RAMPANT TECHPRESS

Documenting Oracle Databases

Complete Oracle database schema auditing

Mike Ault

Retail Price $12.95 US/$19.95 Canada
ISBN: 0-9740716-6-8

RAMPANT TECHPRESS
eBook

Oracle eBook

Rampant TechPress

Documenting Oracle Databases

Complete Oracle database schema auditing

Mike Ault

Notice

While the author makes every effort to ensure the information presented in this white paper is accurate and without error, Rampant TechPress, its authors and its affiliates takes no responsibility for the use of the information, tips, techniques or technologies contained in this white paper. The user of this white paper is solely responsible for the consequences of the utilization of the information, tips, techniques or technologies reported herein.

Documenting Oracle Databases

Complete Oracle database schema auditing

By Mike Ault

Published by Rampant TechPress, Kittrell, North Carolina, USA

Series Editor: Don Burleson

Production Editor: Teri Wade

Cover Design: Bryan Hoff

Visit **www.rampant.cc** for information on other *Oracle In-Focus* books.

ISBN: 0-9740716-6-8

Table Of Contents

Introduction

Many times the Oracle DBA is hired once the database becomes to much to handle without one, inherits the position or is appointed to it "since they are so good with computers". In any case, they usually get an undocumented system that follows no conventions and holds many traps for the unwary DBA.

In other situations the DBA may find they have the task of recreating the database for which they are responsible. If export and import can be used, this task is fairly easy to accomplish, however, sometimes the DBA will be required to provide DDL (Data Definition Language) for this purpose. The DBA may also wish to document existing procedures, views, constraints and such other structures as they see fit, with human readable output and a minimum of re-editing.

In all of the above cases, the Oracle provided methods fall woefully short of the mark in providing the DBA with documentation. It falls on the DBA's shoulders to develop SQL, PL/SQL and SQLPLUS code to delve into the inner workings of the Oracle Data Dictionary tables and re-generate the required DDL.

This presentation will demonstrate techniques to use the Oracle instance to document itself.

The Oracle Data Dictionary, an Overview

The heart of the whole matter is a collection of C constructs, Oracle tables and Oracle views that are collectively called the Oracle Data Dictionary. At the lowest level are the "hidden" C structures known as the X$ tables. These X$ tables are usually best left alone. Indeed, to even see the contents a DBA has to jump through a few hoops and once they do get to

them, they are not well documented and have such logical attributes as "mglwmp" or "tabsrpr". Needless to say, unless the DBA has several long nights available and a penchant for solving riddles it is suggest that they leave these to Oracle.

The next layer of the Oracle Data Dictionary is the $ tables. These are more human readable cuts of the X$ tables and have such names as COL$ or TAB$. While being a step above the X$ structures, it is still suggested that the DBA only use them when it is really needed. Some of the reports discussed in this paper will use these tables.

The third layer of the dictionary is the V$ views and their cousins the V_$ tables (actually, Siamese twins since for nearly all of the V$ views there is a corresponding V_$ view). These are the workhorses of the Data Dictionary and what most of the scripts in this paper deal with.

The final layer of the dictionary (for our purposes) is the DBA_ series of views. These views are made from the V$ and $ sets of views and tables and provide a very friendly interface for viewing the insides of your Oracle Data Dictionary. The USER_ and ALL_ views are based on the DBA_ views. Some of the user specific scripts, such as for constraints, uses the USER_ subset of views, however, the script can be made more general by adding reference to the "OWNER" attribute and changing USER_ to DBA_ if the DBA desires.

Documenting or Rebuilding?

This paper will use the terms documenting and rebuilding interchangeably. A good set of build scripts will provide excellent documentation for an instance. Essentially there are two types of documentation, "hard" documentation such as DDL that an experienced database developer can understand, and "soft" documentation such as generalized reports showing structures and relations that anyone with a smattering of database experience can use. The paper will deal more with the "hard" type of documentation, that is, the scripts to rebuild the database, where

applicable, reports that provide "soft" documentation will be mentioned, and included with the paper.

The Database

Databases contain the following items:

Hard Objects:

- Tablespaces and their associated Datafiles
- Control Files
- Redo Logs
- Rollback Segments
- Tables
- Indexes
- Database Initialization file
- Clusters

Stored Objects:

- View definitions
- Constraints
- Procedures
- Functions
- Packages
- Package Bodies
- Triggers
- User definitions
- Roles

- Grants
- Database Links
- Snapshots and Snapshot Logs

In the following pages this paper will cover how to generate the required DDL to recreate or document each of the above database objects using SQL, PLSQL and SQLPLUS. This paper will also discuss creation of scripts to allow dropping a set of objects (such as constraints) and recreating them.

The Control File

To begin at the beginning, we should first document the Control file since this is the only location where the information for MAXDATAFILES, MAXLOGFILES MAXMEMBERS and other instance specific data can be found. This information will be added by hand to the Database rebuild script covered later in the paper.

Having a script around that rebuilds the control file is a very good idea. In the case where the Oracle instance was built by people with very little understanding of how Oracle builds default instances (poorly) you will usually get a very dysfunctional database from a vanilla build. The DBA will find their MAXDATAFILES will be set to 20 or maybe 32 and that other hard database limits are probably set too low as well. Once the DBA generates a script to rebuild their control files, they can change these hard database limits without rebuilding the database by simply rebuilding the control files. In other situations, a disk crash can wipe out the control file (hopefully there is more than one, but maybe a DBA didn't create the database?) and leave the database unable to even startup (actually there is a way, but it is beyond the scope of this paper.) With a rebuild script on hand, it makes these situations easier to handle.

A DBA can always backup the control file with the command:

```
ALTER DATABASE BACKUP CONTROLFILE to 'filename' REUSE;
```

However, the above command only makes a machine readable copy, a good idea for when changes are made to structures, A DBA should usually backup the control file with each set of backups. Of course, this doesn't provide documentation of the control file.

Since there are no tables which document the control files (other than v$parameter which only documents their location) the DBA must depend on a system command to provide them with the require script:

```
ALTER DATABASE
       BACKUP CONTROLFILE TO TRACE NORESTLOGS;
```

The output of this script looks like so:

```
Dump file H:\ORAWIN\RDBMS71\trace\ORA14071.TRC
Sat Mar 30 10:05:53 1996
ORACLE V7.1.4.1.0 - Production.
vsnsta=0
vsnsql=a vsnxtr=3
MS-WINDOWS Version 3.10
Sat Mar 30 10:05:52 1996
Sat Mar 30 10:05:53 1996

*** SESSION ID:(5.3)
# The following commands will create a new control file and use it
# to open the database.
# No data other than log history will be lost. Additional logs may
# be required for media recovery of offline data files. Use this
# only if the current version of all online logs are available.
STARTUP NOMOUNT
CREATE CONTROLFILE REUSE DATABASE "ORACLE" NORESETLOGS NOARCHIVELOG
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 32
    MAXINSTANCES 16
    MAXLOGHISTORY 1600
LOGFILE
  GROUP 1 'H:\ORAWIN\DBS\wdblog1.ora'  SIZE 500K,
  GROUP 2 'H:\ORAWIN\DBS\wdblog2.ora'  SIZE 500K
DATAFILE
  'H:\ORAWIN\DBS\wdbsys.ora' SIZE 10M,
  'H:\ORAWIN\DBS\wdbuser.ora' SIZE 3M,
  'H:\ORAWIN\DBS\wdbrbs.ora' SIZE 3M,
  'H:\ORAWIN\DBS\wdbtemp.ora' SIZE 2M
;

# Recovery is required if any of the datafiles are restored backups,
# or if the last shutdown was not normal or immediate.
RECOVER DATABASE
```

```
# Database can now be opened normally.
ALTER DATABASE OPEN;
```

As can be seen from a glance at the above script, all the required "hard" database setpoints are now documented. The output from the command is located wherever the system places its system level trace files (check v$parameter on a wildcard name search for '%dump%').

Documenting the Database Initialization file

The initialization file, commonly called INIT.ORA or initSID.ora where the SID is the database instance name, is another key file for the database documentation set. Sadly, the initialization file is often overlooked, in fact many "DBAs" don't even know how to find it. All of the values used in the initialization process are stored in the V$PARAMETER table. The following script will provide a complete INIT.ORA
file for a system:

```
REM
REM NAME               : init_ora_rct.sql
REM FUNCTION           : Recreate the instance init.ora file
REM USE                      : GENERAL
REM Limitations        : None
REM
SET NEWPAGE 0 VERIFY OFF
SET ECHO OFF feedback off termout off PAGES 300 lines 80 heading off
column name  format a41
column value format a37
column dbname new_value db noprint
select value dbname from v$parameter where name = 'db_name';
DEFINE OUTPUT = 'rep_out\&db\init.ora'
SPOOL &OUTPUT
SELECT '# Init.ora file from v$parameter' name, null value from dual
union
select '# generated on:'||sysdate name, null value from dual
union
select '# script by MRA 11/7/95 TRECOM' name, null value from dual
union
select '#' name, null value from dual
UNION
SELECT name||' =' name,value FROM V$PARAMETER
WHERE value is not null;
SPOOL OFF
CLEAR COLUMNS
```

```
set termout on
pause Press enter to continue
exit
```

An example output from this report follows:

```
#
# Init.ora file from v$parameter
# generated on:06-MAR-96
# script by MRA 11/7/95 TRECOM
audit_file_dest        =                            ?/rdbms/audit
audit_trail    =                                  NONE
background_core_dump   =                         full
background_dump_dest   =        /bto/sys/oracle/product/7.2.2/rdbms/log
blank_trimming =                                FALSE
cache_size_threshold   =                          20
ccf_io_size    =                                  65536
checkpoint_process     =                          FALSE
cleanup_rollback_entries      =                    20
close_cached_open_cursors     =                   FALSE
commit_point_strength  =                        1
compatible     =                                   7.1.0.0
control_files  =
       /vol2/oracle1/ORCSPCD1/ctrl1ORCSPCD101.ctl,/vol2/oracle2/ORCSPCD1/c
trl2ORCSPCD102.ctl,/vol3/oracle3/ORCSPCD1/ctrl3ORCSPCD103.ctl
core_dump_dest =                     /bto/sys/oracle/product/7.2.2/dbs
cpu_count      =                                    1
cursor_space_for_time  =                       FALSE
.
. (the actual output is 3 pages long, so this is the short version)
.
thread  =                                       0
timed_statistics       =                             FALSE
transactions   =                                   66
transactions_per_rollback_segment     =           16
use_readv      =                                    FALSE
user_dump_dest =              /bto/sys/oracle/product/7.2.2/rdbms/log
```

The DBA will have to edit the output to add quotes and parenthesis, but this is still better than rebuilding the file by hand.

Most DBAs will see a number of parameters they weren't aware existed in their listing from this script. In fact, many of the parameters may not be documented. This script lists all of the parameters available on your platform and uses their default value if the DBA hasn't supplied a value via the initialization file.

