

Macro Solution to Oracle Limitation

Diane Cunningham, Southern Company Services, Atlanta, GA

ABSTRACT

Have you ever run SAS® on a UNIX platform and needed pull several thousand accounts from Oracle databases using the SQL pass-thru facility. If you have tried to write these accounts into a macro variable, you probably received an error stating a 1,000 item limit. This paper presents a macro based solution to this Oracle limitation.

INTRODUCTION

In this example, we are connecting to an oracle using the pass-thru facility. We need to pull information regarding several thousand accounts. Each time the program is run, the number of accounts is different. If you have ever tried to create a where clause to use in PROC SQL that contained over 1,000 items, then you've seen the following error:

```
ERROR: ORACLE prepare error: ORA-01795: maximum number of expressions in a list is
1000.
```

TWO POSSIBLE SOLUTIONS

One possible solution is to bring back all the data from the Oracle connection (filtered as much as possible) and merge with the list of accounts in another dataset. If the Oracle data is very large, this method could be very inefficient.

The other solution, and the one described in this paper is to use a macro loop and only bring back data for the accounts of interest

First determine how many accounts are required and set a loop counter.

```
%macro split1000(dsin,dsout);

proc sql noprint;
    select count(*) into: cknum
    from &dsin;
quit;
    %if &cknum > 0 %then %do;

    %do i = 1 %to %sysevalf(&cknum/1000,ceil);
```

The dsin variable represents the dataset that contains the accounts of interest. The dsout variable represents the final output dataset. The PROC SQL used in this manner will create a macro variable, cknum, that contains the total number of records in the dsin dataset. The do loop is then set up to run as many times as necessary to read in all the account numbers, 1,000 at a time. The ceiling function is used to make sure the loop is run one last time for any number of accounts over a multiple of 1,000.

The next step is to subset the dataset that contains the accounts and create a macro variable (balist) containing a comma delimited list of the subset .

```
%let balist = ;
data sublist;
    set &dsin(firstobs=%sysevalf((&i-1)*1000 + 1)
            obs=%sysevalf(1000*&i));
run;
proc sql noprint;
    select ky_ba into :balist separated by ','
    from sublist;
```

```
quit;
```

The next step is to run the PROC SQL connection to the Oracle table(s). Here is an example that pulls monthly data from an Oracle table for a time period of June 2007 – December 2008. In this example, ky_ba is the variable in the Oracle table containing account numbers and balist is the macro variable created in the previous step. We append each subset to the final dataset (dsout).

```
proc sql;
  connect to oracle as mkt (user='user' orapw='xxxxx' path='css1');
  create table temp as
  select * from
  connection to mkt
  (select * from mousage
  where ky_ba in (&balist)
  and ((rev_year = 2007 and rev_mth >= 6) or (rev_year = 2008))
  );
  disconnect from mkt ;
quit;

proc datasets;
  append base=&dsout data=temp;
run;
```

The last step is to close the do loop and end the macro.

```
%end;
%end;
%mend; ** split1000;
```

In the main section of the code, create the list of accounts and call the macro. Then the code can continue with the results of oracle data pull.

```
Data custlist;
  set lib.customers;
run;

%split1000(custlist,oraout);

data next;
  set oraout;
  . . . ;
```

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Diane Cunningham
Southern Company Services
241 Ralph McGill Blvd.
Bin 10206
Atlanta, GA 30308
Work Phone: 404-506-3220
Fax: 404-506-3180
E-mail: dmcunnin@southernco.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.