

Creating an Intranet Toolbox of Selective Oracle Metadata

John Charles Gober, Deborah Mullen, Jana Smith

Bureau of the Census, Demographic Surveys Methodology Division, Washington, DC

INTRODUCTION

The purpose of this paper is to demonstrate an alternative method of allowing users to view the attributes of their Oracle® Databases. This application is useful for programmers and developers who do not have access to Oracle Designer or other similar applications which allow visual representations of your databases.

Even though this application is fairly simple in its appearance, it does give the user critical information about their database tables, variables, and indexes. By being able to cross reference tables with variables, variables with indexes, and indexes with tables this in turn will allow the rapid development of programs and applications which need to use these tables.

Because of the length of the code involved with this application it will all appear at the end of this paper.

ACCESSING THE ORACLE METADATA

Oracle, like SAS, keeps a wealth of overhead information stored in its administrative tables. These meta tables are analogous to the ones found in the automatic SASHELP library. For our intranet toolbox application we needed only seven variables from five of these tables. These variables were table_name, column_name, constraint_type, data_type, data_length, index_name, and column_position. The table names can be viewed in the inserted code.

There were two methods that immediately came to mind that would allowed us to extract the information needed from these variables. The first was to use a data step with the tables referenced through a LIBNAME statement. The LIBNAME in turn would be associated with the Oracle access engine. The second method would be to use PROC SQL. Even though the second method was the least familiar with our users this was the one that was chosen. The primary reason being that much of our work is done interactively through the display manager and we have a bad habit of leaving our session

open all day long and many times into the night. Forgetting to disassociate a LIBNAME can cause grief among the database managers when it comes time to do routine maintenance. PROC SQL automatically breaks the connection to the database at the termination of the procedure thus eliminating future conflict. When using a LIBNAME we must remember to also CLEAR when we are done with our query.

The first set of SQL code in our application (Section One) was developed to extract just the names of our database tables, the variable names, and their attributes. This gave us all of the information for our initial set of pages. At this point we could have just executed an ODS HTML statement, run the data through a PROC REPORT, made a few cosmetic changes, and finally uploaded it to our server. A user could now click on a database table and view the variables and some variable attributes within the table but could do no more.

In order to cross reference the variables back to their respective tables and tables with their indexes we needed a few more steps. A second PROC SQL was needed to extract the index names, indexed variables and their associated tables (Section Two).

COMBINING THE INFORMATION

The next step (Section Three) was to eliminate tables in the database that the user would never look at and to count the number of tables left. These are the tables that the database administrator would normally use but not us. Tables that contain file sizes, extents, user and system information, etc. No need to clutter the screens with extraneous information. This is also the section where we linked the index information with the table and variable information. This was also where we totaled and sequenced the tables and variables and to make sure that all fields were populated in order to insure reliable links for our html pages.

CREATING THE HYPERLINKS

When SAS designed ODS HTML they opened up a whole new avenue of expression for the average SAS programmer. Without the need for extensive training in html a user with very little experience can create their own web pages complete with frames, indexes, table of contents, and bodies. And in combination with PROC TEMPLATE and other procedures these web pages can be customized with very little effort. However all good things need help. ODS only creates one way links from the table of contents frame to the body frames. If you wish to create links inside the body frames referencing other body frames then you must create your own hyperlinks to be inserted into your procedures to be used by ODS.

The code to create the hyperlinks (Section Four) looks slightly intimidating but it is only a series of datasets that, when used in combination with PROC FORMAT, create tables of sequential page names or be use as targets for all of our desired links. The need for this process is best understood if viewed in context with the PROC REPORT code while looking at the actual pages that have been generated.

PROC TEMPLATE

With each licensed copy of SAS there are supplied several basic templates that give the user the ability to choose a predefined look and feel to the html pages and contents. These templates give the user a predefined starting point when it comes to changing the look, feel, and behavior of their pages. To view the code for these templates the user needs only to go to the Results Window, click on View, Templates, sashelp.tmplmst. This application used the DEFAULT template with only a few minor changes.

Not being totally familiar with PROC TEMPLATE (Section Five) this was the most difficult aspect of the application. However with a little research and a few calls to our help desk it was finally discovered that only minor changes to the procedure were needed and SAS has done an excellent job of making the editing of templates quite easy. Even though the basic changes worked, the appearance of the titles, the fonts, the program names in the table of contents, and the background colors needed to be changed.

PROC REPORT

There was only one reason that PROC REPORT was selected for the visual representation for our pages. This was the procedure used in the example from the **SAS Guide to the Output Delivery System**[®] and the **PROC TEMPLATE FAQ** on the SAS Technical Support web pages.