The Database Itself

Now that the control file settings and the initialization file settings have been documented, it is time to document the database creation script itself. The following script will generate a "bare bones" CREATE DATABASE command. The values from the CREATE CONTROLFILE script for the setting of MAX parameters generated prior to this should be edited into the command script.

```
REM FUNCTION: SCRIPT FOR CREATING DB
REM
REM           This script must be run by a user with the DBA role.
REM
REM           This script is intended to run with Oracle7.
REM
REM           Running this script will in turn create a script to
REM           rebuild the database.  This created
REM           script, crt_db.sql,  is run by SQLDBA
REM
REM           Only preliminary testing of this script was performed.
REM           Be sure to test it completely before relying on it.
REM
REM M. Ault 3/29/96 TRECOM
REM

set verify off;
set termout off;
set feedback off;
set echo off;
set pagesize 0;

set termout on
prompt Creating db build script...
set termout off;
create table db_temp
     (lineno NUMBER,  text VARCHAR2(255))
/


DECLARE
   CURSOR dbf_cursor IS
        select  file_name,bytes
        from  dba_data_files
        where tablespace_name='SYSTEM';
   CURSOR mem_cursor (grp_num number) IS
        select   member
        from     v$logfile
        where    group#=grp_num
        order by member;
    CURSOR thread_cursor IS
```

```
        select thread#, group#
        from v$log
        order by thread#;

   grp_member          v$logfile.member%TYPE;
   db_name             varchar2(8);
   db_string             VARCHAR2(255);
   db_lineno             number := 0;
   thrd                number;
   grp                 number;
   filename            dba_data_files.file_name%TYPE;
   sz                  number;
   begin_count         number;

   procedure write_out(p_line INTEGER,
                       p_string VARCHAR2) is
   begin
      insert into db_temp (lineno,text)
             values (db_lineno,db_string);
   end;

BEGIN
        db_lineno:=db_lineno+1;
   SELECT 'CREATE DATABASE '||value into db_string
        FROM v$parameter where name='db_name';
     write_out(db_lineno,db_string);
        db_lineno:=db_lineno+1;
   SELECT 'CONTROLFILE REUSE' into db_string
        FROM dual;
     write_out(db_lineno,db_string);
        db_lineno:=db_lineno+1;
   SELECT 'LOGFILE (' into db_string
        FROM dual;
     write_out(db_lineno,db_string);
commit;
if thread_cursor%ISOPEN then
        close thread_cursor;
        open thread_cursor;
else
        open thread_cursor;
end if;
loop
        fetch thread_cursor into thrd,grp;
        exit when thread_cursor%NOTFOUND;
        db_lineno:=db_lineno+1;
        db_string:= 'THREAD '||thrd||' GROUP '||grp||' (';
        write_out(db_lineno,db_string);
        if mem_cursor%ISOPEN then
                close mem_cursor;
                open mem_cursor(grp);
        else
                OPEN mem_cursor(grp);
        end if;
        db_lineno:=db_lineno+1;
        begin_count:=db_lineno;
        loop
                fetch mem_cursor into grp_member;
                exit when mem_cursor%NOTFOUND;
```

```
                if begin_count=db_lineno then
                        db_string:=''''||grp_member||'''';
                        write_out(db_lineno,db_string);
                        db_lineno:=db_lineno+1;
                else
                        db_string:=','||''''||grp_member||'''';
                        write_out(db_lineno,db_string);
                        db_lineno:=db_lineno+1;
                end if;
        end loop;
        db_lineno:=db_lineno+1;
        db_string:=' )';
        write_out(db_lineno,db_string);
end loop;
db_lineno:=db_lineno+1;
SELECT ')' into db_string from dual;
write_out(db_lineno,db_string);
commit;
if dbf_cursor%ISOPEN then
        close dbf_cursor;
        open dbf_cursor;
else
        open dbf_cursor;
end if;
begin_count:=db_lineno;
loop
        fetch dbf_cursor into filename, sz;
        exit when dbf_cursor%NOTFOUND;
        if begin_count=db_lineno then
                db_string:='DATAFILE '||''''||filename||''''||' SIZE
'||sz||' REUSE';
        else
                db_string:=','||''''||filename||''''||' SIZE '||sz||'
REUSE';
        end if;
        db_lineno:=db_lineno+1;
        write_out(db_lineno,db_string);
end loop;
commit;
SELECT decode(value,'TRUE','ARCHIVELOG','FALSE','NOARCHIVELOG')
        into db_string from v$parameter where name='log_archive_start';
        db_lineno:=db_lineno+1;
        write_out(db_lineno,db_string);
SELECT ';' into db_string from  dual;
        db_lineno:=db_lineno+1;
        write_out(db_lineno,db_string);
CLOSE dbf_cursor;
CLOSE mem_cursor;
CLOSE thread_cursor;
commit;
END;
/
column dbname new_value db noprint
select value dbname from v$parameter where name='db_name';
set heading off pages 0 verify off
set recsep off
spool rep_out\&db\crt_db.sql
col text format a80 word_wrap
```

```
select   text
from     db_temp
order by  lineno;
spool off
set feedback on verify on termout on
drop table db_temp;
prompt Press enter to exit
exit
```

The reason this script is so long is that there may be multiple datafiles for the SYSTEM tablespace and there most certainly will be multiple redo logs, redo log groups and possibly redo log threads. This multiplicity of files results in the need for a cursor for each of the possible recursions and a loop-end loop construct to support the selection of the data from the database.

It is ironic that all of the above code is used to produce the following output:

```
CREATE DATABASE ORCSPCD1
CONTROLFILE REUSE
LOGFILE (
THREAD 1 GROUP 1 (
'/vol2/oracle1/ORCSPCD1/log1ORCSPCD1.dbf'
)
THREAD 1 GROUP 2 (
'/vol2/oracle2/ORCSPCD1/log2ORCSPCD1.dbf'
)
THREAD 1 GROUP 3 (
'/vol3/oracle3/ORCSPCD1/log3ORCSPCD1.dbf'
)
)
DATAFILE '/vol2/oracle1/ORCSPCD1/systORCSPCD1.dbf' SIZE 41943040 REUSE
NOARCHIVELOG
;
```

The generated script would be longer if there were multiple datafiles in the SYSTEM tablespace and if mirrored redo logs where in use. As was said above, the script should have the MAX set of parameters from the CREATE CONTROLFILE command edited into it between the DATAFILE and NOARCHIVELOG (or ARCHIVELOG) clauses.

Soft documentation for database related items would consist of reports on the redo logs, system parameters, and, datafiles for the SYSTEM tablespace. The scripts redo_log.sql, db_thrd.sql, db_parm.sql and

datafile.sql provide soft documentation reports of these items. The soft documentation scripts are in appendix A.

Documenting Tablespaces

Tablespaces and their underlying datafiles should be created immediately after the database itself. Therefore, we will cover the generation of the tablespace DDL next. The following script will create the DDL required to make an exact duplicate of a systems existing tablespace/datafile profile. The DBA may wish to edit the resulting script to combine multiple datafiles into a single large datafile per tablespace or, the inverse, spread a large datafile across several physical platters for improved performance. The script follows:

```
REM
REM  FUNCTION: SCRIPT FOR CREATING TABLESPACES
REM
REM  FUNCTION: This script must be run by a user with the DBA role.
REM
REM This script is intended to run with Oracle7.
REM
REM FUNCTION: Running this script will in turn create a script to build
REM FUNCTION: all the tablespaces in the database.  This created script,
REM FUNCTION: crt_tbls.sql, can be run by any user with the DBA role
REM FUNCTION: or with the 'CREATE TABLESPACE' system privilege.
REM
REM Only preliminary testing of this script was performed.  Be sure to
test
REM it completely before relying on it.
REM

set verify off;
set termout off;
set feedback off;
set echo off;
set pagesize 0;

set termout on;
select 'Creating tablespace build script...' from dual;
set termout off;

create table ts_temp (lineno number, ts_name varchar2(30),
                      text varchar2(800))
/
DECLARE
   CURSOR ts_cursor IS select   tablespace_name,
                                initial_extent,
                                next_extent,
```

```
                                min_extents,
                                max_extents,
                                pct_increase,
                                status
                        from    sys.dba_tablespaces
                        where tablespace_name != 'SYSTEM'
                        and status != 'INVALID'
                        order by tablespace_name;
   CURSOR df_cursor (c_ts VARCHAR2) IS
       select   file_name,
                bytes
      from      sys.dba_data_files
      where     tablespace_name = c_ts
                and tablespace_name != 'SYSTEM'
      order by file_name;
   lv_tablespace_name   sys.dba_tablespaces.tablespace_name%TYPE;
   lv_initial_extent    sys.dba_tablespaces.initial_extent%TYPE;
   lv_next_extent       sys.dba_tablespaces.next_extent%TYPE;
   lv_min_extents       sys.dba_tablespaces.min_extents%TYPE;
   lv_max_extents       sys.dba_tablespaces.max_extents%TYPE;
   lv_pct_increase      sys.dba_tablespaces.pct_increase%TYPE;
   lv_status            sys.dba_tablespaces.status%TYPE;
   lv_file_name         sys.dba_data_files.file_name%TYPE;
   lv_bytes             sys.dba_data_files.bytes%TYPE;
   lv_first_rec         BOOLEAN;
   lv_string            VARCHAR2(800);
   lv_lineno            number := 0;

   procedure write_out(p_line INTEGER, p_name VARCHAR2,
             p_string VARCHAR2) is
   begin
     insert into ts_temp (lineno, ts_name, text) values
            (p_line, p_name, p_string);
   end;

BEGIN
   OPEN ts_cursor;
   LOOP
      FETCH ts_cursor INTO lv_tablespace_name,
                           lv_initial_extent,
                           lv_next_extent,
                           lv_min_extents,
                           lv_max_extents,
                           lv_pct_increase,
                           lv_status;
      EXIT WHEN ts_cursor%NOTFOUND;
      lv_lineno := 1;
      lv_string := ('CREATE TABLESPACE '||lower(lv_tablespace_name));
      lv_first_rec := TRUE;
      write_out(lv_lineno, lv_tablespace_name, lv_string);
      OPEN df_cursor(lv_tablespace_name);
      LOOP
         FETCH df_cursor INTO lv_file_name,
                              lv_bytes;
         EXIT WHEN df_cursor%NOTFOUND;
         if (lv_first_rec) then
            lv_first_rec := FALSE;
            lv_string := 'DATAFILE ';
```

```
         else
            lv_string := lv_string || ',';
         end if;
         lv_string:=lv_string||''''||lv_file_name||''''||
                    ' SIZE '||to_char(lv_bytes) || ' REUSE';
      END LOOP;
      CLOSE df_cursor;
         lv_lineno := lv_lineno + 1;
         write_out(lv_lineno, lv_tablespace_name, lv_string);
         lv_lineno := lv_lineno + 1;
         lv_string := (' DEFAULT STORAGE (INITIAL ' ||
                      to_char(lv_initial_extent) ||
                      ' NEXT ' || lv_next_extent);
         write_out(lv_lineno, lv_tablespace_name, lv_string);
         lv_lineno := lv_lineno + 1;
         lv_string := (' MINEXTENTS ' ||
                      lv_min_extents ||
                      ' MAXEXTENTS ' || lv_max_extents);
         write_out(lv_lineno, lv_tablespace_name, lv_string);
         lv_lineno := lv_lineno + 1;
         lv_string := (' PCTINCREASE ' ||
                      lv_pct_increase || ')');
         write_out(lv_lineno, lv_tablespace_name, lv_string);
         lv_string := ('   '||lv_status);
         write_out(lv_lineno, lv_tablespace_name, lv_string);
         lv_lineno := lv_lineno + 1;
         lv_string:='/';
         write_out(lv_lineno, lv_tablespace_name, lv_string);
         lv_lineno := lv_lineno + 1;
         lv_string:='                                                    ';
         write_out(lv_lineno, lv_tablespace_name, lv_string);
   END LOOP;
   CLOSE ts_cursor;
END;
/

spool rep_out\cre_tbsp.sql
set heading off
set recsep off
col text format a80 word_wrap
select   text
from     ts_temp
order by ts_name, lineno;
spool off;
drop table ts_temp;
exit
```

