of the Oracle client libraries. If you are using the 64-bit version SAS, you must have the 64-bit version of the Oracle client libraries.

For the examples in this paper, Oracle 11.2 Server was installed on a network host running the Linux operating system. The host name of this server is `hrothgar`, and the name of the configured database instance is `sample`. (Since this system is running in a test network it is not necessary to supply the fully qualified domain name, nor has a schema name been defined.) The Oracle Client is installed on a Windows system running Microsoft Windows 7 Professional; SAS 9.3 Integration Technologies software was also installed on this Windows host. This is a fairly typical setup, where SAS software is running on the client and connects remotely to the database server. The following diagram illustrates this simple configuration; your site probably will be more complex but similar in concept..

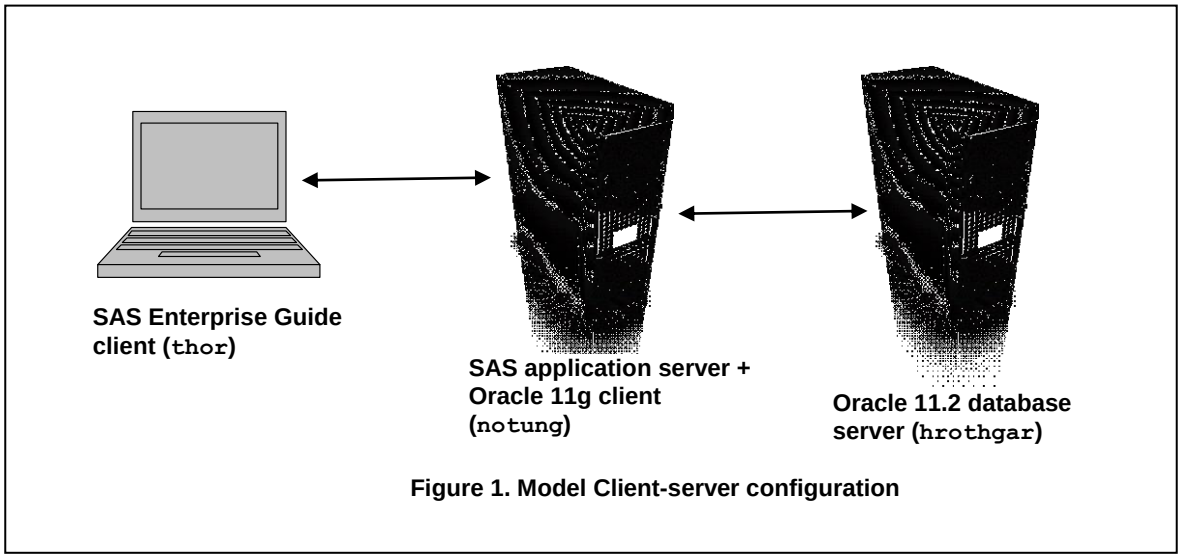

Figure 1. Model Client-server configuration

The most important thing, from the end-user standpoint, is that the Oracle Client must be set up and correctly configured on the local machine running SAS. You will need to contact your database administrator to determine the requirements for your particular environment. Oracle supplies a set of *Configuration and Migration Tools* to make this a little easier. The Oracle Net client software is available at no cost for educational and research purposes, but your site almost certainly has restrictions on what you are allowed to install. It may be that your site does things differently from the examples in this paper. Your IT support person should absolutely be consulted before attempting any of these activities, and certainly before attempting to access organizational resources.

The Oracle *Net Manager* is used to provide the Service Name and port on which the client will connect to the server. Unless this is done, SAS/ACCESS will not be able to find your server. In this case, the Service Name is the same as the Oracle SID. Ask your database administrator for the correct values at your site.

