USING SAS® WITH ORACLE® PRODUCTS FOR DATABASE MANAGEMENT AND REPORTING

Henry W. Buffum, R. O. W. Sciences, Inc.
Darryl J. Keith, U.S. Environmental Protection Agency

Abstract:

Data for a large environmental monitoring project that were managed for several years in SAS® datasets were recently loaded into an Oracle® database. In order to meet the needs of users who rely on SAS for reporting and statistical analyses, an interface between Oracle and SAS must be maintained. SAS provides two products that facilitate this interface: the SQL procedure, and the SAS/ACCESS facility. This paper will describe how these tools were used to convert a database from SAS datasets to Oracle tables and to maintain links between Oracle and the SAS system. Special attention will be paid to: (1) Proc SQL, which allows SAS to process Standard Query Language commands; (2) SAS/ACCESS view descriptors, which can be created to make Oracle tables directly accessible to SAS procedures; (3) the SQL pass-through facility, which makes it possible to submit SQL commands to the Oracle server from a SAS job; and (4) the DBLOAD procedure, which can copy a SAS dataset directly into an Oracle table. Problems encountered in configuring these tools to run on a clustered VAX environment under VMS will be discussed, and possible uses of multi-platform client-server configurations will be suggested.

Introduction:

The Environmental Monitoring and Assessment Program (EMAP) is a nationwide program designed to monitor and assess the condition of the nation's ecological resources. The EMAP Estuaries resource group is responsible for monitoring the estuarine environment of the United States' coastline. One section of the coast line, the Virginian Province (extending from Cape Cod MA to Cape Henry VA) has now been sampled for four consecutive years. Each year, approximately 6000 samples of sediment, fish tissue and water were collected, along with many field observations and continuous measurements from electronic instruments. All field observations and data resulting from laboratory analyses of these samples were originally stored in SAS datasets on a VMS platform. SAS was chosen as a database management system primarily for its statistical modules, and also for its flexible data management facilities. During the first four years of the project, many SAS programming techniques were utilized to meet data management needs, including complex data step programming (Buffum et al, 1993), SAS/AF® application development (Battista et al, 1993), and extensive use of the SAS macro language and Proc SQL.

From the program's beginning, the information management plan called for EMAP data to be stored in a relational database management system (RDBMS). Because large quantities of data collected by different resource groups would be stored at several sites, a powerful RDBMS is needed to manage a distributed database. In 1993, the Oracle system was chosen to be the EMAP RDBMS.

While Oracle will now house the EMAP data, and Oracle tools such as SQL*PLUS and SQL*FORMS will be used to perform most data management tasks, the SAS system will continue to play a major role in analyzing data and providing reports. In order to maintain the link between SAS and Oracle, the SAS/ACCESS module was purchased. This module, along with SQL procedure available with base SAS, allows a flow of data between the two software systems.

This paper deals with the major tasks that faced the EMAP Virginian Province data management
team as the transition from a SAS database to Oracle occurred. First, we had to learn which of the tools available from Oracle and SAS would best facilitate the transfer of data between the two software systems. Next we had to transfer the data from SAS datasets into an Oracle database. Third, we had to create a means of accessing Oracle tables from the SAS system.

**Learning the Tools:**

Learning to use Oracle and the SAS/ACCESS to Oracle tools can be frustrating for a veteran SAS programmer. Not only is the command syntax used by these tools different from that of traditional SAS programming; the logic behind the syntax is also quite different. It is important to become familiar with some basic concepts behind the tools.

**SQL**

Oracle, like most other RDBMSs, uses Standard Query Language (SQL) to process data. SQL was developed by IBM®, and has since been adopted by the American National Standards Institute (ANSI) as the standard language for relational database management systems.

Two concepts that are central to SQL programming must be understood: that of a query, and that of a view. A SQL query is simply a means of getting data out of one or more database tables. SQL queries use a very succinct command syntax, and can select data elements from several different tables in one step. SQL queries can be used to create reports, and also to create new tables and views.