Example output from this script follows:

```
CREATE TABLESPACE users
DATAFILE '/oracle1/ORCSPCD1/data/users01.dbf' SIZE 157286400 REUSE
DEFAULT STORAGE (INITIAL 10240 NEXT 10240
MINEXTENTS 1 MAXEXTENTS 121
PCTINCREASE 50)
ONLINE
/
```

```
CREATE TABLESPACE temp
DATAFILE '/oracle1/ORCSPCD1/data/temp01.dbf' SIZE 157286400 REUSE
DEFAULT STORAGE (INITIAL 10240 NEXT 10240
MINEXTENTS 1 MAXEXTENTS 121
PCTINCREASE 50)
ONLINE
/

CREATE TABLESPACE rbs
DATAFILE '/oracle1/ORCSPCD1/data/rbs01.dbf' SIZE 157286400 REUSE
DEFAULT STORAGE (INITIAL 102400 NEXT 102400
MINEXTENTS 2 MAXEXTENTS 121
PCTINCREASE 0)
ONLINE
/


CREATE TABLESPACE cspc_data_dyn
DATAFILE '/oracle1/ORCSPCD1/data/cspc_data_dyn01.dbf' SIZE 157286400 REUSE
DEFAULT STORAGE (INITIAL 10240 NEXT 10240
MINEXTENTS 1 MAXEXTENTS 121
PCTINCREASE 50)
ONLINE
/

CREATE TABLESPACE cspc_data_stat
DATAFILE '/oracle2/ORCSPCD1/data/cspc_data_stat01.dbf' SIZE 52428800 REUSE
DEFAULT STORAGE (INITIAL 10240 NEXT 10240
MINEXTENTS 1 MAXEXTENTS 121
PCTINCREASE 50)
ONLINE
/

CREATE TABLESPACE cspc_indx
DATAFILE '/oracle3/ORCSPCD1/data/cspc_indx01.dbf' SIZE 157286400 REUSE
DEFAULT STORAGE (INITIAL 10240 NEXT 10240
MINEXTENTS 1 MAXEXTENTS 121
PCTINCREASE 50)
ONLINE
/

CREATE TABLESPACE cspc_rbs
DATAFILE '/oracle4/ORCSPCD1/data/cspc_rbs01.dbf' SIZE 157286400 REUSE
DEFAULT STORAGE (INITIAL 10240 NEXT 10240
MINEXTENTS 1 MAXEXTENTS 121
PCTINCREASE 50)
ONLINE
/

CREATE TABLESPACE cspc_temp
DATAFILE '/oracle4/ORCSPCD1/data/cspc_temp01.dbf' SIZE 31457280 REUSE
DEFAULT STORAGE (INITIAL 10240 NEXT 10240
MINEXTENTS 1 MAXEXTENTS 121
PCTINCREASE 50)
ONLINE
/
```

One thing to note about this output is that the SYSTEM tablespace will not be included, if you have multiple datafiles for your existing SYSTEM tablespace and want to rebuild it exactly as it exists, add a ALTER TABLESPACE system ADD DATAFILE section to the script for these files. The sizes and locations can be obtained from the DBA_DATA_FILES view with the following select:

```
SELECT file_name,bytes
from DBA_DATA_FILES
where tablespace_name='SYSTEM';
```

Soft documentation of the tablespaces in a database can be obtained from the scripts datafile.sql, free_spc.sql, db_tbsp.sql located in Appendix A.

Documentation of Rollback Segments

Before anything else in the database can be created, the rollback segments have to be built. If the scripts so far have been run a functioning database will result, a copy of our tablespaces, but only the default SYSTEM rollback segment. The next script will create DDL to make an exact copy of the existing rollback segment profile.

```
REM rbk_rct.sql
REM FUNCTION: SCRIPT FOR CREATING ROLLBACK SEGMENTS
REM
REM This script must be run by a user with the DBA role.
REM
REM This script is intended to run with Oracle7.
REM
REM Running this script will in turn create a script to re-build
REM the database rollback segments.  The created script is called
REM crt_rbks.sql and can be run by any user with the DBA
REM role or with the 'CREATE ROLLBACK SEGMENT' system privilege.
REM
REM NOTE:  This script will NOT capture the optimal storage for
REM        a rollback segment that is offline.
REM
REM        The rollback segments must be manually brought online
REM        after running the crt_rbks.sql script.
REM
REM        Only preliminary testing of this script was performed.  Be
REM        sure to test it completely before relying on it.
```

```
REM

set verify off
set feedback off
rem set termout off
rem set echo off
set pagesize 0

set termout on
prompt Creating rollback segment build script...
set termout off

create table rb_temp (lineno NUMBER, rb_name varchar2(30),
                   text varchar2(800))
                      tablespace temp;

DECLARE
   CURSOR rb_cursor IS select segment_name,
                            tablespace_name,
                            decode (owner, 'PUBLIC', 'PUBLIC ', NULL),
                            segment_id,
                            initial_extent,
                            next_extent,
                            min_extents,
                            max_extents,
                            status
                       from sys.dba_rollback_segs
                            where segment_name <> 'SYSTEM';
   CURSOR rb_optimal (r_no number) IS select usn,
                             decode(optsize, null, 'NULL',
to_char(optsize))
                       from sys.v_$rollstat
                             where usn=r_no;
   lv_segment_name        sys.dba_rollback_segs.segment_name%TYPE;
   lv_tablespace_name     sys.dba_rollback_segs.tablespace_name%TYPE;
   lv_owner               VARCHAR2(10);
   lv_segment_id          sys.dba_rollback_segs.segment_id%TYPE;
   lv_initial_extent      sys.dba_rollback_segs.initial_extent%TYPE;
   lv_next_extent         sys.dba_rollback_segs.next_extent%TYPE;
   lv_min_extents         sys.dba_rollback_segs.min_extents%TYPE;
   lv_max_extents         sys.dba_rollback_segs.max_extents%TYPE;
   lv_status              sys.dba_rollback_segs.status%TYPE;
   lv_usn                 sys.v_$rollstat.usn%TYPE;
   lv_optsize             VARCHAR2(40);
   lv_string              VARCHAR2(800);
   lv_lineno             number := 0;

   procedure write_out(p_line INTEGER, p_name VARCHAR2, p_string VARCHAR2)
is
   begin
       insert into rb_temp (lineno, rb_name, text) values
              (p_line, p_name, p_string);
   end;

BEGIN
  OPEN rb_cursor;
  LOOP
     FETCH rb_cursor INTO lv_segment_name,
```

```
                        lv_tablespace_name,
                        lv_owner,
                        lv_segment_id,
                        lv_initial_extent,
                        lv_next_extent,
                        lv_min_extents,
                        lv_max_extents,
                        lv_status;
     EXIT WHEN rb_cursor%NOTFOUND;
      lv_lineno := 1;
  OPEN rb_optimal(lv_segment_id);
  LOOP
     FETCH rb_optimal INTO lv_usn,
                           lv_optsize;
     EXIT WHEN rb_optimal%NOTFOUND;
  END LOOP;
  CLOSE rb_optimal;
if lv_status = 'ONLINE' then
lv_string:='CREATE ' || lv_owner || 'ROLLBACK SEGMENT ' ||
            lower(lv_segment_name);
write_out(lv_lineno, lv_segment_name, lv_string);
lv_lineno := lv_lineno + 1;
lv_string:='TABLESPACE ' || lower(lv_tablespace_name);
write_out(lv_lineno, lv_segment_name, lv_string);
lv_lineno := lv_lineno + 1;
lv_string:='STORAGE ' || '(INITIAL ' || lv_initial_extent || ' NEXT ' ||
            lv_next_extent || ' MINEXTENTS ' || lv_min_extents ||
            ' MAXEXTENTS ' || lv_max_extents ||
            ' OPTIMAL ' || lv_optsize || ')' ;
write_out(lv_lineno, lv_segment_name, lv_string);
lv_lineno := lv_lineno + 1;
lv_string:='/';
write_out(lv_lineno, lv_segment_name, lv_string);
else
lv_string:='CREATE ' || lv_owner || 'ROLLBACK SEGMENT ' ||
            lower(lv_segment_name);
write_out(lv_lineno, lv_segment_name, lv_string);
lv_lineno := lv_lineno + 1;
lv_string:='TABLESPACE ' || lower(lv_tablespace_name);
write_out(lv_lineno, lv_segment_name, lv_string);
lv_lineno := lv_lineno + 1;
lv_string:='STORAGE ' || '(INITIAL ' || lv_initial_extent || ' NEXT ' ||
            lv_next_extent || ' MINEXTENTS ' || lv_min_extents ||
            ' MAXEXTENTS ' || lv_max_extents || ')' ;
write_out(lv_lineno, lv_segment_name, lv_string);
lv_lineno := lv_lineno + 1;
lv_string:='/';
write_out(lv_lineno, lv_segment_name, lv_string);
end if;
lv_lineno := lv_lineno + 1;
lv_string:='                                                    ';
write_out(lv_lineno, lv_segment_name, lv_string);
END LOOP;
  CLOSE rb_cursor;
END;
/
 column dbname new_value db noprint
select value dbname from v$parameter where name='db_name';
```

```
spool rep_out\&db\crt_rbks.sql
set heading off
col text format a80 word_wrap
select   text
from     rb_temp
order by rb_name, lineno;
spool off;
Drop table rb_temp;
exit
```

Example output from this script follows:

```
CREATE ROLLBACK SEGMENT r01
TABLESPACE cspc_rbs
STORAGE (INITIAL 1048576 NEXT 1048576 MINEXTENTS 2 MAXEXTENTS 121 OPTIMAL
4216832)
/

CREATE ROLLBACK SEGMENT r02
TABLESPACE cspc_rbs
STORAGE (INITIAL 1048576 NEXT 1048576 MINEXTENTS 2 MAXEXTENTS 121 OPTIMAL
4216832)
/

CREATE ROLLBACK SEGMENT r03
TABLESPACE cspc_rbs
STORAGE (INITIAL 1048576 NEXT 1048576 MINEXTENTS 2 MAXEXTENTS 121 OPTIMAL
4216832)
/

CREATE ROLLBACK SEGMENT r04
TABLESPACE cspc_rbs
STORAGE (INITIAL 1048576 NEXT 1048576 MINEXTENTS 2 MAXEXTENTS 121 OPTIMAL
4216832)
/
```

The script documents the rollbacks segments as they currently exist, the DBA should edit the scripts to make any improvements that are required or desired before executing it on the new database.

Soft documentation of rollback segments can be obtained with the rbk1.sql, rbk2.sql and db_rbk.sql scripts located in Appendix A.

Documenting Roles, Grants and Users

Before database objects such as tables, indexes, constraints and the like can be created, there must be users (or schemas) with the appropriate roles and grants to create and own them. The next set of scripts is used to generate DDL to create users, roles and grants. Since a user must have roles and grants, the scripts to document these objects will be discussed first.