Here is an example of what the Oracle Net Manager configuration looks like on Windows 7:

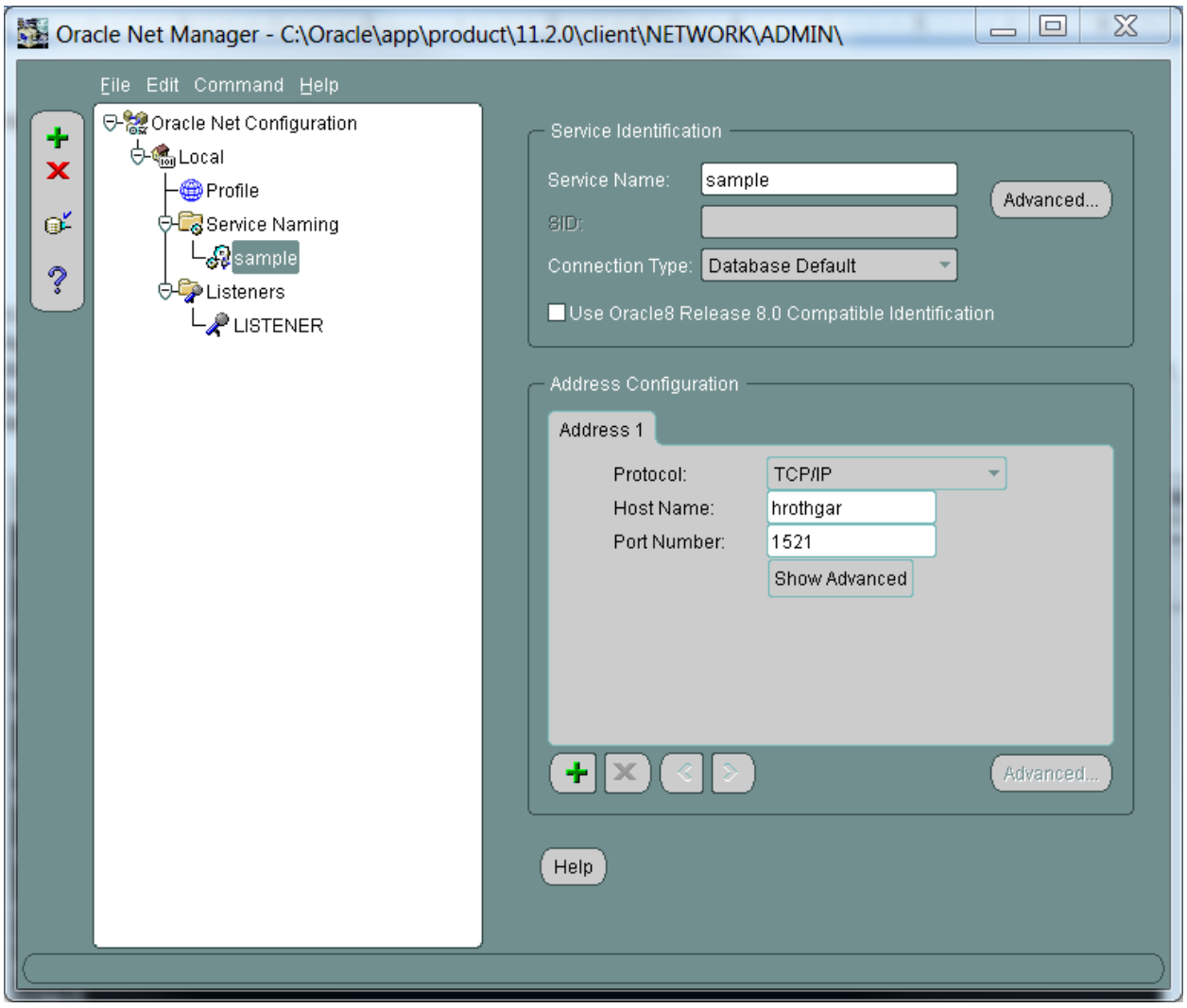

Figure 2. Oracle Net Manager

You can test that the Oracle client is correctly installed on your system by opening Oracle *SQL Plus* from the Application Development section of the Windows **Start>Programs>Oracle** menu. The standard Oracle sample database comes with a few pre-loaded tables, owned by user “scott”. These are used for the examples in this paper, but of course you will have your own user name, database SID, and table names. If you are not sure, again, please talk to your database administrator to get the correct values you need for your site.