A SQL view is a kind of dataset that retrieves data stored in other physical locations. A view does not actually contain data; instead it provides the query logic that points to data stored in other tables. Thus new views may be created and stored without using much disk space.

Along with the query commands (which can be used to create views), SQL also includes non-query commands that can be used to drop tables, grant and revoke access to tables for specific users, and other data management tasks. Examples of SQL query and non-query commands will be provided below.

**Oracle tools**

Although Oracle software is often thought of as one unified system, Oracle Corporation actually markets several different products. It is important to distinguish between these. The main component of the Oracle system, and the only one needed to create an Oracle database, is the database server (sometimes called the kernel). The Oracle server is a complete RDBMS - a system for storing and maintaining data in relationally joined tables. It has the ability to process SQL commands, and also includes many data dictionary tables and views, and utility programs to manage data and security.

Additional tools available from Oracle include a programming system called SQL*Plus®, and an additional set of programming commands called PL/SQL™. SQL*FORMS® is an online applications development system that can be used to create complex interactive systems for an Oracle database. SQL*REPORT® is used to create reports from the database. Many additional tools are available from Oracle corporation.

While many of these tools are being used by the EMAP information management team, they will not be discussed in this paper. One other product that is important to any client-server applications is SQL*NET. This is Oracle’s networking software, that allows products based on different platforms to communicate with each other. If a client tool, such as SAS/ACCESS, resides on a separate machine, it will need SQL*NET to connect to the server. SQL*NET needs to be installed on both the client and the server machine.

The Oracle server provides a sophisticated database security system that allows the database administrator (DBA) to grant access privileges for database objects to different groups of users. All database users must have an Oracle user id. and
password. For any of the Oracle tools to gain access to the database server, the user id and password must be entered. This is also true for any third party client tools (such as SAS/ACCESS) that access the server. The same privileges apply to an Oracle user no matter which tool is used to access the server. SAS/ACCESS provides some additional security features, but the Oracle DBA must take the primary responsibility for database security.

**Proc SQL**

The first step in learning to use any RDBMS such as Oracle is to become familiar with SQL. The base SAS system now includes the SQL procedure, which provides SAS programmers an excellent chance to learn and use this language. This procedure allows SAS programmers to use SQL to query SAS datasets, and to create and store views of datasets. With the addition of the SAS/ACCESS product, Proc SQL can be used to query and create views of data in a foreign database.

Using PROC SQL queries and views can be very useful to SAS programmers, even if foreign databases are not being accessed. The following examples illustrate the way that traditional SAS code and Proc SQL perform the same query

1) Using traditional SAS code:

```sas
DATA TEMP;
MERGE V90.SAMPLOG
   V90.EVENTLOG;
BY EVNTNUM EVNTDATE;
KEEP STATION EVNTNUM EVNTDATE SAMPLEID;
IF SAMP'YP.E="SEDCBM";
PROC SORT; BY STATION EVNTDATE;
```

2) Using Proc SQL:

```sql
PROC SQL;
CREATE VIEW 1EMP AS
   SELECT STATION, A.EVNTNUM, A.EVNTDATE, SAMPLEID
   FROM V90.SAMPLOG AS A, WO.EVENTLOG AS B
   WHERE A.EVNTNUM=B.EVNTNUM AND
     A.EVNTDATE=B.EVNTDATE AND
     SAMP'YP.E="SEDCBM"
ORDER BY STATION, A.EVNTDATE;
```

The real advantage of Proc SQL comes into play in more complex queries, where several datasets are joined. A query to get variables from several different SAS datasets, that would require several SORT and MERGE steps in traditional SAS coding, can often be written in just a few lines of SQL code. Also, SQL's summary functions can be very useful to SAS programmers. However, newcomers to Proc SQL should study the manual carefully before proceeding. SQL's succinct but powerful syntax make it very easy to encode logic errors that produce enormous, incorrect output files.