This is where all of our efforts came together (Section Six). Titles and footnotes were added to our pages, hyperlinks applied, and columns justified. Even a link back to our database main menu selection screen was create so the user could view other databases they had permission to use.

CONCLUSION

Data is of no use unless you know what it contains. By creating this small application from examples readily available in the SAS community you can create an easy to use interactive table/variable cross reference table that can be used to visually access not only your Oracle databases but also your SAS libraries and other data structures.

CREDITS AND ACKNOWLEDGEMENTS

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.

Oracle is a registered trademark, and ConText, Oracle Alliance and Oracle8 are trademarks or registered trademarks of Oracle Corporation.

CONTACT INFORMATION

Questions and comments concerning this paper can be directed to:

john.charles.gober@census.gov
deborah.m.mullen@census.gov
jana.l.smith@census.gov
Bureau of the Census
FOB3 MS8700
Washington DC 20233-8700

*CREATING AN INTRANET METATABLE;
 *Code has been intentionally reformatted to fit the page;

```
libname wk '/work';

*SECTION ONE;
proc sql;
connect to oracle(user=xx password=xx
schema=xx path='mydatabase');
create table wk.dbtavbar as
select * from connection to oracle
(select distinct acc.table_name,
acc.column_name, ac.constraint_type,
atc.data_type, atc.data_length from
    all_constraints ac,
    all_cons_columns acc,
    all_tab_columns atc,
    dba_tables dbt
where ac.constraint_name =
acc.constraint_name and
    atc.column_name =
acc.column_name and
    atc.table_name =
acc.table_name and
    atc.table_name =
dbt.table_name);
run;

*SECTION TWO;
proc sql;
connect to oracle(user=xx password=xx
schema=xx path='mydatabase');
create table wk.allindexes as
select * from connection to oracle
(select distinct ind.index_name,
ind.table_name, ind.column_name,
ind.column_position
from all_ind_columns ind);
run;

*SECTION THREE;
data wk.sampts;
set wk.dbtavbar(where=(not
indexc(table_name,'$', '#')));
by table_name column_name;
if constraint_type not in ('P', 'R')
then constraint_type = ' ';
if last.column_name then output;
run;

proc sort data=wk.sampts out=wk.sampts;
by table_name ; run;

proc sort data=wk.allindexes
out=wk.allindexes;
by table_name column_name;
where not
(indexc(table_name,'$', '#')); run;

data wk.sampts(drop=_idx);
retain _idx 0;
merge wk.sampts(in=in1)
wk.allindexes(in=in2
keep=index_name table_name);
by table_name ;
_idx + 1;
if in1;
if index_name = ' ' then
index_name='noindex' || _idx;
run;
```

```
proc sort data=wk.allindexes
out=wk.allindexes;
by table_name index_name
column_position column_name; run;

proc sort data=wk.sampts out=wk.samptsa;
by table_name column_name; run;

data wk.samptsa;
set wk.samptsa end=last;
by table_name column_name;
table_name2 = table_name;
if first.table_name then tottables + 1;
if last then call
symput('tottables',tottables);
if first.column_name; run;

proc sort data=wk.sampts(keep=column_name
table_name index_name) out=wk.samptsb
nodupkey;
by column_name table_name; run;

*SECTION FOUR;
data makefmt;
retain counter &tottables;
set wk.samptsb end=last;
by column_name;
if first.column_name then
do;
fmtname = '$varfmt';
start = column_name;
label = 'samptsbody' || put(counter,4.);
counter + 1;
output;
end;
if last then call
symput('startindx',counter);
run;

proc format cntlin=makefmt; run;

data makefmt2;
retain counter 0; length label $45;
set wk.samptsa;
by table_name;
if first.table_name then
do;
fmtname = '$dbfmt';
start = table_name;
if counter=0 then label='samptsbody';
else
label='samptsbody' || put(counter,4.);
counter + 1;
output;
end;
run;

proc format cntlin=makefmt2; run;

data makefmt2b;
retain counter (&startindx);
length label $100 start $50;
set wk.samptsa end=last;
by table_name;
if first.table_name then
do;
fmtname = '$db2fmt';
start = table_name;
label = '<A
HREF="http://mydir/myfiles/samptsbody'
```

```

||trim(left(counter))|| '.htm">indexes
</A>';
  if substr(index_name,1,7) ne 'noindex'
  then
  do;
    counter + 1;
    output;
  end;
end;
if first.table_name and
substr(index_name,1,7) = 'noindex' then
do;
  fmtname = '$db2fmt';
  start = table_name;
  label = ' ';
  output;
end;
run;

proc format cntlin=makefmt2b; run;

proc sort data=wk.samptsa out=makefmt3
nodupkey ;
by table_name index_name; run;

data makefmt3;
  retain counter &startindx;
  length label $15 start $50end $50;
  set makefmt3 end=last;
  by table_name index_name;
  fmtname = '$idxfmt';
  start = index_name;
  end = index_name;
  label = 'samptsbody' ||put(counter,4.);
  if substr(index_name,1,7) = 'noindex'
  and substr(lag(index_name),1,7) ne
  'noindex' then counter = counter - 1;
  if substr(index_name,1,7) ne 'noindex'
  then output;
  if last.table_name then
  do;
    counter + 1;
  end;
end;
run;

proc format cntlin=makefmt3; run;

proc sort
data=wk.samptsa(keep=index_name)
out=makefmt4 nodupkey ;
  by index_name;
  where substr(index_name,1,7) ne
'noindex';
run;

data makefmt4;
  length label $15;
  length start $50;
  length end $50;
  set makefmt4 end=last;
  fmtname = '$keyfmt';
  start = index_name;
  end = index_name;
  label = 'YES';
  output;
  if last then
  do;
    start = 'other';
    end = 'other';
    label = ' ';
    output;
  end;
end;

proc format cntlin=makefmt4; run;

proc sort data=wk.sampts out=makefmt5
nodupkey;
  by index_name; run;

proc sort data=makefmt5;
  by table_name; run;

data makefmt5b;
  length start $50;
  length label $20;
  retain counter 0;
  set makefmt5 end=last;
  by table_name;
  fmtname = '$stabfmt';
  start = index_name;
  if counter = 0 then label =
'samptsbody';
  else
  label =
'samptsbody' ||put(counter,4.);
  output;
  if last.table_name then
  do;
    * if substr(index_name,1,7) ne
'noindex' then counter + 1;
    counter + 1;
  end;
end;
run;

proc format cntlin=makefmt5b; run;

*SECTION FIVE;
proc template;
  edit styles.default as styles.test;
  style contentfolder / LISTENTRYANCHOR =
ON;
  style color_list / 'BGA1' =
CXE5C76B;
  style colors / 'CONTENTBG' =
COLOR_LIST('BGA1');
  style colors / 'DOCBG' =
COLOR_LIST('BGA1');
  style confolderfg / background =
colors('contentbg');
  style text / 'CONTENT TITLE'
= 'DSMD INTRANET TOOLBOX';
style table / cellpadding=0
cellspacing=0;
END;
RUN;

ods listing close;
ods noresults;
ods html file='samptsbody.htm'
style=styles.test
contents='samptscontents.htm'
frame='samptsframe.htm'
path='/mydir/myfiles' (url=None)
newfile=page;

options nobyline;

*SECTION SIX;
title1 "MY Database";
title2 "Column Names for Table";