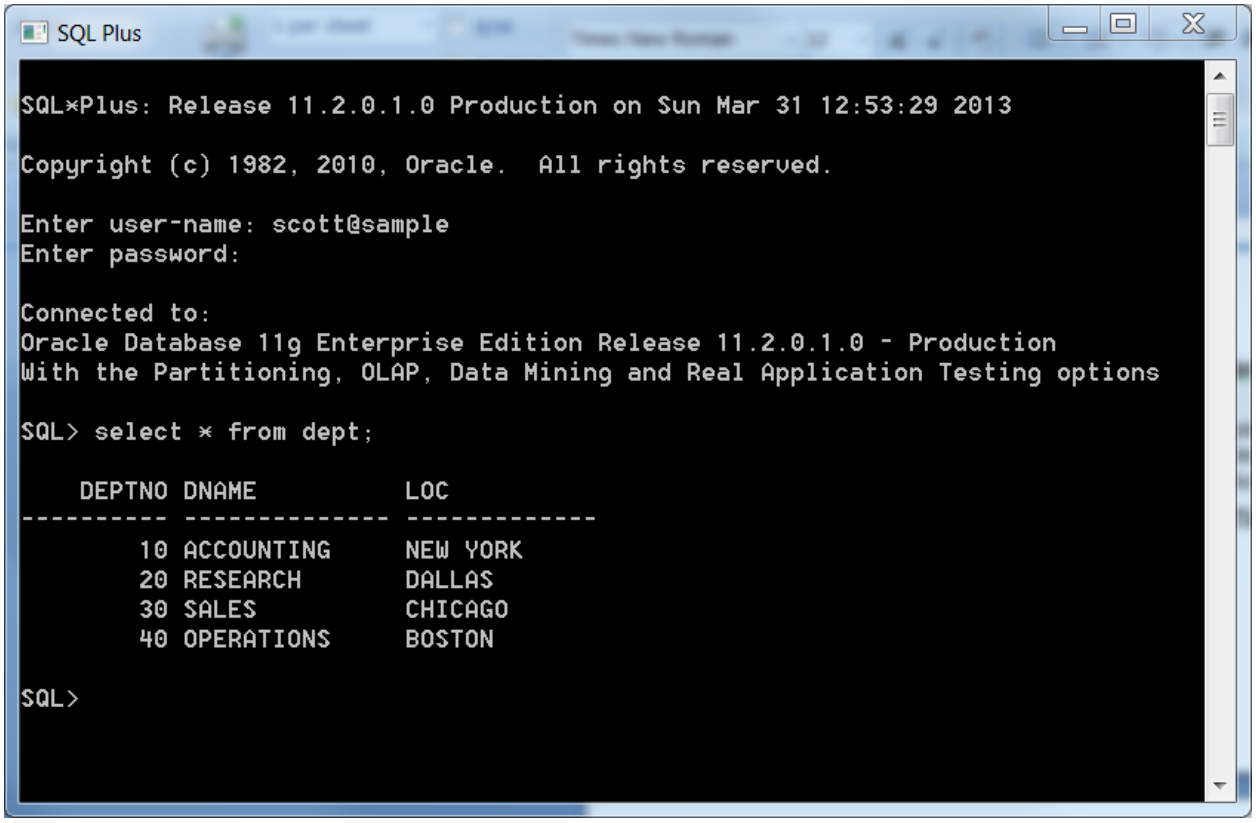

Figure 3. Oracle SQL Plus

This sample illustrates that the user “scott” is connected to the remote service “sample”. If you see something like this, your Oracle client is set up properly and you can connect to the server. Now you are ready to try SAS/ACCESS.

OVERVIEW: ACCESS TO RELATION DATABASE MANAGEMENT SYSTEMS (RDBMS)

There are SAS Access Interface products available for a large variety of database management systems; here is the current list:

- DB2
- Informix
- Microsoft SQL
- MySQL
- ODBC
- OLE DB
- Oracle
- Sybase
- Sybase IQ
- Teradata

While this paper focuses solely on relational databases, it should be noted that SAS also offers access products for data warehouse appliances such as Aster, Netezza, Greenplum and Oracle Exadata, the HADOOP distributed file system, as well as legacy systems such as IMS and IDMS. (See http://www.sas.com/software/data-management/access/index.html#list for an up-to-date list.)

The complexity of the SAS/Access® software results from the necessity of sending database commands from the client to be executed by the server. The product is available in 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.

Client	Server	Available Interfaces
Unix	Unix	SAS/Access to <RDBMS>
Windows	Unix	SAS/Access to <RDBMS> SAS/Access to ODBC SAS/Access to OLEDB
Windows	Windows	SAS/Access to <RDBMS> SAS/Access to ODBC SAS/Access to OLEDB SAS/Access to PC File Formats

Table 1. SAS/ACCESS Interfaces by Platform

As noted above, this paper focuses on using an Oracle Windows client to access a database on a Linux host server, but this is only one of many possible combinations; the reader is directed to the extensive SAS documentation available for clues as to how to implement this functionality in specific local environments.