In some (but not all) cases, Proc SQL queries make more efficient use computer resources than traditional SAS code. And storing SQL views of SAS datasets instead of physical datasets can help conserve disk space.

**SAS/ACCESS to Oracle**

The SAS/Access to Oracle module consists of two new procedures- Proc ACCESS and Proc DBLOAD; and a new feature that is used with the SQL procedure called SQL pass-through. Proc ACCESS allows the SAS user to create view descriptors of Oracle tables. These view descriptors are similar in many ways to SQL views: they appear to the SAS system as datasets, but they do not contain any actual data. Instead, they provide pointers to data stored in an Oracle table. An Access view can be used to read and to write to the Oracle table. However, unlike a SQL view, access descriptors can only point to a single Oracle table.

Proc Access provides a user-friendly, on-line facility for creating view descriptors, as well as the means to create them in batch mode.

The DBLOAD procedure can be used to load data from a SAS dataset directly into an Oracle table. This procedure actually creates the Oracle table and populates it with data in one step. Variables that have SAS date formats attached to them are automatically converted to Oracle date format, and missing values are converted to null. The DBLOAD procedure includes many options to...
control the structure of the resulting Oracle table. However, some important parameters that should be set when creating an Oracle table, such as the primary key field and certain storage parameters, can not be defined with the SAS DBLOAD procedure.

The DBLOAD procedure also provides the SAS user with a means of passing non-query SQL commands to the Oracle server to execute. This feature allows simple database management tasks to be accomplished from within a SAS session.

Finally, the SQL pass-through facility provides the means to send query and non-query SQL commands to the Oracle server to execute. This facility is executed from within the SQL procedure. The pass-through facility allows Proc SQL queries to not only access the Oracle tables, but to actually let the Oracle server process the query. Using the Oracle server in this way is generally the best way to optimize computer resources.

The SQL pass-through facility allows views of the Oracle database to be created that can then be read by SAS procedures. Unlike view descriptors created with Proc Access, single views can be coded that select data from more than one Oracle table. However, these views can only be used by SAS in the read mode. They can not be used to add to or modify the Oracle database.

All of these SAS/Access tools require an Oracle user id and password in order to reach the database. The security privileges that are granted to the SAS program are those that have been granted to the user id that is named. If SQL*NET is used to access the Oracle server, the SAS/Access routines will require a pathname that defines the SQL*NET protocol.

These tools are all well documented in the SAS/ACCESS to Oracle Usage and Reference, as well as in some recent SUGI papers (see Stranieri, 1991, for a useful overview). The following examples of code will describe how these tools can be used to perform specific tasks.

Converting data from SAS to Oracle

The first task in converting a SAS database to an Oracle database is to consider redesigning the table structures. It may not be wise to simply convert SAS datasets to Oracle tables with the identical structure. To use the best features of the Oracle system, the database should be fully normalized--redundant data elements should be eliminated. Primary key fields should also be identified in all tables to uniquely identify each row.

The EMAP Virginian province database, as it was originally implemented in SAS, incorporated these design principals. Therefore, very little redesign was required for our conversion to Oracle. However, some modifications to tables were required. These design changes were first applied to the SAS datasets themselves. Next, the redesigned SAS datasets were converted directly to Oracle tables.

There are at least three methods for converting a SAS dataset to an Oracle table. The easiest method is to rely completely on the SAS DBLOAD procedure. But this method has some disadvantages--there are important features of an Oracle database that can not be utilized with this approach. An alternative method is create an empty table with Oracle SQL statements, write out the data from the SAS dataset to an ASCII file, and then use the Oracle utility program SQL*LOADER to load it into the table. This method is quite cumbersome, but allows you to be use all the features of the Oracle system. Another alternative combines the best elements of the first two methods, using Oracle SQL commands to create the database table, and SAS tools to load it with data.

