

Exporting Oracle Tables into SAS Tables Made Simple with Version 7 of The SAS System

Danny Walker
Delta Technology, Atlanta, Georgia

Abstract

Over the past few years Delta Air Lines has been involved in a major upgrade of their information system technologies. Delta Technology was created as a wholly owned subsidiary of Delta Air Lines to facilitate this process. Part of this effort was moving legacy mainframe systems to alternate platforms such as UNIX. Some were re-written as others were replaced with third party software. Tools and architecture were also standardized which include the use of relational database systems such as Oracle.

As these new systems become operational, the analytical community at Delta continues to require quick and seamless access to their data. This paper demonstrates how to easily extract Oracle data into SAS tables using only Base SAS, Oracle SQL/Plus, and SQL/NET on HP V class 2250 UX11.0 connecting to Oracle 7.3.4.

Background

In the renovation process of software systems at Delta Air Lines, several applications were identified for immediate attention. One such system was Flight Scheduling. The current application was written in a number of languages and resided mostly on mainframe VM/CMS. Its primary function is to publish a schedule on a weekly basis. This schedule would then be distributed to various groups which included the reservation agencies. Ideally, Delta would like to publish the schedule on a daily basis to better respond to competitor activity in the marketplace. To accomplish this, a third party software application from The Sabre Group was chosen as opposed to

an in-house re-write. Sabre's Integrated Flight Scheduling System (IFSS) demonstrated the capability of handling most of the scheduling challenges that Delta was facing. IFSS consisted of a core Oracle database that several other support modules could feed from.

One of the supporting modules - The Airline Profitability Model (APM) - also required SAS software for calibration of the model. APM allows analysts to make schedule changes such as departure/arrival time, type of equipment, flight insertion and deletions, etc. and calculate resulting marginal revenue. The model is a result of an exhaustive logistic regression procedure which will predict the probability that a passenger will chose a flight based on equipment, day of week, time of day, origin, destination, etc. The SAS tools required to accomplish this are Base, FSP, STAT, ETS, and Graph. In addition, we added SAS/Access to Oracle to the mix to support the analysts in their quest for data.

Problem

We installed SAS 6.12 and the above mentioned tools and went to work on calibrating the model. After this was well underway, we began testing the SAS/Access to Oracle product and discovered shared library problems. After discussing this with SAS Technical Support, we were advised to install SAS 7.0 and a patch from their ftp site. Apparently there was no fix for this problem on 6.12. After installing 7.0 and the patch, we continued to experience problems with Access. For several weeks subsequent, Delta Technology and SAS Technical Support exchanged numerous

listings and libraries in an attempt to resolve this problem. After working closely with their on-site HP staff, SAS resolved the problem and we then downloaded the corrected file and went to work with Access.

Solution

During this Access trouble shooting period, we went ahead with a method for extracting Oracle tables into SAS tables using the tools we currently had, namely Base SAS. Although only a novice SQL coder, I knew how to direct the results of a describe and select to an output file. The describe file could then be read in by SAS and used to create another query to write the whole table to another file in delimited format. Finally, this delimited file could be read into a SAS data set. But what makes the process simple is the ability of SAS Version 7 and 8 to handle long data set and variable name lengths. The resulting SAS data set (now referred to as a table) will have the same name as the Oracle table and the SAS variables will have the same name as the Oracle column names.

Process

The following is a description of the SAS process along with the source code and intermediate files created by the SQL queries.

The beginning of the program is set up to accept parameters for table name and security information. Generally, this is all that is required to be changed for any given table export.

```

*Oracle Database ==>; %let db=ifssprod;
*Oracle Table    ==>; %let table=icpr_acf;
*Oracle User     ==>; %let user=ora_user;
*Oracle Password ==>; %let pw=ora_pw;

*Scratch Pad Dir ==>; %let dir=/home/danw/;

```

```

*SASV7 Dir      ==>; %let sasdir=/home/danw
                                     /sasv7;

```

```

title;
filename desc "&dir.desc_&table..sql";
filename in "&dir.&table..desc";
filename exp "&dir.exp_&table..sql";
filename pgm "&dir.&table..sas";
libname out "&sasdir";

```

Its important to note that the SASV7 directory is specified because SAS will use the V6 engine to write out data sets if there are any V6 data sets present.