```

```

title3 '#byval(table_name)';

footnote1 'Developed and Shared by Your
Friendly Staff at CENSUS';
ods proclabel 'MY Database';

proc report data=wk.samptsa
            nowd contents=''
style(column)=[font_face=Courier
font_size=2 asis=on]
style(lines)={protectspecialchars=off};
  by table_name ;
  column /*index_name*/ constraint_type
column_name data_type data_length
table_name2;
  define table_name2 /noprint;
  define column_name / "Column" display
format=$30. center;
  define constraint_type / "Key Type"
display format=$30. center;
  define data_type / "Format" display
format=$12. center;
  define data_length / "Length" display
format=5. center;
  compute before _page_ / left;
  line ' ';
  line ' ';
  line table_name2 $db2fmt. ;
  line ' ';
endcomp;
  compute after _page_ / left;
  line ' ';
  line ' ';
endcomp;
  compute column_name;
  call
define(_col_,"URLBP",compress(trim(put(co
lumn_name,$varfmt.)))||'.htm') ;
  endcomp;
  compute index_name;
  call
define(_col_,"URLBP",compress(trim(put(in
dex_name,$idxfmt.)))||'.htm') ;
  endcomp;
  label table_name = 'TABLE';
run;

ods proclabel 'Variables elsampts';
title1 "Tables Containing the Variable";
title2 '#byval(column_name)';
footnote1 'Developed and Shared by Your
Friendly Staff at CENSUS';

proc report data=wk.samptsb
            (keep=table_name
column_name) nowd contents=''
style(column)=[font_face=Courier
font_size=2 asis=on];
  by column_name ;
  column table_name;
  define table_name /'Tables' left;
  compute table_name;