Roles

A script to recreate the roles in the database follows:

```
REM role_rct.sql
REM
REM FUNCTION: SCRIPT FOR CREATING ROLES
REM
REM This script must be run by a user with the DBA role.
REM
REM This script is intended to run with Oracle7.
REM
REM Running this script will in turn create a script to build all the
REM roles in the database.  This created file, crt_role.sql, can
REM be run by any user with the DBA role or with 'CREATE ROLE' system
REM privilege.
REM
REM Since it is not possible to create a role under a specific schema,
REM it is essential that the original creator be granted 'ADMIN' option
REM to the role. Therefore, such grants will be made at the end of the
REM crt_role.sql script.  Since it is not possible to distinguish the
REM creator from someone who was simply granted 'WITH ADMIN OPTION', all
REM grants will be spooled. In addition, the user who creates the role
REM is automatically granted 'ADMIN' option on the role, therefore, if
REM this script is run a second time, this user will also be granted
REM 'ADMIN' on all the roles.  You must explicitly revoke 'ADMIN OPTION'
REM from this user to prevent this from happening.
REM
REM NOTE:  This script will not capture the create or grant on the
REM Oracle predefined roles, CONNECT, RESOURCE, DBA, EXP_FULL_DATABASE,
REM or IMP_FULL_DATABASE.
REM
REM Only preliminary testing of this script was performed.  Be sure to
REM test it completely before relying on it.
REM
set verify off feedback off termout off echo off pagesize 0 embedded on
```

```
set termout on
Prompt 'Creating role build script...'
set termout off

 column dbname new_value db noprint
select value dbname from v$parameter where name='db_name';
spool rep_out\&db\crt_role.sql

select 'CREATE ROLE ' || lower(role) || ' NOT IDENTIFIED;'
  from sys.dba_roles
  where role not in ('CONNECT','RESOURCE','DBA', 'EXP_FULL_DATABASE',
                     'IMP_FULL_DATABASE')
  and password_required='NO'
/
select 'CREATE ROLE ' || lower(role) || ' IDENTIFIED BY VALUES ' ||
       '''' || password || '''' || ';'
  from sys.dba_roles, sys.user$
  where role not in ('CONNECT','RESOURCE','DBA', 'EXP_FULL_DATABASE',
                     'IMP_FULL_DATABASE')
  and password_required='YES' and
     dba_roles.role=user$.name
     and user$.type=0
/

select 'GRANT ' || lower(granted_role) || ' TO ' || lower(grantee) ||
       ' WITH ADMIN OPTION;'
  from sys.dba_role_privs
  where admin_option='YES'
  and granted_role not in ('CONNECT','RESOURCE','DBA',
'EXP_FULL_DATABASE',
                           'IMP_FULL_DATABASE')
  order by grantee
/
spool off
exit
```

Example output from the above script follows:

```
CREATE ROLE cspc_dev NOT IDENTIFIED;
CREATE ROLE application_developer NOT IDENTIFIED;
CREATE ROLE cspc_user NOT IDENTIFIED;
GRANT cspc_dev TO sys WITH ADMIN OPTION;
GRANT application_developer TO system WITH ADMIN OPTION;
GRANT cspc_user TO system WITH ADMIN OPTION;
```

The next step for roles is recreating the grants which have been given to the roles created. The next script creates the DDL to grant the privileges to the roles created above:

```
REM FUNCTION:   SCRIPT FOR CAPTURING ROLE GRANTS
REM
REM This script must be run by a user with the DBA role.
REM
REM This script is intended to run with Oracle7.
```

```
REM
REM Running this script will create a script of all the grants
REM of roles to users and other roles.  This created script,
REM grt_role.sql, must be run by a user with the DBA role.
REM
REM Since role grants are not dependant on the schema that issued the
REM grant, the grt_role.sql script will not issue the grant of a role by
REM the original grantor.  All grants will be issued by the user
REM specified when running this script.
REM
REM NOTE:  Grants made to 'SYS','CONNECT','RESOURCE','DBA',
REM            'EXP_FULL_DATABASE','IMP_FULL_DATABASE' are not captured.
REM
REM        Only preliminary testing of this script was performed.  Be
REM        sure to test it completely before relying on it.
REM
set verify off feedback off termout off echo off pagesize 0 embedded on

set termout on
select 'Creating role grant script...' from dual;
set termout off

column dbname new_value db noprint
select value dbname from v$parameter where name='db_name';
spool rep_out\&db\grt_role.sql

select 'GRANT ' || lower(granted_role) || ' TO ' || lower(grantee) ||
       decode(admin_option,'YES',' WITH ADMIN OPTION;',';')
  from sys.dba_role_privs
  where grantee not in ('SYS','CONNECT','RESOURCE','DBA',
       'EXP_FULL_DATABASE','IMP_FULL_DATABASE')
order by grantee
/
spool off
exit
```

Example Output from above script:

```
GRANT connect TO cspcdba WITH ADMIN OPTION;
GRANT cspc_dev TO cspcdba;
GRANT dba TO cspcdba WITH ADMIN OPTION;
GRANT exp_full_database TO cspcdba WITH ADMIN OPTION;
GRANT imp_full_database TO cspcdba WITH ADMIN OPTION;
GRANT resource TO cspcdba WITH ADMIN OPTION;
GRANT connect TO cspcdev_1;
GRANT cspc_user TO cspcdev_1;
GRANT connect TO ops$oracle;
GRANT dba TO ops$oracle;
GRANT resource TO ops$oracle;
```

The above script will capture all privileges granted to roles. Any system level privileges and be captured by a similar script that uses the DBA_SYS_GRANTS view.

```
REM FUNCTION:   SCRIPT FOR CAPTURING ROLE SYSTEM GRANTS
REM
REM This script must be run by a user with the DBA role.
REM
REM This script is intended to run with Oracle7.
REM
REM Running this script will create a script of all the grants
REM of roles to users and other roles.  This created script,
REM grt_sys.sql, must be run by a user with the DBA role.
REM
REM Since role grants are not dependant on the schema that issued the
REM grant, the grt_role.sql script will not issue the grant of a role by
REM the original grantor.  All grants will be issued by the user
REM specified when running this script.
REM
REM NOTE:  Grants made to 'SYS','CONNECT','RESOURCE','DBA',
REM            'EXP_FULL_DATABASE','IMP_FULL_DATABASE' are not captured.
REM
REM        Only preliminary testing of this script was performed.  Be
REM        sure to test it completely before relying on it.
REM
set verify off feedback off termout off echo off pagesize 0 embedded on
set heading off

set termout on
prompt Creating role sys grant script...
set termout off

column dbname new_value db noprint
select value dbname from v$parameter where name='db_name';
spool rep_out\&db\grt_sys.sql

select 'GRANT ' || lower(privilege) || ' TO ' || lower(grantee) ||
       decode(admin_option,'YES',' WITH ADMIN OPTION;',';')
  from sys.dba_sys_privs
  where grantee not in ('SYS','CONNECT','RESOURCE','DBA',
       'EXP_FULL_DATABASE','IMP_FULL_DATABASE')
order by grantee
/
spool off
exit
```

Example output from the above script:

```
GRANT alter session TO application_developer;
GRANT create sequence TO application_developer;
GRANT create session TO application_developer;
GRANT create table TO application_developer;
GRANT create view TO application_developer;
GRANT unlimited tablespace TO cspcdba WITH ADMIN OPTION;
GRANT create session TO cspcdev_1;
GRANT create procedure TO cspc_dev;
GRANT create sequence TO cspc_dev;
GRANT create synonym TO cspc_dev;
GRANT create table TO cspc_dev;
GRANT create trigger TO cspc_dev;
```

```
GRANT create view TO cspc_dev;
GRANT create session TO cspc_user;
GRANT analyze any TO ops$oracle;
GRANT unlimited tablespace TO ops$oracle;
GRANT unlimited tablespace TO system WITH ADMIN OPTION;
```

The next layer of grants which must be allowed for are table grants. The following script will regenerate table grants for roles and users:

```
REM FUNCTION:   SCRIPT FOR CAPTURING TABLE GRANTS
REM
REM This script must be run by a user with the DBA role.
REM
REM This script is intended to run with Oracle7.
REM
REM Running this script will create a script of all the grants
REM of roles to users and other roles.  This created script,
REM grt_sys.sql, must be run by a user with the DBA role.
REM
REM Since role grants are not dependant on the schema that issued the
REM grant, the grt_role.sql script will not issue the grant of a role by
REM the original grantor.  All grants will be issued by the user
REM specified when running this script.
REM
REM NOTE:  Grants made to 'SYS','CONNECT','RESOURCE','DBA',
REM            'EXP_FULL_DATABASE','IMP_FULL_DATABASE' are not captured.
REM
REM        Only preliminary testing of this script was performed.  Be
REM        sure to test it completely before relying on it.
REM
set verify off feedback off termout off echo off pagesize 0 embedded on

set termout on
prompt Creating table grant script...
set termout off

break on line1
column dbname new_value db noprint
select value dbname from v$parameter where name='db_name';
spool rep_out\&db\grt_tabs.sql

select 'CONNECT '||grantor||'/'||grantor line1,
'GRANT
'||lower(privilege)||' on '||owner||'.'||table_name||' to '||
lower(grantee) ||
decode(grantable,'YES',' WITH ADMIN OPTION;',';')
from sys.dba_tab_privs
where grantee not in ('SYS','CONNECT','RESOURCE','DBA',
'EXP_FULL_DATABASE','IMP_FULL_DATABASE')
order by line1,grantee
/
spool off
exit
```

This script will produce a list of grantors and the commands to grant whatever access they have granted to other users and roles on their tables. The script will have to be edited to insert the appropriate passwords for the granting users.

Example of the output of this script:

```
CONNECT CSPCDBA/CSPCDBA
GRANT
alter on CSPCDBA.ACTSEQ to cspcdev_1;
GRANT
select on CSPCDBA.ACTSEQ to cspcdev_1;
GRANT
delete on CSPCDBA.ACCCAR to cspcdev_1;
GRANT
select on CSPCDBA.CSPCSEQ to cspcdev_1;
GRANT
update on CSPCDBA.ACCCAR to cspcdev_1;
GRANT
delete on CSPCDBA.ACT to cspcdev_1;
GRANT
select on CSPCDBA.ACT to cspcdev_1;
```

The final layer of grants are the column level grants. The next script will allow documentation and recreation of these grants.

```
REM FUNCTION:   SCRIPT FOR CAPTURING TABLE COLUMN GRANTS
REM
REM This script must be run by a user with the DBA role.
REM
REM This script is intended to run with Oracle7.
REM
REM Running this script will create a script of all the grants
REM of roles to users and other roles.  This created script,
REM grt_sys.sql, must be run by a user with the DBA role.
REM
REM Since role grants are not dependant on the schema that issued the
REM grant, the grt_role.sql script will not issue the grant of a role by
REM the original grantor.  All grants will be issued by the user
REM specified when running this script.
REM
REM NOTE:  Grants made to 'SYS','CONNECT','RESOURCE','DBA',
REM            'EXP_FULL_DATABASE','IMP_FULL_DATABASE' are not captured.
REM
REM        Only preliminary testing of this script was performed.  Be
REM        sure to test it completely before relying on it.
REM
set verify off feedback off termout off echo off pagesize 0 embedded on
set heading off

set termout on
prompt Creating table grant script...
```

```
set termout off

break on grantor
column dbname new_value db noprint
select value dbname from v$parameter where name='db_name';
spool rep_out\&db\grt_tabc.sql

select 'connect '||grantor||'/'||grantor line1,
'grant
'||lower(privilege)||' on '||owner||'.'||table_name||'.'||column_name||'
'||' to '||lower(grantee)||
decode(grantable,'YES',' WITH ADMIN OPTION;',';')
 from sys.dba_col_privs
where grantee not in ('SYS','CONNECT','RESOURCE','DBA',
'EXP_FULL_DATABASE','IMP_FULL_DATABASE')
order by grantor,grantee
/
spool off
exit
```