The next code sequence creates the describe query and executes it.

```

data _null_;
  len=length("&table");
  file desc;
  put @1 'set heading off;' /
      @1 'spool '
      @7 "&table"
      @(7+len) '.desc;' /
      @1 'describe'
      @10 "&table"
      @(10+len) ';' /
      @1 'exit;';

```

```

data _null_;
  call system("sqlplus &user/&pw@&db
               @desc_&table");

```

Generated SQL query:

```

set heading off;
spool icpr_acf.desc;
describe icpr_acf;
exit;

```

Results of describe:

Name	Null?	Type
-----	-----	----
ACF_OWN	NOT NULL	VARCHAR2(3)
EQP_CD	NOT NULL	VARCHAR2(4)
FLE_CD	NOT NULL	VARCHAR2(3)
ACF_DSC	NOT NULL	VARCHAR2(20)
ER_IND	NOT NULL	VARCHAR2(1)
STG_Nr		VARCHAR2(1)
WT_TON		NUMBER(6)
MTOW		NUMBER(6)
ACF_SVC_TYP		VARCHAR2(1)

```

CORP_IND          VARCHAR2(1)
ACF_SPEED         NUMBER(5)
COMMUT_IND        VARCHAR2(1)
SSIM_EQP_CD       VARCHAR2(4)
ICAO_EQP_CD       VARCHAR2(4)
CREW_GRP_CD       VARCHAR2(4)
EMP_NR            VARCHAR2(6)
TMST              NOT NULL   DATE

```

This section of the code reads in the result of the describe, keeps the relevant records, and concatenates a pipe to the variables that will be used in the SQL select statement. This will cause the output variables to be delimited by the pipe.

```

data read;
  infile in length=l;
  input @;
  l_type=l-42;
  if l gt 43 then
    input @2 var $30.
      @43 type $varying200. l_type;

```

```

if substr(type,1,1) in('V','N','D');

```

```

data delimit;
  set read end=eof;
  l=length(var);
  newvar=(var||"|" || "");
  if eof then do;
    newvar=substr(newvar,1,l);
  end;

```

We now generate and execute the SQL that will produce an ASCII file with all of the columns delimited by a pipe.

```

data _null_;
  set delimit end=eof;
  l=length("&table");
  file exp;
  if _n_=1 then put @1 'spool off;' /
    @1 'set heading off;' /
    @1 'set echo off;' /
    @1 'set pagesize 2000;' /
    @1 'set linesize 350;' /
    @1 'spool'
    @7 "&table..dat;" /
    @1 'select';
  put @1 newvar $38.;
  if eof then put @1 'from'
    @6 "&table"
    @(6+l) ';' /
    @1 'exit;';

```

```

data _null_;
  call system("sqlplus &user/&pw@&db
    @exp_&table");

```

The following is the query generated and a sample of the output file.

```

spool off;
set heading off;
set echo off;
set pagesize 2000;
set linesize 350;
spool icpr_acf.dat;
select
ACF_OWN          |' '|
EQP_CD           |' '|
FLE_CD           |' '|
ACF_DSC          |' '|
ER_IND           |' '|
STG_NR           |' '|
WT_TON           |' '|
MTOW             |' '|
ACF_SVC_TYP      |' '|
CORP_IND         |' '|
ACF_SPEED        |' '|
COMMUT_IND       |' '|
SSIM_EQP_CD      |' '|
ICAO_EQP_CD      |' '|
CREW_GRP_CD      |' '|
EMP_NR           |' '|
TMST
from icpr_acf;
exit;

```

Sample output generated:

```

AF |146 |146 |BRITISH AERO 146 |N |3 |0 |0 |J || |A | | - |
AF |310 |310 |AIRBUS INDUS A310 |Y |3 |0 |0 |J | | | | |
AF |32S |320 |AIRBUS INDUS A320 |N |3 |0 |0 |J | | | | |
AF |340 |340 |AIRBUS INDUS A340 |Y |3 |0 |0 |J | | | | |
AF |733 |733 |BOEING 737-300 |N |3 |0 |0 |J | | | | | |Ksig
|23-
AF |735 |735 |BOEING 737-500 |N |3 |0 |0 |J | | | | | |Ksig
|23-
AF |737 |737 |BOEING 737 |N |3 |0 |0 |J | | | | | |Ksig |23-
AF |743 |747 |BOEING 747-300 |Y |3 |0 |0 |J | | | | | |Ksig |23-
AF |744 |747 |BOEING 747-400 |Y |3 |0 |0 |J | | | | | |Ksig |23-

```

We can now create a SAS program to read in this file. The first data step will generate SAS code which defines the lengths for all the variables. The second data step will generate the code for variable input. Lastly we add code to remove the last three records. These records at the end of the