ACCESS TO RELATIONAL DATA

The ACCESS procedure (as distinct from the SAS/ACCESS interface product) as originally designed was extremely cumbersome to use. Some twenty years ago, I described it as "probably the worst designed SAS product of the decade". This was because in SAS Version 6 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. The DBLOAD procedure was a complementary way to bulk load data into an RDBMS. These products were described in earlier versions of this paper, but as the most recent SAS documentation indicates:

> For indirect access to DBMS data, you can use the ACCESS and DBLOAD procedures. Although SAS still supports these procedures for database systems and environments on which they were available for SAS 6, they are no longer the recommended method for accessing DBMS data. (SAS Institute, 2012a).

Instead, there are two different recommended ways to read relational data in SAS; the documentation notes that these are the *libname statement* and the *SQL Pass-through facility*:

- To assign SAS librefs to DBMS objects such as schemas and databases, you can use the LIBNAME statement. After you associate a database with a libref, you can use a SAS two-level name to specify any table or view in the database. You can then work with the table or view as you would with a SAS data set.
- To interact with a data source using its native SQL syntax without leaving your SAS session, you can use the SQL Pass-through facility. SQL statements are passed directly to the data source for processing. (SAS Institute, 2012a).

In SAS/Access 9.3 Interface software, there should be very little performance difference between these two choices, since the query push-down feature of the optimizer will try to run the submitted SAS code in the database if at all possible. For example, if you want to join two Oracle tables, as long as they are in the same schema, SAS will try to join them in Oracle and only return the results to the client, as opposed to copying both tables to the client and performing the join there, a considerably slower process.

As the following code illustrates, by using a "dynamic libname engine" SAS considers the database as if it were a SAS data library, and you can use the standard SAS data set options with an Oracle table or view.

```
LIBNAME oralib ORACLE USER="scott" PASSWORD=XXXXXXXX PATH="sample";

NOTE: Libref ORALIB was successfully assigned as follows:
      Engine:        ORACLE
      Physical Name: sample

PROC PRINT DATA=oralib.emp (WHERE=(deptno = 20));
       ID empno;
       VAR ename job;
RUN;
```

PASSWORD SECURITY

It may be that you do not want to hard code the database connection parameters into the program code. (I know I don't.) You can always use an encrypted password, but that still means that database access information is contained in your programs. Other strategies, such as using a `%include` file or macros to generate passwords have the same security flaws. The simplest fix is to try the following:

```
LIBNAME mydblib ORACLE DBPROMPT=yes;

PROC PRINT DATA=mydblib.emp;
       ID empno;
       VAR ename job;
       WHERE deptno = 20;
RUN;
```

You should see a database prompt, as follows. Note that you don't get the prompt until you actually try to connect, that is, in this example when the PRINT procedure attempts to access the Oracle table. You only have to supply the credentials the first time you connect.