Example of the output from the above script follows:

```
connect SYSTEM/SYSTEM
grant
update on SYSTEM.TOOL_DEPENDENT.DEPCHANGED
 to public;
```

The scripts to create the roles and role grants can be run at any time. The scripts for user grants and table grants must be run after the users and tables are created. The next script will recreate the database users as they currently exist (except for passwords which the DBA will have to edit in afterwards.)

```
REM rct_usrs.sql
REM
REM FUNCTION: Create a script to recreate users
REM
REM This script is designed to run on an ORACLE7.x database
REM
REM This script creates a script called crt_usrs.sql that
REM recreates the CREATE USER commands required to rebuild the database
REM user community. The script includes the tablespace quota grants for
REM each user as a set of ALTER USER commands. The user's passwords are
REM initially set to the username so editing is suggested if other
REM values are desired.
REM
REM Only preliminary testing has been accomplished on this script,
REM please fully qualify it for your environment before use
REM
REM M. Ault TRECOM 3.30.96
REM
set verify off feedback off termout off echo off pagesize 0 embedded on
```

```
set termout on
prompt Creating user create script...
set termout off

column dbname new_value db noprint
select value dbname from v$parameter where name='db_name';
spool rep_out\&db\crt_usrs.sql

SELECT 'create user '||username||' identified by '||username||'
 '||' default tablespace '||default_tablespace||'
 '||' temporary tablespace '||temporary_tablespace||'
 '||' profile '||profile||'
 '||' quota unlimited on '||default_tablespace||';'
FROM dba_users
WHERE username not in ('SYS','SYSTEM')
UNION
SELECT 'alter user '||username||'
'||'quota '||bytes||' on '||tablespace_name||';'
FROM dba_ts_quotas
WHERE username not in ('SYS','SYSTEM')
/
spool off
exit
```

Example output from the above script:

```
create user CSPCDEV_1 identified by CSPCDEV_1
  default tablespace CSPC_DATA_DYN
  temporary tablespace CSPC_TEMP
  profile DEFAULT
  quota unlimited on CSPC_DATA_DYN;

create user OPS$ORACLE identified by OPS$ORACLE
  default tablespace USERS
  temporary tablespace CSPC_TEMP
  profile DEFAULT
  quota unlimited on USERS;
```

Now we have fully documented all users, roles and grants. The next sets of scripts provide documentation for the tables and their associated indexes, constraints, procedures, triggers and views.

Soft documentation of roles, grants and users can be obtained by using the db_roles.sql, db_grnts.sql and db_usrs.sql scripts located in Appendix A.

Documenting Tables

So far the database control file, redo logs, initialization parameters, tablespaces, rollback segments, roles, grants and users have been documented. All non-data related structures. It is now time to document the application tables. If the database has been properly set up (always a questionable proposition) the tables and other related data structures should be owned by one application owner or schema. If the application is owned by more than one schema, or, there is more than one application in the database, the following script will need to be run multiple times.

The script to document existing tables follows:

```
REM tab_rct.sql
REM
REM FUNCTION: SCRIPT FOR CREATING TABLES
REM
REM           This script can be run by any user .
REM
REM           This script is intended to run with Oracle7.
REM
REM           Running this script will in turn create a script to
REM           build all the tables owned by the user in the database.
REM           This script, crt_tab.sql, can be run by any user with the
REM           'CREATE TABLE' system privilege.
REM
REM NOTE:     The script will NOT include constraints on tables.  This
REM           script will also NOT capture tables created by user 'SYS'.
REM
REM Only preliminary testing of this script was performed.  Be sure to
REM test it completely before relying on it.
REM
set verify off
rem set feedback off
rem set termout off
set echo off;
set pagesize 0
set termout on
Prompt Creating table build script...
set termout off

create table t_temp
     (lineno NUMBER, tb_owner VARCHAR2(30), tb_name VARCHAR2(30),
      text VARCHAR2(255))
/

DECLARE
   CURSOR tab_cursor IS
```

```
      select         table_name,
             pct_free,
             pct_used,
             ini_trans,
             max_trans,
             tablespace_name,
             initial_extent,
             next_extent,
             min_extents,
             max_extents,
             pct_increase
      from     user_tables
      order by  table_name;

  CURSOR col_cursor (c_tab VARCHAR2) IS
      select
        column_name,
        data_type,
        data_length,
        data_precision,
        data_scale,
        nullable
      from     user_tab_columns
      where    table_name = c_tab
      order by column_name;

  lv_table_name                    user_tables.table_name%TYPE;
  lv_pct_free              user_tables.pct_free%TYPE;
  lv_pct_used              user_tables.pct_used%TYPE;
  lv_ini_trans             user_tables.ini_trans%TYPE;
  lv_max_trans                     user_tables.max_trans%TYPE;
  lv_tablespace_name       user_tables.tablespace_name%TYPE;
  lv_initial_extent        user_tables.initial_extent%TYPE;
  lv_next_extent           user_tables.next_extent%TYPE;
  lv_min_extents           user_tables.min_extents%TYPE;
  lv_max_extents           user_tables.max_extents%TYPE;
  lv_pct_increase          user_tables.pct_increase%TYPE;
  lv_column_name           user_tab_columns.column_name%TYPE;
  lv_data_type             user_tab_columns.data_type%TYPE;
  lv_data_length           user_tab_columns.data_length%TYPE;
  lv_data_precision        user_tab_columns.data_precision%TYPE;
  lv_data_scale            user_tab_columns.data_scale%TYPE;
  lv_nullable              user_tab_columns.nullable%TYPE;
  lv_first_rec             BOOLEAN;
  lv_lineno                NUMBER := 0;
  lv_string                VARCHAR2(80);

  procedure write_out(p_line INTEGER,  p_name VARCHAR2,
                    p_string VARCHAR2) is
       begin
          insert into t_temp (lineno, tb_name, text)
                 values (p_line,p_name,p_string);
        end;
BEGIN
  OPEN tab_cursor;
  LOOP
     FETCH tab_cursor INTO          lv_table_name,
                           lv_pct_free,
```

```
                        lv_pct_used,
                        lv_ini_trans,
                        lv_max_trans,
                        lv_tablespace_name,
                        lv_initial_extent,
                        lv_next_extent,
                        lv_min_extents,
                        lv_max_extents,
                        lv_pct_increase;
EXIT WHEN tab_cursor%NOTFOUND;
  lv_lineno := 1;
  lv_string := 'DROP TABLE '|| lower(lv_table_name)||';';
  write_out(lv_lineno,  lv_table_name, lv_string);
  lv_lineno := lv_lineno + 1;
  lv_first_rec := TRUE;
  lv_string := 'CREATE TABLE '|| lower(lv_table_name)||' (';
  write_out(lv_lineno,  lv_table_name, lv_string);
  lv_lineno := lv_lineno + 1;
lv_string := null;
OPEN col_cursor(lv_table_name);
LOOP
   FETCH col_cursor INTO  lv_column_name,
                        lv_data_type,
                        lv_data_length,
                        lv_data_precision,
                        lv_data_scale,
                        lv_nullable;
   EXIT WHEN col_cursor%NOTFOUND;
   if (lv_first_rec) then
      lv_first_rec := FALSE;
   else
      lv_string :=  ',';
   end if;
   lv_string := lv_string || lower(lv_column_name) ||
              ' ' || lv_data_type;
   if ((lv_data_type = 'CHAR') or (lv_data_type = 'VARCHAR2')) then
      lv_string := lv_string || '(' || lv_data_length || ')';
   end if;
   if (lv_nullable = 'N') then
      lv_string := lv_string || ' NOT NULL';
   end if;
write_out(lv_lineno, lv_table_name, lv_string);
lv_lineno := lv_lineno + 1;
END LOOP;
CLOSE col_cursor;
lv_string := ')';
write_out(lv_lineno, lv_table_name, lv_string);
lv_lineno := lv_lineno + 1;
lv_string := null;
lv_string := 'PCTFREE ' || to_char(lv_pct_free) ||
         '   PCTUSED ' || to_char(lv_pct_used);
write_out(lv_lineno,  lv_table_name, lv_string);
lv_lineno := lv_lineno + 1;
lv_string := 'INITRANS ' || to_char(lv_ini_trans) ||
           ' MAXTRANS ' || to_char(lv_max_trans);
write_out(lv_lineno,  lv_table_name, lv_string);
lv_lineno := lv_lineno + 1;
lv_string := 'TABLESPACE ' || lv_tablespace_name;
```

```
      write_out(lv_lineno,  lv_table_name, lv_string);
      lv_lineno := lv_lineno + 1;
      lv_string := 'STORAGE (';
      write_out(lv_lineno,  lv_table_name, lv_string);
      lv_lineno := lv_lineno + 1;
      lv_string := 'INITIAL ' || to_char(lv_initial_extent) ||
                   ' NEXT ' || to_char(lv_next_extent);
      write_out(lv_lineno,  lv_table_name, lv_string);
      lv_lineno := lv_lineno + 1;
      lv_string := 'MINEXTENTS ' || to_char(lv_min_extents) ||
                ' MAXEXTENTS ' || to_char(lv_max_extents) ||
              ' PCTINCREASE ' || to_char(lv_pct_increase) || ')';
      write_out(lv_lineno,  lv_table_name, lv_string);
      lv_lineno := lv_lineno + 1;
      lv_string := '/';
      write_out(lv_lineno,  lv_table_name, lv_string);
      lv_lineno := lv_lineno + 1;
      lv_string:='                                                    ';
      write_out(lv_lineno,  lv_table_name, lv_string);
   END LOOP;
   CLOSE tab_cursor;
END;
/

set heading off
spool rep_out\crt_tabs.sql
select   text
from     T_temp
order by  tb_name, lineno;
spool off
drop table t_temp
exit
```

Again, the recursion required by the multiple entries in the USER_TAB_COLUMNS view cause multiple cursors to be used and multiple loop-end loop constructs to be required. This script can be made generic by adding reference to the OWNER column of the DBA_TABLES and DBA_TAB_COLUMNS views throughout the logic. An example of this scripts output follows:

```
DROP TABLE accar;
CREATE TABLE acccar (
acccaridcd NUMBER
,acccarrdrccd VARCHAR2(3)
,acccartypcd VARCHAR2(10)
,acccustnmabbr VARCHAR2(3) NOT NULL
,custcarnmabbr VARCHAR2(3)
,fk_cococd VARCHAR2(15) NOT NULL
,fk_cofk_leleid NUMBER NOT NULL
)
PCTFREE 60 PCTUSED 40
INITRANS 1 MAXTRANS 255
TABLESPACE CSPC_DATA_DYN
STORAGE (
```

```
INITIAL 118784 NEXT 55296
MINEXTENTS 1 MAXEXTENTS 121 PCTINCREASE 50)
/
 DROP TABLE act;
CREATE TABLE act (
actseqnbr NUMBER NOT NULL
,actstpdt DATE
,actstrtdt DATE NOT NULL
,actvyhilgtind VARCHAR2(1)
,fk_acttypacttypnm VARCHAR2(16)
,fk_acttypacttypvrs CHAR(7)
,fk_eventevntid NUMBER NOT NULL
)
PCTFREE 60 PCTUSED 40
INITRANS 1 MAXTRANS 255
TABLESPACE CSPC_DATA_DYN
STORAGE (
INITIAL 4096 NEXT 10240
MINEXTENTS 1 MAXEXTENTS 121 PCTINCREASE 50)
/
.
. (Since this database has nearly 300 tables, the output has been
.  truncated)
.
DROP TABLE wrkgrp_table;
CREATE TABLE wrkgrp_table (
wrkgrpnm VARCHAR2(8) NOT NULL
)
PCTFREE 10 PCTUSED 40
INITRANS 1 MAXTRANS 255
TABLESPACE CSPC_DATA_DYN
STORAGE (
INITIAL 10240 NEXT 10240
MINEXTENTS 1 MAXEXTENTS 121 PCTINCREASE 50)
/
```

As the remarks section of the code relates, this code does not capture table constraints. The reasoning for this is to allow data loading to occur before the constraints are enabled. In the following sections a script the re-creates all primary key, unique, foreign key, non-"not null" check type and default value constraints will be examined.