SQL listing have two blank lines and information on the number of records retrieved.

```
data _null_;
  set delimit end=eof;
  l1=length("&dir&table..dat");
  l2=length("&table");
  l3=length(var);
  start=index(type,'(');
  end=index(type,');');
  if type='V' then
    fmt="$||substr(type,(start+1),(end-start)-1);
  if type='D' then
    fmt='$9';
  if type='N' then
    fmt='4';
  file pgm;
  if _n_=1 then
    put @1'filename in "'
      @14 "&dir&table..dat"
      @(14+l1) ""
      @(14+l1+1) ';' /
      @1 'data'
      @6 "&table"
      @(6+l2) ';' /
      @1 "infile in dlm='|' missover
        firstobs=2;" /
      @1 'length';
  put @1 var
      @(l3+2) fmt;
  if eof then put @1 ';' /
      @1 'input';
```

```
data _null_;
  set delimit end=eof;
  l1=length("&table");
  start=index(type,'(');
  end=index(type,');');
  file pgm mod;
  put @1 var;
  if eof then do;
    put @1 ';' /;

    put @1 'data _null_;' /
      @1 "set &table" /
      @1 'end=eof;' /
      @1 'if eof;' /
      @1 "call symput('recs',_n_);" /
      @1 'run;' /;
  put @1 '%macro sub;' /
      @1 'data'
      @6 "out.&table"
      @(10+l1) ';' /
      @1 "set &table"
```

```
@(6+l1) ';' /
@1 'if _n_ le (&recs-3);' /
@1 '%mend;' /
@1 '%sub;' /
@1 'run;';
end;
run;

%include pgm ;
run;
```

The final SAS code which is immediately included is as follows:

```
filename in "/home/danw/icpr_acf.dat";
data icpr_acf;
infile in dlm='|' missover firstobs=2;
length
ACF_OWN $3
EQP_CD $4
FLE_CD $3
ACF_DSC $20
ER_IND $1
STG_NR $1
WT_TON 4
MTOW 4
ACF_SVC_TYP $1
CORP_IND $1
ACF_SPEED 4
COMMUT_IND $1
SSIM_EQP_CD $4
ICAO_EQP_CD $4
CREW_GRP_CD $4
EMP_NR $6
TMST $9
;

input
ACF_OWN
EQP_CD
FLE_CD
ACF_DSC
ER_IND
STG_NR
WT_TON
MTOW
ACF_SVC_TYP
CORP_IND
ACF_SPEED
COMMUT_IND
SSIM_EQP_CD
ICAO_EQP_CD
CREW_GRP_CD
EMP_NR
TMST
```

```
;
data _null_;
set icpr_acf
end=eof;
if eof;
call symput('recs',_n_);
run;

%macro sub;
data out.icpr_acf;
set icpr_acf ;
if _n_ le (&recs-3);
%mend;
%sub;
run;
```

Resulting data set displayed with Viewtable

	AIRCRAFT	AIRCRAFT	AIRCRAFT	AIRCRAFT	AIRCRAFT	AIRCRAFT	AIRCRAFT	AIRCRAFT	AIRCRAFT
1	AF	146	146	BOEING 747-400	N	3			
2	AF	118	118	AFBIS INDOS A200	N	3			
3	AF	120	120	AFBIS INDOS A200	N	3			
4	AF	146	146	AFBIS INDOS A200	N	3			
5	AF	120	120	BOEING 747-400	N	3			
6	AF	120	120	BOEING 747-400	N	3			
7	AF	120	120	BOEING 747-400	N	3			
8	AF	120	120	BOEING 747-400	N	3			
9	AF	120	120	BOEING 747-400	N	3			
10	AF	120	120	BOEING 747-400	N	3			
11	AF	120	120	BOEING 747-400	N	3			
12	AF	120	120	BOEING 747-400	N	3			
13	AF	120	120	BOEING 747-400	N	3			
14	AF	120	120	BOEING 747-400	N	3			
15	AF	120	120	BOEING 747-400	N	3			

This final SAS data set can be processed, summarized, printed, etc. with the SAS tool of your choosing. Additionally, the SAS Export Wizard provides a convenient way to further format this data for quick import into office favorites such as Microsoft Excel.

Conclusion

Previous to Version 7 or Nashville Enablement, without SAS/ACCESS a fair amount of effort was required to convert Oracle tables into SAS data sets. We can now make straight conversions without regard to table or variable name length. There are, however, limitations to the methods outlined in this paper. First, this

program does not take into account table size and associated space requirements. An additional sub-setting where clause could be added to manage this problem. Second, dates are converted as a character string. Further processing or modifying of this procedure, although minimal, would be required to convert the date strings to a SAS date.

References

SAS Language: Reference, Version 6, First Edition 1990, Cary NC: SAS Institute Inc.

Conversion Methods From Oracle Tables to SAS Data Sets Using SAS/ACCESS, SAS Macro Language, and the UNIX Shell Script. Alex Gaber, Long Ngo. Proceedings of the Twenty-Second Annual SAS Users Group International Conference.

Acknowledgments

Gregory Barnett, Delta Technology
Bala Ganeshan, Ph.D., Delta Technology
Nick Salehzadeh, Delta Technology
Sam Loflin, SAS Institute

SAS is a registered trademark of SAS Institute, Incorporated.

The Sabre Group is a registered trademark of The Sabre Group, Incorporated.