```

```

            call
define(_col_,"URLBP",compress(trim(put(ta
ble_name,$dbfmt.)))||'.htm');
            endcomp;
            label column_name = 'VAR';
run;

ods proclabel 'Indexes elsampts';
title1 "Indexes for Table";
title2 '#byval(table_name)';
footnote1 'Developed and Shared by Your
Friendly Staff at CENSUS';
proc report data=wk.allindexes
            (keep=index_name table_name
column_name column_position) nowd
contents=''
style(column)=[font_face=Courier
font_size=2 asis=on];
  by table_name ;
  column index_name column_position
column_name ;
  define index_name / group 'Index Name'
left;
  define column_position / 'Key Order'
left;
  define column_name /'Column ' left;
  compute index_name;
  call
define(_col_,"URLBP",compress(trim(put(in
dex_name,$stabfmt.)))||'.htm') ;
  endcomp;
run;

options byline;
title2;
ods html close; ods listing; ods results;

filename cont '/mywebpages/
mydatabase/samptscontents.htm' lrecl=255;
filename temp1 temp lrecl=255;

data _null_;
  infile cont pad missover;
  input @001 coll_5 $char5.
        @001 coll_255 $char255.;
  file temp1;
  put coll_255;
  if coll_5 = '<BODY' then
do;
  put '<a target=_parent
href="http://mywebpages/DB_MAIN_PAGE.html
">MAIN MENU</a>';
  put '<br>';
  end;
run;

data _null_;
  infile temp1 pad missover;
  input @001 coll_255 $char255.;
  file cont;
  put coll_255;
run;

```

Figure 1 - Click on Table= to bring up the column listing.

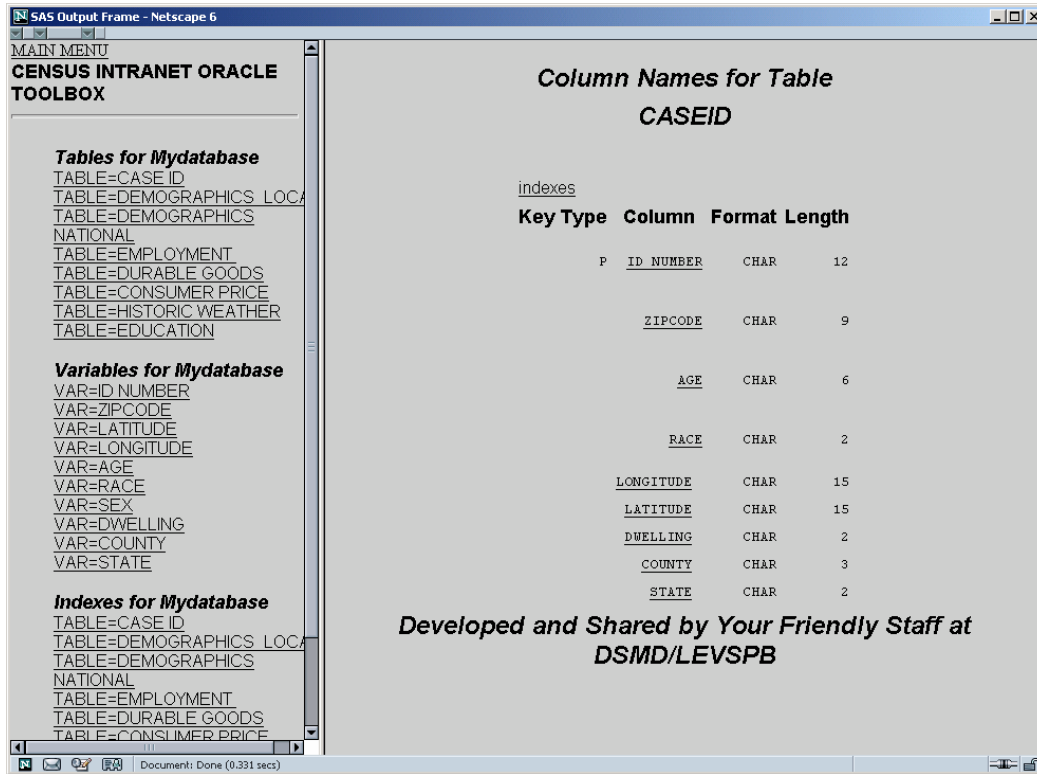


Figure 2 - Click on 'indexes' to bring up the index listing. Then click on the 'Index Name' to return to the Column Listing. Not shown is the variables by database cross reference page.