Soft documentation of tables can be generated using the db_tbl.sql, and tab_stat.sql scripts in Appendix A.

A subset of tables, called clusters, are documented using the cluster scripts in Appendix A. Since very few applications use clusters, this paper won't cover rebuilding them.

Documenting Database Constraints

The next database objects which will be documented are the table constraints. In many cases the constraints will have been specified during the CREATE TABLE command and probably will not have been named. This lack of naming for a constraint will result in Oracle selecting a name consisting of SYS_C with a sequence number appended to it, such as SYS_C00123. Using a derivation of the next script these constraints can be renamed to a more informative naming convention, however that is the topic for another paper and beyond the scope of this one.

Constraints come is several forms, primary key (P), foreign key (R), unique (U), check (C) and default value (V). Check constraints can be the standard NOT-NULL type constraint or can be a full blown verification algorithm. Default value constraints provide a default value for those fields effected by them.

The following script generates three SQL scripts, one for primary key and unique constraints, one for foreign key constraints and one for non-NOT-NULL check constraints and default value constraints:

```
REM
REM FUNCTION: SCRIPT FOR RE-CREATING DATABASE CONSTRAINTS
REM
REM FUNCTION: This script must be run by the constraint owner.
REM
REM FUNCTION: This script is intended to run with Oracle7.
REM
REM FUNCTION: Running this script will in turn create scripts to build
REM FUNCTION: constraints owned in the database. The scripts are called
REM FUNCTION: 'pk_rct.sql', 'fk_rct.sql', and 'ck_rct.sql'.
REM FUNCTION: The primary key and unique
REM FUNCTION: constraints will include the USING INDEX clause, you may
REM FUNCTION: point this to an INDEX tablespace if the  constraints
REM FUNCTION: don't already. The foreign key constraints build an index
REM FUNCTION: if one exists with the same name as the constraint.
REM FUNCTION: Otherwise, the foreign key isn't built.
REM FUNCTION: You may wish to edit this part of the script if this isn't
REM FUNCTION: the case in your database
REM
REM Only preliminary testing of this script was performed.  Be sure to
```

```
REM test it completely before relying on it.
REM
REM M. Ault 2/25/96 TRECOM
REM

set verify off
rem set feedback off
rem set termout off
rem set echo off
set pagesize 0
set long 4000
set termout on
select 'Creating constraint build script...' from dual;
rem set termout off

create table cons_temp (owner varchar2(30),
                        constraint_name varchar2(30),
                        constraint_type varchar2(11),
                        search_condition varchar2(2000),
                        table_name varchar2(30),
                        referenced_owner varchar2(30),
                        referenced_constraint varchar2(30),
                        delete_rule varchar2(9),
                        constraint_columns varchar2(2000),
                        con_number number);

DECLARE

   CURSOR cons_cursor IS select      owner,
                              constraint_name,
   decode(constraint_type,'P','Primary Key',
                        'R','Foreign Key',
                        'U','Unique',
                        'C','Check',
                        'D','Default'),
                              table_name,
                              search_condition,
                              r_owner,
                              r_constraint_name,
                              delete_rule
     from user_constraints
     where owner not in ('SYS','SYSTEM')
     order by owner;

     cursor cons_col is select
               owner,
               constraint_name,
               column_name
     from user_cons_columns
     where owner not in ('SYS','SYSTEM')
     order by owner, constraint_name,
                              position;

       cursor get_cons (tab_nam in varchar2) is
       select distinct
              OWNER,TABLE_NAME,CONSTRAINT_NAME,CONSTRAINT_TYPE
       from cons_temp
       where table_name=tab_nam
```

```
        and constraint_type='Foreign Key'
        order by owner,table_name,constraint_name;

        cursor get_tab_nam is
        select distinct table_name
        from cons_temp
        where constraint_type='Foreign Key'
        order by table_name;

        tab_nam         user_constraints.table_name%TYPE;
        cons_owner              user_constraints.owner%TYPE;
        cons_name               user_constraints.constraint_name%TYPE;
        cons_type               varchar2(11);
        cons_sc         user_constraints.search_condition%TYPE;
        cons_tname              user_constraints.table_name%TYPE;
        cons_rowner             user_constraints.r_owner%TYPE;
        cons_rcons              user_constraints.r_constraint_name%TYPE;
        cons_dr         user_constraints.delete_rule%TYPE;
        cons_col_own    user_cons_columns.owner%TYPE;
        cons_col_nam    user_cons_columns.constraint_name%TYPE;
        cons_column             user_cons_columns.column_name%TYPE;
        cons_tcol_name user_cons_columns.table_name%TYPE;
        all_columns     varchar2(2000);
        counter         integer:=0;
        cons_nbr        integer;

BEGIN
   OPEN cons_cursor;
   LOOP
      FETCH cons_cursor INTO  cons_owner,
                              cons_name,
                              cons_type,
                              cons_sc,
                              cons_tname,
                              cons_rowner,
                                      cons_rcons,
                              cons_dr;
      EXIT WHEN cons_cursor%NOTFOUND;
       all_columns :='';
       counter := 0;
       open cons_col;
       loop
              fetch cons_col into
                                      cons_col_own,
                                      cons_col_nam,
                                      cons_column;
       exit when cons_col%NOTFOUND;
              if cons_owner = cons_col_own and cons_name=cons_col_nam
              then
                     counter := counter+1;
                     if counter = 1 then
                     all_columns := all_columns||cons_column;
                     else
                     all_columns := all_columns||', '||cons_column;
                     end if;
              end if;
       end loop;
       close cons_col;
```

```
        insert into cons_temp values (cons_owner,
                                      cons_name,
                                      cons_type,
                                      cons_tname,
                                      cons_sc,
                                      cons_rowner,
                                      cons_rcons,
                                      cons_dr,
                                all_columns,
                                0);
   commit;
   END LOOP;
   CLOSE cons_cursor;
   commit;
begin
 open get_tab_nam;
loop
  fetch get_tab_nam into tab_nam;
  exit when get_tab_nam%NOTFOUND;
/*sys.dbms_output.put_line(tab_nam);*/
  open get_cons (tab_nam);
  cons_nbr:=0;
  loop
    fetch get_cons into cons_owner,
                        cons_tname,
                        cons_name,
                        cons_type;
    exit when get_cons%NOTFOUND;
    cons_nbr:=cons_nbr+1;
/*    sys.dbms_output.put_line('cons_nbr='||cons_nbr);*/
/*sys.dbms_output.put_line(cons_owner||'.'||cons_name||' '||cons_type);*/
    update cons_temp set con_number=cons_nbr where
        constraint_name=cons_name and
        constraint_type=cons_type and
        owner=cons_owner;
  end loop;
  close get_cons;
  commit;
end loop;
close get_tab_nam;
commit;
end;
END;
/
REM The following indexes make the create part of this script run
REM much faster
REM
create index pk_cons_temp on cons_temp(constraint_name);
create index lk_cons_temp on cons_temp(constraint_type);
create index lk_cons_temp2 on cons_temp(referenced_constraint);

set heading off feedback off pages 0 termout off echo off
set recsep off verify off
set embedded on
REM
REM Do the check and default values first since they are usually the
smallest
REM
```

```
spool ck_rct.sql
select 'alter table '||a.table_name||'
drop constraint '||a.constraint_name||';
alter table '||a.table_name||' add constraint '||a.constraint_name||'
'||constraint_type||'( '||a.search_condition||' )
;'
from cons_temp a where constraint_type in ( 'Check','Default' ) and
search_condition not like '%NULL%'
/
spool off
REM
REM Next, do the primary constraints
REM
spool pk_rct.sql
select 'alter table '||a.table_name||'
drop constraint '||a.constraint_name||';'||'
drop index '||a.constraint_name||';'||'
alter table '||a.table_name||' add constraint '||a.constraint_name||'
'||constraint_type||'( '||a.constraint_columns||' )'||'
'||'using index tablespace '||b.tablespace_name||'
'||'pctfree '||b.pct_free||'
'||'initrans '||b.ini_trans||'
'||'maxtrans '||b.max_trans||'
'||'storage (
'||'initial '||b.initial_extent||'
'||'next '||b.next_extent||'
'||'minextents '||b.min_extents||'
'||'maxextents '||b.max_extents||'
'||'pctincrease '||b.pct_increase||'
'||' freelist groups '||b.freelist_groups||')
;'
from cons_temp a, user_indexes b where
a.constraint_type in ('Primary Key','Unique') and
a.constraint_name=b.index_name
/
spool off
REM
REM Finally do the foreign key constraints
REM
spool fk_rct.sql
select 'alter table '||a.table_name||'
drop constraint '||a.constraint_name||';
drop index '||c.index_name||';
alter table '||a.table_name||' add constraint '||a.constraint_name||'
'||a.constraint_type||'(
 '||a.constraint_columns||' )
'||'references '||a.referenced_owner||'.'||b.table_name||' (
'||b.constraint_columns||' )
'||decode (a.delete_rule,'CASCADE','on delete cascade','')||'
;'||'
'||'create index '||c.index_name||' on '||a.table_name||'(
'||a.constraint_columns||')
'||'tablespace '||c.tablespace_name||'
'||'pctfree '||c.pct_free||' '||'initrans '||c.ini_trans||' maxtrans
'||c.max_trans||'
'||'storage ( '||'initial '||c.initial_extent||' next '||c.next_extent||'
'||'minextents '||c.min_extents||' maxextents '||c.max_extents||'
'||'pctincrease '||c.pct_increase||'
```

```
'||' freelist groups '||c.freelist_groups||')
;'
from cons_temp a, cons_temp b, user_indexes c
where a.constraint_type = 'Foreign Key' and
a.referenced_constraint=b.constraint_name
and a.constraint_name=c.index_name
/
drop table cons_temp;
exit
```

As the remarks at the top of the script indicate, this script expects the foreign key indexes to exist and be named the same as the foreign key. If this isn't the case, modify the last bit of code to either not build the index at all or hard code the location of the index and its name and storage parameters.