1) Using Proc DBLOAD

To create an Oracle table from a SAS dataset with Proc DBLOAD, you simply name the SAS dataset in the data= clause, and the output Oracle table in the table= clause. An optional WHERE clause can control the observations that are copied into the Oracle table.
Host Systems

DBLOAD Example:

```
PROC DBLOAD DBMS=ORACLE
   DATA=VA90_SAMPLE_LOG;
   TABLE=VA90_SAMPLE_LOG;
   WHERE EMAPYEAR=1990;
   USER=****;
   ORAPW=****;
   PATH='@D:\NARDEB:"TASK=ORNORA_PROD"';
   TABLESPACE=EMAPV_DATA LOAD;
RUN;
```

The USER, ORAPW, and PATH statements provide the Oracle user id, password and path needed to access the server. A long list of options allows the SAS procedure to define features of the Oracle table. (Here, the optional TABLESPACE statement names an Oracle-defined data storage area). However certain important Oracle table parameters cannot be specified, including storage parameters and the primary key parameter. When creating a permanent Oracle database, it is important to specify these parameters. Therefore, I recommend using this method only for creating temporary Oracle tables, where optimization is not so important.

2) Using Oracle's SQL*LOADER

In order to specify all of the Oracle table parameters, a SQL script must be submitted to the Oracle server to create an empty table. The following Oracle SQL code defines the same table that was created by the previous DBLOAD example, but defines additional storage parameters, and a primary key field:

```
CREATE TABLE VA90_SAMPLE_LOG
   (EVNTDATE DATE,
    SAMPLEID NUMBER NOT NULL,
    EVNTNUM NUMBER,
    SAMPTYPE VARCHAR(6),
    PRIMARY KEY (SAMPLEID))
   TABLESPACE EMAPV_DATA
   STORAGE(INITIAL 509107 NEXT 2048
            MINEXTENTS 1 MAXEXTENTS 100 PCTINCREASE 0)
    PCTFREE 10 PCTUSED 80;
```

This code can be submitted directly to Oracle, using SQL*Plus. Alternately, this SQL command could be submitted to the server from SAS, using the SQL statement of Proc DBLOAD. (See next example.)

Once the table has been created, Oracle’s SQL*LOADER can be used to read raw ASCII data into the table. The SQL*LOADER requires a control file to be written to define the input data file, that is similar to a SAS INPUT statement. The SQL*LOADER routine provides a detailed listing file that gives statistics on the load. This routine also allows any rejected records to be written to an error table.

This method can be used to convert SAS datasets to Oracle tables without using any of the SAS/ACCESS tools. Most of the permanent EMAP database tables were created using this technique.

3) A Hybrid Method

A third method of converting data in a SAS dataset into an Oracle table combines elements of the first two methods. This approach requires using all of the SAS/ACCESS tools. First, the CREATE TABLE command that creates an empty table with all the correct parameter settings is submitted to the Oracle Server using the SQL statement of the DBLOAD procedure. Once the empty Oracle table is created, and PROC ACCESS is used to create an Access view descriptor for that table. Finally, the SAS Append Procedure is used, pointing to the Access Descriptor as the "base" dataset, and the SAS dataset containing the data as the "data" dataset. The data elements in the SAS dataset are thus added to the Oracle table by going through the Access View descriptor.

This method has the advantage of having the table created explicitly with an Oracle SQL command defining the necessary parameters, and can be accomplished without leaving the SAS system.

The following code uses Proc DBLOAD to issue two SQL statements: the first to drop the existing VA90_SAMPLE_LOG table, and the second to create a new one:
PROC DBLOAD DBMS=ORACLE;
USER=******;
ORAPW=******;
PATH='@D:NARSRV::"TASK=ORDNORA_PROD"';
SQL
DROP TABLE VA90_SAMPLE_LOG;
SQL
CREATE TABLE VA93_SAMPLE_LOG
(EVNTDATE DATE,
SAMPLEID NUMBER NOT NULL,
EVNTNUM NUMBER,
SAMPTYPE VARCHAR(6),
PRIMARY KEY (SAMPLEID))
TABLESPACE EMAPV
DATA STORAGE (INITIAL 509107 NEXT 2048 MINEXTENTS 1 MAXEXTENTS 100 PCTINCREASE 0)
PCTFREE 10 PCTUSED 80;
RUN;

The next section of SAS code runs Proc Access in batch mode to create an access view descriptor of the Oracle table created in the above step. The descriptor is given the name TEMP in this example. The SELECT statement uses the keyword ALL to include all table elements (variables) in the descriptor.

