

# Access to Relational Databases Using SAS ®

Frederick Pratter, Department of Computer Science, University of Montana, Missoula MT

## 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 and Sybase.

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.<sup>1</sup>

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 RELATION 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.

**Table 1. SAS/Access Interfaces by Platform**

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

The only option available for a user at a Unix client workstation is to use one of the database specific products<sup>2</sup>. 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)

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.

The following program shows how this works for a remote Oracle database table:

```
proc access dbms=oracle;
  create work.test.access;
  user="scott";
  orapw="tiger";
  table="scott.emp";
  path="sample";
```

<sup>1</sup> The SAS/Share® product was added to allow this functionality for reading SAS data libraries.

<sup>2</sup> Currently supported products include: DB2, CA-OpenIngres, Informix, ODBC, OLE DB, ORACLE, Oracle Rdb, SYBASE, MS SQL Server and Teradata.

```

assign=yes;
list all;
create work.test.view;
select EMPNO ENAME JOB;
subset where DEPTNO = 20;

proc print data=test;
id EMPNO;

```

This program first creates an "Access descriptor" called *test* in the work directory. The user ID, password, and server are all just as for the Oracle SQL Net client. Oracle field names are converted to SAS 6.12 format, and the contents of the database are listed. Note that the table "emp", owner "scott", has to be specified.

Then a "View descriptor", also called *test* is created containing some of the rows and columns from the remote table EMP. The resulting database view can then be used to read (or update) the database, according to the Oracle permissions available to the specified user.

Note that the syntax for PROC ACCESS is very specific; for example, even though the access and view descriptors are created in the temporary WORK library, it is still necessary to specify the three level names *work.test.access*, and *work.test.view*.

In Version 7 and higher, 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"
preserve_tab_names=yes;

```

## PROC DBLOAD

The DBLoad procedure works in the opposite direction. 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.

The following example illustrates creating a new Oracle table from SAS using DBLoad:

```

proc dbload dbms=oracle data=test;
orapw="tiger";
user="scott";
path="sample";
table="emp2";
label;
reset all;
load;

run;

```

The syntax is fairly close to that of PROC ACCESS. A SAS file called "test" is copied to Oracle as a table called "emp2". SAS variable labels are used as Oracle field names. Note the "load" statement—if this does not appear the table will not be created.

## SQL PASS-THROUGH

The least complicated way to manage remote database tables in Version 6 is with PROC SQL. This alternative offers a 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. It is important to note however that there are DBMS specific differences from system to system. It is also necessary to license Oracle SQL Plus on the client workstation. The best advice is to consult the documentation that comes with the specific version of the SAS/Access® product, and be sure to coordinate with your DB administrator before accessing her tables remotely!

The following example illustrates how to use PROC SQL 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 required: *Connect* and *disconnect* attach to the database and detach respectively. The SQL *select* statement has two parts: the parenthesized expression (*select \* from EMP*) 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 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 “select”).

## 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 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, 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 DSN. This 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.

One can also use DBLoad with ODBC to create an Oracle table from a Windows SAS dataset:

```
proc dbload dbms=odbc data=mylib.dept;
  dsn="oracle";
  uid="scott";
  pwd="tiger";
  table="emp2";
  limit=0;
  load;
run;
```

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 as if they were SAS datasets:

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

Obviously this is simpler for users unfamiliar with SQL syntax, and can replace both the PROC SQL and PROC DBLOAD syntax shown above.

## 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 . Also, PROC IMPORT and PROC EXPORT are available as standalone procedures, not just as menu items. The new release also includes the CV2ODBC conversion utility, since Access to SQL Server no longer available; SAS currently recommends licensing the ODBC product to read SQL server files.

## 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 7 and 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 offer libname support for external data sources of all sorts.

## REFERENCES

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

"SAS/ACCESS Sample Programs for UNIX." [http://www.sas.com/service/techsup/sample/unix\\_access.html](http://www.sas.com/service/techsup/sample/unix_access.html). SAS Institute. Cary, NC.

"SAS/ACCESS Software, Release 8.2: What's New Highlights." <http://www.sas.com/products/sassystem/release82/accessnew.html>. SAS Institute. Cary, NC.

"Available SAS/ACCESS Interfaces." <http://www.sas.com/rnd/warehousing/access/available.html>. 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" (02Apr96). SAS Institute. Cary, NC.

"TS-518D: SAS/ACCESS® Guidelines for Connecting to ORACLE® Databases in the UNIX® Environment" (16Oct97). SAS Institute. Cary, NC.

"TS518: SAS/ACCESS Guidelines for Connecting to Relational Database Systems in the UNIX Environment" (16Oct97). 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". SUGI 25-91. SAS Institute. Cary, NC.

## CONTACT INFORMATION

Frederick Pratter  
Department of Computer Science  
University of Montana  
Missoula, MT 59812  
[pratter@cs.umt.edu](mailto:pratter@cs.umt.edu)