An excerpt from the pk_rct.sql script follows:

```
alter table ACCCAR
drop constraint PK_ACCCAR;
drop index PK_ACCCAR;
alter table ACCCAR add constraint PK_ACCCAR
Primary Key( FK_COCOCD, FK_COFK_LELEID )
using index tablespace CSPC_INDX
pctfree 10
initrans 2
maxtrans 255
storage (
initial 10240
next 10240
minextents 1
maxextents 121
pctincrease 50
 freelist groups 1)
;
alter table ACT
drop constraint PK_ACT;
drop index PK_ACT;
alter table ACT add constraint PK_ACT
Primary Key( FK_EVENTEVNTID, ACTSEQNBR )
using index tablespace CSPC_INDX
pctfree 10
initrans 2
maxtrans 255
storage (
initial 10240
next 24576
minextents 1
maxextents 121
pctincrease 50
 freelist groups 1)
;
alter table ACTASGT
```

```
drop constraint PK_ACTASGT;
drop index PK_ACTASGT;
alter table ACTASGT add constraint PK_ACTASGT
Primary Key( FK_ACTACTSEQNBR, FK_ACTFK_EVENTEVNT, ACTASGTROLECD )
using index tablespace CSPC_INDX
pctfree 10
initrans 2
maxtrans 255
storage (
initial 10240
next 10240
minextents 1
maxextents 121
pctincrease 50
 freelist groups 1)
;
```

An excerpt from the fk_rct.sql script follows:

```
alter table ACCCAR
drop constraint FK_ACCCAR_1;
drop index FK_ACCCAR_1;
alter table ACCCAR add constraint FK_ACCCAR_1 Foreign Key(
 FK_COFK_LELEID, FK_COCOCD )
references CSPCDBA.CO (
FK_LELEID, COCD )
on delete cascade
;
create index FK_ACCCAR_1 on ACCCAR(
FK_COFK_LELEID, FK_COCOCD)
tablespace CSPC_INDX
pctfree 10 initrans 2 maxtrans 255
storage ( initial 10240 next 10240
minextents 1 maxextents 121
pctincrease 50
 freelist groups 1)
;
alter table ACT
drop constraint FK_ACT_1;
drop index FK_ACT_1;
alter table ACT add constraint FK_ACT_1 Foreign Key(
 FK_EVENTEVNTID )
references CSPCDBA.EVENT (
EVNTID )
on delete cascade
;
```

The scripts for constraint recreation should be run after the tables have been built and loaded with data. They should be run in the following order:

1. pk_rct.sql

2. fk_rct.sql
3. ck_rct.sql

There may be additional lookup indexes in the database, in this case, these must also be recreated or performance may suffer. The next script allows you to recreate the other indexes in the database.

The soft documentation for constraints can be generated through use of the pk_fk.sql script in Appendix A.

Documenting Indexes in the Database

Indexes are usually create for one of four purposes, enforcing a primary key, enforcing a unique value, lookup support on a foreign key or performance enhancement. The constraints rebuilding script handles primary key, unique and foreign key type indexes. The next script will allow the DBA to rebuild all indexes in a database, since Oracle will not allow multiple indexes on the same set of columns, running the resulting DDL after rebuilding your constraints will allow creation of any look up indexes without the possibility of damaging already existing indexes used in support of primary, unique or foreign key constraints.

```
REM in_rct.sql
REM
REM FUNCTION: SCRIPT FOR CREATING INDEXES
REM
REM           This script must be run by a user with the DBA role.
REM
REM           This script is intended to run with Oracle7.
REM
REM           Running this script will in turn create a script to
REM           build all the indexes in the database.  This created
REM           script, create_index.sql, can be run by any user with
REM           the DBA role or with the 'CREATE ANY INDEX' system
REM           privilege.
REM
REM           The script will NOT capture the indexes created by
REM           the user 'SYS'.
REM
REM NOTE:     Indexes automatically created by table CONSTRAINTS will
```

```
REM           also be INCLUDED in the create_index.sql script.  It may
REM           cause a problem to create an index with a system assigned
REM           name such as SYS_C00333.
REM
REM           Only preliminary testing of this script was performed.
REM           Be sure to test it completely before relying on it.
REM

set verify off;
set termout off;
set feedback off;
set echo off;
set pagesize 0;

set termout on
Prompt Creating index build script...
set termout off;

create table i_temp
     (lineno NUMBER, id_owner VARCHAR2(30), id_name VARCHAR2(30),
      text VARCHAR2(255))
/

DECLARE
   CURSOR ind_cursor IS select   owner,
                                 index_name,
                                 table_owner,
                                 table_name,
                                 uniqueness,
                                 tablespace_name,
                                 ini_trans,
                                 max_trans,
                                 initial_extent,
                                 next_extent,
                                 min_extents,
                                 max_extents,
                                 pct_increase,
                                 pct_free
                         from     dba_indexes
                         where owner != 'SYS'
                         order by index_name;
   CURSOR col_cursor (i_own VARCHAR2, c_ind VARCHAR2, c_tab VARCHAR2) IS
               select   column_name
                                       from     dba_ind_columns
                                       where    index_owner = i_own
                                         and    index_name = c_ind
                                         and    table_name = c_tab
                                       order by column_position;

   lv_index_owner             dba_indexes.owner%TYPE;
   lv_index_name              dba_indexes.index_name%TYPE;
   lv_table_owner             dba_indexes.table_owner%TYPE;
   lv_table_name              dba_indexes.table_name%TYPE;
   lv_uniqueness              dba_indexes.uniqueness%TYPE;
   lv_tablespace_name         dba_indexes.tablespace_name%TYPE;
   lv_ini_trans               dba_indexes.ini_trans%TYPE;
   lv_max_trans               dba_indexes.max_trans%TYPE;
   lv_initial_extent          dba_indexes.initial_extent%TYPE;
```

```
   lv_next_extent             dba_indexes.next_extent%TYPE;
   lv_min_extents             dba_indexes.min_extents%TYPE;
   lv_max_extents             dba_indexes.max_extents%TYPE;
   lv_pct_increase            dba_indexes.pct_increase%TYPE;
   lv_pct_free                dba_indexes.pct_free%TYPE;
   lv_column_name             dba_ind_columns.column_name%TYPE;
   lv_first_rec               BOOLEAN;
   lv_string                  VARCHAR2(80);
   lv_lineno                  NUMBER := 0;

   procedure write_out(p_line INTEGER, p_owner varchar2, p_name VARCHAR2,
                       p_string VARCHAR2) is
   begin
      insert into i_temp (lineno,id_owner, id_name,text)
            values (p_line,p_owner,p_name,p_string);
   end;

BEGIN
   OPEN ind_cursor;
   LOOP
      FETCH ind_cursor INTO     lv_index_owner,
                                lv_index_name,
                                lv_table_owner,
                                lv_table_name,
                                lv_uniqueness,
                                lv_tablespace_name,
                                lv_ini_trans,
                                lv_max_trans,
                                lv_initial_extent,
                                lv_next_extent,
                                lv_min_extents,
                                lv_max_extents,
                                lv_pct_increase,
                                lv_pct_free;
      EXIT WHEN ind_cursor%NOTFOUND;
        lv_lineno := 1;
      lv_first_rec := TRUE;
      lv_string:= 'DROP INDEX ' || lower(lv_index_owner) || '.' ||
                       lower(lv_index_name)||';';
        write_out(lv_lineno, lv_index_owner, lv_index_name, lv_string);
        lv_lineno := lv_lineno+1;
      if (lv_uniqueness = 'UNIQUE') then
        lv_string:= 'CREATE UNIQUE INDEX ' || lower(lv_index_owner) || '.'
||
                    lower(lv_index_name);
        write_out(lv_lineno, lv_index_owner, lv_index_name, lv_string);
        lv_lineno := lv_lineno + 1;
      else
        lv_string:= 'CREATE INDEX ' || lower(lv_index_owner) || '.' ||
                    lower(lv_index_name);
        write_out(lv_lineno, lv_index_owner, lv_index_name, lv_string);
        lv_lineno := lv_lineno + 1;
      end if;
      OPEN col_cursor(lv_index_owner,lv_index_name,lv_table_name);
      LOOP
        FETCH col_cursor INTO  lv_column_name;
        EXIT WHEN col_cursor%NOTFOUND;
        if (lv_first_rec) then
```

```
            lv_string := '   ON '|| lower(lv_table_owner) || '.' ||
                         lower(lv_table_name)||' (';
         lv_first_rec := FALSE;
         else
            lv_string := lv_string || ',';
         end if;
         lv_string := lv_string || lower(lv_column_name);
      END LOOP;
      CLOSE col_cursor;
      lv_string := lv_string || ')';
      write_out(lv_lineno, lv_index_owner, lv_index_name, lv_string);
      lv_lineno := lv_lineno + 1;
      lv_string := null;
      lv_string := 'PCTFREE ' || to_char(lv_pct_free);
      write_out(lv_lineno, lv_index_owner, lv_index_name, lv_string);
      lv_lineno := lv_lineno + 1;
      lv_string := 'INITRANS ' || to_char(lv_ini_trans) ||
               ' MAXTRANS ' || to_char(lv_max_trans);
      write_out(lv_lineno, lv_index_owner, lv_index_name, lv_string);
      lv_lineno := lv_lineno + 1;
      lv_string := 'TABLESPACE ' || lv_tablespace_name || ' STORAGE (';
      write_out(lv_lineno, lv_index_owner, lv_index_name, lv_string);
      lv_lineno := lv_lineno + 1;
      lv_string := 'INITIAL ' || to_char(lv_initial_extent) ||
                   ' NEXT ' || to_char(lv_next_extent);
      write_out(lv_lineno, lv_index_owner, lv_index_name, lv_string);
      lv_lineno := lv_lineno + 1;
      lv_string := 'MINEXTENTS ' || to_char(lv_min_extents) ||
               ' MAXEXTENTS ' || to_char(lv_max_extents) ||
               ' PCTINCREASE ' || to_char(lv_pct_increase) || ')';
      write_out(lv_lineno, lv_index_owner, lv_index_name, lv_string);
      lv_lineno := lv_lineno + 1;
      lv_string := '/';
      write_out(lv_lineno, lv_index_owner, lv_index_name, lv_string);
      lv_lineno := lv_lineno + 1;
      lv_lineno := lv_lineno + 1;
      lv_string:='                                                    ';
      write_out(lv_lineno, lv_index_owner, lv_index_name, lv_string);
   END LOOP;
   CLOSE ind_cursor;
END;
/
column dbname new_value db noprint
select value dbname from v$parameter where name='db_name';
spool rep_out\&db\crt_indx.sql
set heading off
set recsep off
col text format a80 word_wrap
select   text
from     I_temp
order by id_owner, id_name, lineno;
spool off
drop table i_temp;
exit
```

An excerpt from the resulting DDL script follows:

```
DROP INDEX cspcdba.lu_ofctypview_1;
CREATE INDEX cspcdba.lu_ofctypview_1
ON cspcdba.ofctypview (fk_viewnmviewnm,fk_viewnmviewtyp)
PCTFREE 60
INITRANS 2 MAXTRANS 255
TABLESPACE DSPT_INDX STORAGE (
INITIAL 4096 NEXT 2048
MINEXTENTS 1 MAXEXTENTS 121 PCTINCREASE 50)
/
DROP INDEX cspcdba.lu_ofctypview_2;
CREATE INDEX cspcdba.lu_ofctypview_2
ON cspcdba.ofctypview (fk_ofctypofctypcd)
PCTFREE 60
INITRANS 2 MAXTRANS 255
TABLESPACE DSPT_INDX STORAGE (
INITIAL 4096 NEXT 2048
MINEXTENTS 1 MAXEXTENTS 121 PCTINCREASE 50)
/
.
.
.
DROP INDEX cspcdba.lu_eventcloitem_2;
CREATE INDEX cspcdba.lu_eventcloitem_2
ON cspcdba.eventcloitem
(fk_cloitemcloitemn,fk_cloitemfk_clofk,
fk_cloitemfk_cloc0,fk_cloitemfk_clocl)
PCTFREE 60
INITRANS 2 MAXTRANS 255
TABLESPACE DSPT_INDX STORAGE (
INITIAL 4096 NEXT 2048
MINEXTENTS 1 MAXEXTENTS 121 PCTINCREASE 50)
/
```

So far the “hard” database objects and a few “soft” ones have been documented. What is left are the remainder of the “soft” database objects, these being packages, package bodies, procedures, functions, triggers and views. The next script handles packages, package bodies, procedures and functions.