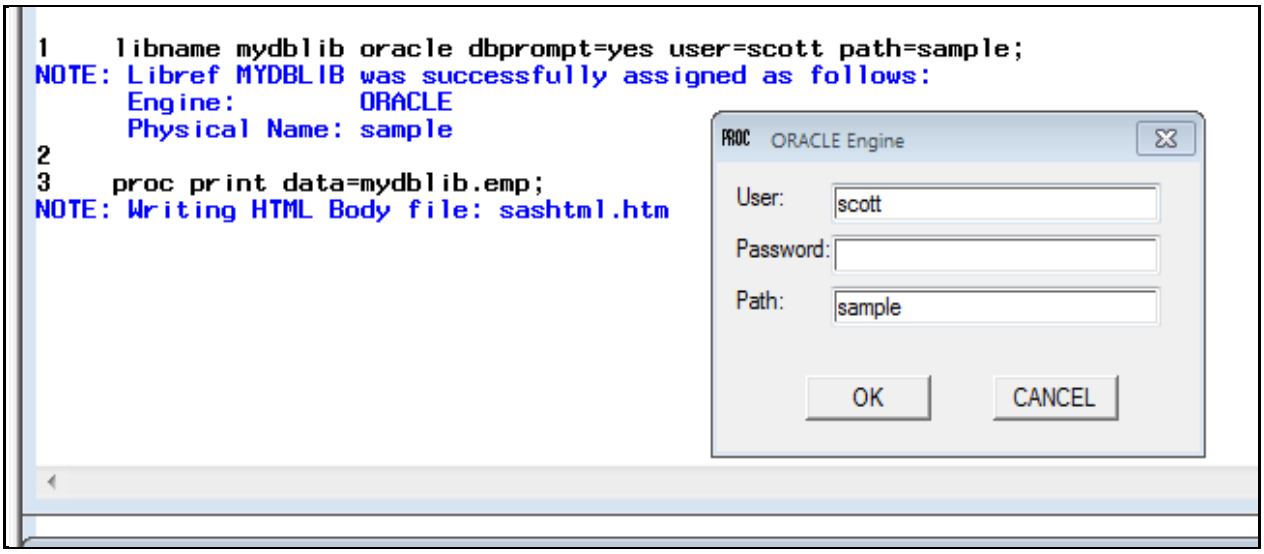

Figure 4. Using the libname statement interactively

Unfortunately, this simple solution does not always work. There is **SAS Usage Note 7980**, which issues the following warning:

> If remote submitting SAS/ACCESS code that includes DBPROMPT=YES to query for userid and password, an error will occur because the remote server has no attached terminal and no way to display a prompt window:
> ERROR: No terminal connected to the SAS session.

So this only works if you are actually on the SAS host, not submitting remotely, as for example from Enterprise Guide.

There is another approach available in SAS 93 which can be considered a best practice; this is to use an *authentication domain*:

```
LIBNAME oralib ORACLE AUTHDOMAIN="OracleAuth" PATH=sample;

NOTE:  Libref ORALIB was successfully assigned as follows:
       Engine:        ORACLE
       Physical Name: sample
```

This approach uses the SAS metadata server to store the database connection information, and is probably the most secure way to manage this issue. This is not something that a user can set up, but your SAS system administrator can create an authentication domain and add specific users or (more usually) user groups. For the administrator, it is a multi-step process, which requires defining a database server in SAS Management console, creating an authentication domain for that server, adding libnames to connect to the database, and adding users to the authentication domain; see support.sas.com/documentation/cdl/en/bisecag/63082/PDF/default/bisecag.pdf for the details.

Furthermore, in order to use this strategy, you must either connect to the metadata using SAS Enterprise Guide (or some other product that features a connection profile) or manually insert the metadata server definition in your SAS code. Despite what may seem a high degree of complexity, as the example illustrates this is the most secure way to connect to a database programmatically.

For more detailed information about database security, see Chapter 3. "Data Integrity and Security" in the *SAS Access 9.3 for Relational Databases Reference* manual, in the references at the end of the paper.

SQL PASS-THROUGH

You can always use PROC SQL instead of the DATA step to connect to the database and run queries against a previously defined libname, as follows:

```
proc sql;
       select empno, ename, job
       from oralib.emp (where=(deptno = 20));
run;
```

This should not be confused with the idea of SQL Pass-through, when the SQL code is executed by the database host rather than the client. SAS recommends that you use the libname statement in preference to PROC SQL, since it simplifies the coding. (There does not seem to be much if any performance difference.) However, sometimes it is necessary to execute database specific code, rather than SAS code, as when legacy or machine generated queries are used. The following example illustrates how to use PROC SQL as an alternative to the libname statement:

```
proc sql;
       connect to oracle as myora (authdomain="OracleAuth" path=sample);

       create table EMP as select * from connection to myora
          (select empno, ename, job from EMP where deptno = 20);

NOTE: Table WORK.EMP created, with 5 rows and 3 columns.

        disconnect from myora;
quit;
```

Three SQL statements are required: *Connect* and *disconnect* attach to the database and detach respectively. The SQL *select* statement has two parts: the parenthesized expression (`select empno, ename, job from EMP where deptno = 20`) is "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 myora* returns the result to SAS. Finally, the *create table* clause will cause the results to be saved as the work dataset EMP. If this clause were omitted, PROC SQL would simply display the table in the output window (the default behavior for a “select”).