PROC ACCESS DBMS=ORACLE;
CREATE ACC.TEMP.ACCESS;
USER=******;
ORAPW=******;
PATH='@D:NARSRV::"TASK=ORDNORA_PROD"';
TABLE=VA99_SAMPLE_LOG ASSIGN=YES;
CREATE ACC.TEMP.VIEW;
SELECT ALL;
PROC FREQ DATA=ACC.TEMP;
TABLES SAMPTYPE;

Finally, Proc Append is executed to append all observations in the SAS dataset to the Oracle table. The final Proc step will print all the data from the Oracle table.

PROC APPEND BASE=ACC.TEMP
DATA=V90.SAMPLOG;
RUN;
PROC PRINT DATA=ACC.TEMP;

Accessing Oracle from SAS

Once the data has been successfully loaded into the Oracle database, the next task is to make it accessible to SAS procedures.

The two options for making data in Oracle tables available to SAS are 1) through Access View Descriptors, and 2) through views created with the SQL pass-through facility. As noted before, Access view descriptors allow read and write access to the Oracle tables, whereas views created with SQL pass-through do not.

The following examples of code both create a means of accessing data in the Oracle table VA90_SAMPLE_LOG. The first used Proc ACCESS:

PROC ACCESS DBMS=ORACLE;
CREATE ACC.TEMP.ACCESS;
USER=******;
ORAPW=******;
PATH='@D:NARSRV::"TASK=ORDNORA_PROD"';
TABLE=VA90_SAMPLE_LOG ASSIGN=YES;
CREATE ACC.TEMP.VIEW;
SELECT ALL;
PROC FREQ DATA=ACC.TEMP;
TABLES SAMPTYPE;

The next example accomplishes basically the same thing, using Proc SQL and the SQL pass-through facility:

PROC SQL;
CONNECT TO ORACLE AS DB
(USER=XXXXX
ORAPW=XXXXX
PATH= '@D:NARSRV::"TASK=******"');
CREATE VIEW ACC>TEMP AS
SELECT *
FROM CONNECTION TO DB
(SELECT * FROM VA90_SAMPLE_LOG);
DISCONNECT FROM DB;
PROC FREQ DATA=ACC>TEMP;
TABLES SAMPTYPE;

The SQL SELECT statement that comes directly after FROM CONNECTION TO statement, and is contained within parentheses, is passed to the Oracle server to process. In this example, the SQL pass-through option does not provide much advantage over the Proc ACCESS method. The advantages to using SQL pass-through come into play when complex queries are passed to the Oracle server. If several tables are joined, or records are summarized, or large amounts of data are reordered, it is most efficient to have the Oracle server perform these tasks. This can be accomplished by passing the necessary SQL code to Oracle.
Host Systems

Both these methods of accessing Oracle data from SAS have their uses. But to fully utilize the power of an RDBMS such as Oracle, complex queries should be sent to the server for processing.

Conclusion

The EMAP-Estuaries Virginian Province data management team has been working with SAS/ACCESS tools with an Oracle database for several months now. The interface between the two software systems is so smooth as to be virtually transparent to most users. The challenge that remains for us is to configure our SAS client tools in such a way as to use the Oracle server most efficiently. Utilization of the SQL pass-through facility an important tool in accomplishing this end.

References


Trademarks

SAS, SAS/AF and SAS/ACCESS are registered trademarks of SAS Institute Inc. in the USA and other countries. * indicates USA registration.

Oracle is a registered trademark of Oracle Corporation.