DDL AND STORED PROCEDURES

In addition to select statements, it is also possible to submit DDL code using SQL Pass-through and the *execute* statement. Provided the user has sufficient privileges, it may be possible to create tables and/or views and assign permissions to them, just as if the user was connected directly to the database. For example, the following program creates a new view and assigns select privileges to user "frederick".

```
proc sql;
       connect to oracle(user=authdomain="OracleAuth" path=sample);

       execute (create view employee_dept as select dname, loc, emp.*
       from dept, emp where dept.deptno = emp.deptno) by oracle;

       execute (grant select on employee_dept to frederick) by oracle;

       disconnect from oracle;
quit
```

Note the difference from the previous example, which created a SAS dataset on the client and this one, where the data view is actually in Oracle on the host. The execute statement can also be used to run stored procedures, if the database supports them, and again, if the user has the appropriate privileges. The SAS/Access to Relational Databases Reference manual has much more detail on this topic, and the user is referred to the DBMS specific sections for the syntax and availability of this functionality.

ACCESS TO ODBC

In addition to the SAS/ACCESS Interface to DBMS, if the database is on a UNIX server and SAS is running on a Windows client, it is possible use SAS/ACCESS to ODBC. The main advantage of this approach, 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 one 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, ODBC for Oracle Client 11g. To create an ODBC DSN go to **Start>Control Panel>Administrative Tools** and click on **Data Sources (ODBC)**. The following menu should appear. (The Oracle ODBC client can be installed along with the other Oracle client software described above.)

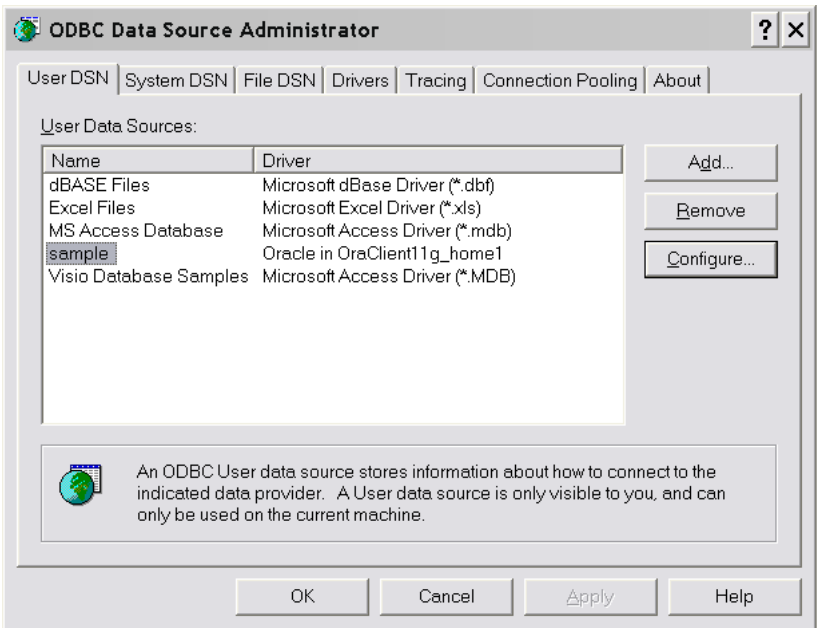

Figure 5. Configuring ODBC for Oracle

A User DSN (the tab shown) is only available to the current user, while a System DSN can be used by anyone connecting to this computer. To add a new data source, just click on add and select the appropriate driver; in this case it is “Oracle in Oraclient11g_home1”. You should then be prompted for the connection parameters, as follows:

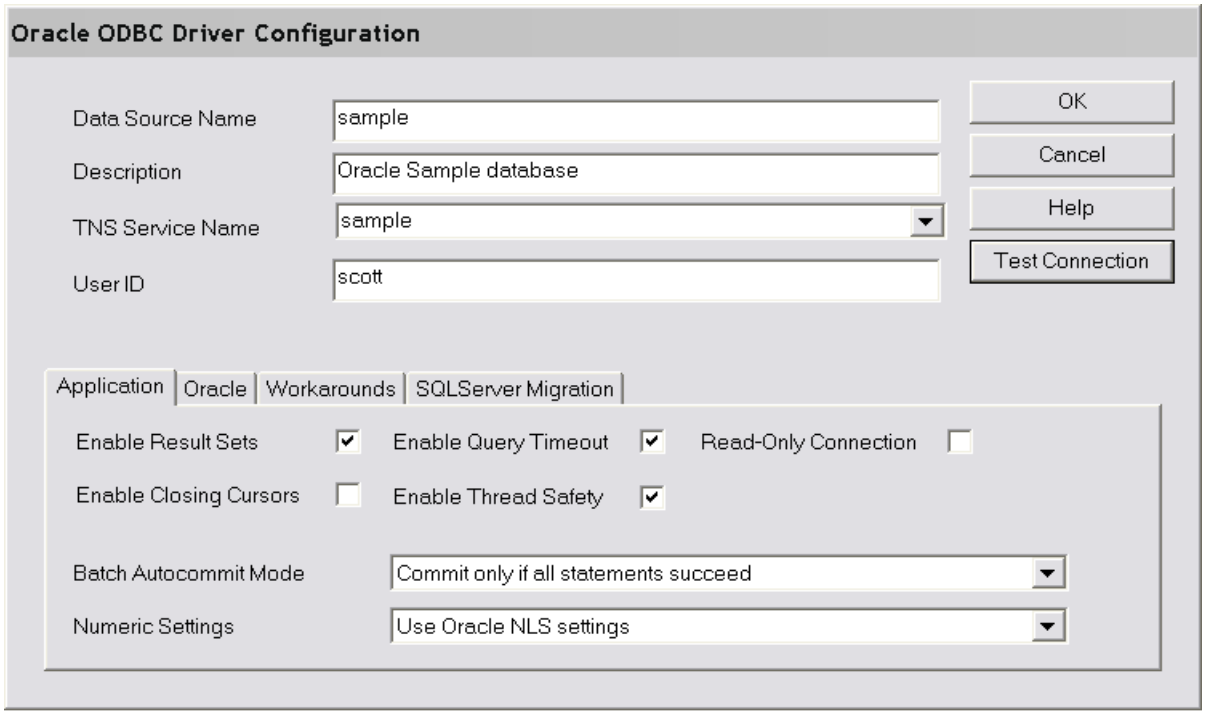

Figure 6. Oracle ODBC Driver configuration

Click on “Test Connection” to make sure that you can access the database. This only needs to be done once for each client workstation. As with the SAS/ACCESS Interface to Oracle, you can specify a `libname` reference:

```
LIBNAME odbclib ODBC DSN=”sample” USER=”scott” PASSWORD=”tiger”;
```

Alternatively, the database is accessible using the following SQL code:

```
proc sql;
       connect to odbc (dsn=sample 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.

ACCESS TO OLEDB

ODBC is an old technology; Microsoft recommends using the newer OLEDB interface instead. In most cases OLEDB gives better performance than ODBC, and does not requires creating a data source name on the client workstation. The only trick is that you need to know the name of the Oracle provider; in this case it is “`MSDAORA.1`”.

The syntax for the OLEDB `libname` statement should be quite familiar by now:

```
LIBNAME oradb OLEDB INIT_STRING="Provider=MSDAORA.1;Password=tiger;User ID=scott;
       Data Source=sample;Persist Security Info=True";
```

You can also connect to an OLEDB resource using the SQL Pass-through facility, as follows:

```
proc sql;
     connect to oledb
       (init_string="Provider=MSDAORA.1;Password=tiger;User ID=scott;
             Data Source=sample;Persist Security Info=True");
       create table EMP as
             select * from connection to oledb
             (select * from EMP);
quit;
```

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, delimited or spreadsheet files.

The easiest way to access the PC File Formats Interface is from the SAS Display manager `File` menu `Import` and `Export Data` commands, which run wizards that can be used to read and write PC files. There is an option in the Wizard to save the resulting SAS statements to a file. Here is an example of the result, which imports the `employees` table from the Microsoft Access Northwind sample database.

```
PROC IMPORT OUT=WORK.emp
     DATATABLE="Employees"
     DBMS=ACCESS REPLACE;
     DATABASE="C:\Program Files\Microsoft Office\OFFICE11\SAMPLES\Northwind.mdb";
     SCANMEMO=YES;
     USEDATE=NO;
     SCANTIME=YES;
RUN;
```

It is important to recognize that Microsoft Access, while relational, is not a true DBMS, since there is no database engine to connect to. All this code does is to make a copy of one file on the computer (the MS Access table) in another file format (SAS).

You can of course also connect to MS Access using ODBC. The following `libname` statement connects to the same table, without converting it, assuming that an ODBC data source has been set up with the name “Northwind” to point to the MS Access sample database.

```
LIBNAME odbclib ODBC DSN="Northwind";
```

IN-DATABASE PROCESSING

One additional topic needs to be considered in any overview of relational database access in SAS. This is the relatively new concept of "in-database processing." At present there are several different "Big Data" strategies to combine high performance computing with high availability, that is, systems that provide robust predictive analytics for industry and government users.

The most widely publicized idea is "cloud computing", basically just virtual servers available over the Internet or a corporate Intranet. SAS provides an Access interface for the HIVE file system, widely used in industry for HADOOP clusters. Many organizations are reluctant to move mission-critical analytics to the cloud because of concerns about security and especially performance. A second approach, then, is so-called "grid computing". In this model, several large computer systems are connected via a network, behind a corporate firewall. This offers significant advantages over the cloud, but still each computing server is limited in the amount of information that can be processed and the time it takes to accomplish a given data analysis. The SAS Platform LSF software is intended to support grid computing.

The third approach, increasingly evident in the information market place, is the "data warehouse appliance" from major vendors such as IBM Netezza, EMC Greenplum and Teradata. These are relatively low cost systems that come pre-configured for Data Marts out of the box. Unlike industry-leading database management systems such as Oracle or Microsoft SQL Server, these systems require little tuning and when properly setup can outperform their rivals by two orders of magnitude or more. Typically, MPCs can reduce database access times significantly by running queries in parallel across multiple processor nodes. Consequently, MPCs have rapidly growing market

penetration. For obvious reasons, these tend to be centered in the very largest organizations, who are concerned about the security of their data as well as the ability to do in-database analytics. The use of a data appliance for online analytic processing (OLAP) and predictive analytics has, for many organizations, become the preferred technology. The important difference that these systems provide is not just distributed processing but also distributed data, which makes it possible to analyze large volumes of data in parallel.

The SAS Access products described in this paper all work on a database appliance in essentially the same ways as on a traditional relational database system. For example, SQL Pass-through can be used with the `execute` statement to run R statistical routines on a Netezza system. As such, this is clearly an emerging technology area, and one in which SAS have provided some significant new tools for the statistician and analyst.

CONCLUSION

SAS software offers a number of options for relational database access, depending on client/server platform. The dynamic libname engines for Oracle, DB2, MySQL, ODBC, and OLEDB available in Version 9.3 are a great improvement over the access and view descriptors of the early releases. SAS continues to enhance `libname` support for external data sources of all sorts, and this, along with the SQL Pass-through facility allows users to access a wide variety of organizational data in an effective and efficient manner.

REFERENCES

- Frederick Pratter, 1993. "Desktop Database Management Using the SAS® System," Proceedings of the Sixth Annual Regional Conference, North East SAS Users Group. Springfield MA, November 7-9, 1993. www.lexjansen.com/nesug/nesug93/NESUG93000.pdf.
- SAS Institute Inc. 2012a. SAS/ACCESS® 9.3 for Relational Databases: Reference, 2nd Edition. Cary, NC: SAS Institute Inc. support.sas.com/documentation/cdl/en/acreldb/65247/PDF/default/acreldb.pdf.
- SAS Institute Inc. 2012b. System Requirements for SAS 9.3 Foundation for Microsoft Windows for x64, Cary, NC: SAS Institute Inc. support.sas.com/documentation/installcenter/en/ikfdtnwx6sr/64432/PDF/default/sreq.pdf.
- SAS Institute Inc., 2011a. System Requirements for SAS® 9.3 Foundation for Linux® for x64, Cary, NC: SAS Institute Inc. support.sas.com/documentation/installcenter/en/ikfdtnlaxsr/64419/PDF/default/sreq.pdf.
- SAS Institute Inc. 2011b. SAS® 9.3 Intelligence Platform: Security Administration Guide. Cary, NC: SAS Institute Inc. support.sas.com/documentation/cdl/en/bisecag/63082/PDF/default/bisecag.pdf.

CONTACT INFORMATION

Frederick Pratter
Destiny Corporation
2075 Silas Deane Highway
Rocky Hill CT 06067
http://www.destinycorp.com