Soft documentation for indexes can be generated using the pk_fk.sql, db_inx.sql and inx_stat.sql scripts located in Appendix A.

Documenting Sequences

Since sequences aren't a part of standard SQL many people tend to overlook them, until they break. Sequences can break if you exceed their maximum (for an ascending sequence) or minimum (for a descending sequence) values or place to much stress on a single sequence. By documenting your sequences you can see how they were created and have the script to recreate them if it becomes needed. In some situations, such as import or use of SQL*loader, sequences used for key values can be come out of sync with the tables they relate to and will require resetting. Use of a documentation script can speed this process. The following script will generate the DDL to rebuild your indexes:

```
REM
REM    FUNCTION: SCRIPT FOR RE-CREATING DATABASE SEQUENCES
REM
REM              This script must be run by 'SYS'.
REM
REM              This script is intended to run with Oracle7.
REM
REM              Running this script will in turn create a script to
REM              build all the sequences in the database.  This created
REm              script is called 'crt_seq.sql'.
REM
REM              This script will start the sequence (start with value)
REM              at the last value of the sequence at the time the
REM              script is run (LAST_NUMBER).
REM
REM
REM Only preliminary testing of this script was performed.  Be sure to
REM test it completely before relying on it.
REM

set verify off feedback off termout off echo off pages 0

set termout on
select 'Creating sequence build script...' from dual;
set termout off

create table seq_temp (grantor_owner varchar2(30),
text VARCHAR2(255))
/
DECLARE
   CURSOR seq_cursor IS select   sequence_owner,
                                 sequence_name,
                                 min_value,
                                 max_value,
                                increment_by,
```

```
                              decode(cycle_flag,'Y','CYCLE','NOCYCLE'),
                              decode(order_flag,'Y','ORDER','NOORDER'),
       decode(to_char(cache_size),'0','NOCACHE','CACHE
'||to_char(cache_size)),
                              last_number
                         from dba_sequences
                        where sequence_owner not in ('SYS','SYSTEM')
                         order by sequence_owner;
       seq_owner       dba_sequences.sequence_owner%TYPE;
       seq_name        dba_sequences.sequence_name%TYPE;
       seq_min         dba_sequences.min_value%TYPE;
       seq_max         dba_sequences.max_value%TYPE;
       seq_inc         dba_sequences.increment_by%TYPE;
       seq_order       VARCHAR2(7);
       seq_cycle       VARCHAR2(7);
       seq_cache       VARCHAR2(15);
       seq_lnum        dba_sequences.last_number%TYPE;
       seq_string     VARCHAR2(255);
   procedure write_out(p_string VARCHAR2) is
   begin
      insert into seq_temp (grantor_owner,text)
                        values (seq_owner,p_string);
   end;

BEGIN
   OPEN seq_cursor;
   LOOP
      FETCH seq_cursor INTO     seq_owner,
                              seq_name,
                              seq_min,
                              seq_max,
                              seq_inc,
                              seq_order,
                              seq_cycle,
                              seq_cache,
                              seq_lnum;
      EXIT WHEN seq_cursor%NOTFOUND;
       seq_string:=('CREATE SEQUENCE '||seq_owner||'.'||seq_name||'
                              INCREMENT BY '||seq_inc||'
                              START WITH '||seq_lnum||'
                              MAXVALUE '||seq_max||'
                              MINVALUE '||seq_min||'
                              '||seq_cycle||'
                              '||seq_cache||'
                              '||seq_order||';');
write_out(seq_string);
   END LOOP;
   CLOSE seq_cursor;
END;
/
column dbname new_value db noprint
select value dbname from v$parameter where name='db_name';
spool rep_out\&db\crt_seqs.sql
break on downer skip 1
col text format a60 word_wrap
col downer noprint
select    grantor_owner downer,text
from     seq_temp
```

```
order by downer
/
spool off
drop table seq_temp;
 set termout on verify on feedback on
prompt Finished build
exit
```

Example output from the above script follwos:

```
CREATE SEQUENCE DSPTDBA.ACTSEQ
INCREMENT BY 1
START WITH 27
MAXVALUE 999999999999999999999999999
MINVALUE 1
NOORDER
CACHE 20
NOCYCLE;

CREATE SEQUENCE DSPTDBA.CSPCSEQ
INCREMENT BY 1
START WITH 217094
MAXVALUE 999999999999999999999999999
MINVALUE 0
NOORDER
CACHE 20
NOCYCLE;
```

The 9's can be removed by eliminating the MAXVALUE specification, allowing it to default to the maximum value allowed.

Soft documentation for sequences can be generated using db_seqs.sql from Appendix A.

Documenting Packages, Package Bodies, Procedures and Functions

Under Oracle 7 PLSQL constructs can be stored in the database as stored objects. The lowest level of objects are procedures and functions. The procedures and functions in an Oracle 7 database can be grouped by application or related function into packages (and associated package bodies). All of the DDL to define PLSQL stored objects such as packages, package bodies, procedures and functions is stored in the DAB_SOURCE

view and additional information on them stored in the DBA_OBJECTS view. By creating a script which joins these two tables we can re-create any of the objects mentioned above. The scripts shown below could more readily be done in PLSQL, but was deliberately done in SQL and SQLPLUS to show that it can be done. The first script is the script which does the actual work of selecting the text from the database and reconstructing the text into a command, it is called fprc_rct.sql:

```
REM
REM NAME:      FPRC_RPT.SQL
REM
REM FUNCTION: Build a script to re-create functions, procedures,
REM           packages or package bodies.
REM
REM
set termout off verify off feedback off lines 132 pages 0 heading off
set recsep off space 0
column text format a79
column line noprint
select 'create or replace '||text,line
from
       dba_source
where
       owner = upper('&&1') and
       type = upper('&&2') and
       name = upper('&&3') and
       line = 1;
select text,line
from
       dba_OBJECTS s1,
       dba_source s2
where
       s1.OBJECT_type = upper('&&2') and
       s1.owner = upper('&&1') and
       s1.object_name = upper('&&3') and
       s1.OBJECT_type = s2.type and
       s1.owner = s2.owner and
       s1.OBJECT_NAME = s2.name and
       line > 1
order by
       2;
```

This script can be run standalone by calling it with the values for owner, object type and object name. However, it was designed to be called by a reiterative program called do_fprc.sql which is created by a script run_fprc.sql, shown below:

```
REM
REM NAME               : RUN_FPRC.SQL
REM FUNCTION           : Generate and execute the crt_fprc.sql procedure
REM USE                : Document the procedures and packages and functions
REM                for a user or users
REM Limitations        : Must have access to dba_source and dba_objects.
rem                The FPRC_RCT.SQL procedure must be in same directory
REM
column dbname new_value db noprint
pause Use % for a wildcard - Press enter to continue
accept owner prompt 'Enter object owner:'
accept type  prompt 'Enter object type:'
accept name  prompt 'Enter object name:'
prompt Working....
set echo off heading off verify off feedback off
select value dbname from v$parameter where name='db_name';
spool rep_out\&db\do_fprc.sql
select unique('start fprc_rct '||owner||' '||'"'||type||'"'||' '||name)
from
       dba_source
where
       owner like upper('&owner') and
       type like upper('&type') and
       name like upper('&name');
spool off
set termout off
spool rep_out\&db\crt_fprc.sql
start rep_out\&db\do_fprc.sql
spool off
exit
```

This script generates a sequence of calls to fprc_rct.sql:

```
start fprc_rct CSPCDBA "PROCEDURE" ADDRMANOTE
start fprc_rct CSPCDBA "PROCEDURE" ENTERRULES
start fprc_rct CSPCDBA "PROCEDURE" ESCALATE
start fprc_rct CSPCDBA "PROCEDURE" GETDETAILDATA
start fprc_rct CSPCDBA "PROCEDURE" GETLEAST
start fprc_rct CSPCDBA "PROCEDURE" GETMAXACTSEQNBR
start fprc_rct CSPCDBA "PROCEDURE" GETSUMMARYHEADERS
start fprc_rct CSPCDBA "PROCEDURE" GET_PAWS0001
start fprc_rct CSPCDBA "PROCEDURE" INSADMINAREA
start fprc_rct CSPCDBA "PROCEDURE" INSAFFLNROL
start fprc_rct CSPCDBA "PROCEDURE" INSAUTSYSAREA
start fprc_rct CSPCDBA "PROCEDURE" INSCO
start fprc_rct CSPCDBA "PROCEDURE" INSCOORG
start fprc_rct CSPCDBA "PROCEDURE" INSCOORGAREA
start fprc_rct CSPCDBA "PROCEDURE" INSCOORGAUTH
start fprc_rct CSPCDBA "PROCEDURE" INSEMP
start fprc_rct CSPCDBA "PROCEDURE" INSERTTMPTABLE
start fprc_rct CSPCDBA "PROCEDURE" INSINDIV
start fprc_rct CSPCDBA "PROCEDURE" INSINDIVAUTH
start fprc_rct CSPCDBA "PROCEDURE" INSORGEMP
start fprc_rct CSPCDBA "PROCEDURE" INSSUM
start fprc_rct CSPCDBA "PROCEDURE" PAWS1
start fprc_rct CSPCDBA "PROCEDURE" POSTERROR
```

The script run_fprc.sql then executes the do_fprc.sql script spooling output to the crt_fprc.sql script. The crt_fprc.sql script contains the commands to rebuild the specified object or objects:

```
create or replace PROCEDURE AddRmaNote (
       hv_eventid     IN event.evntid%TYPE,
       hv_actseqnbr   IN act.actseqnbr%TYPE,
       hv_user        IN INDIV.CUIDNBR%TYPE,
       hv_notes       IN VARCHAR2,
       hv_status      IN OUT NUMBER
       )
AS
       leid    NUMBER(10);
       date1   DATE;
BEGIN
       hv_status := 0;
       leid := 0;
       BEGIN
       SELECT FK_LELEID INTO leid FROM INDIV WHERE ( CUIDNBR = hv_user );
       EXCEPTION WHEN OTHERS THEN
          hv_status := 1;
       END;
       SELECT SYSDATE INTO date1 FROM dual;
       IF ( leid > 0 ) THEN
       INSERT INTO actvy_rmk ( ACTVYRMKDT, ACTVYRMKTXT,
FK_ACTFK_EVENTEVNT,
              FK_ACTACTSEQNBR, FK_INDIVFK_LELEID )
       VALUES ( date1, hv_notes, hv_eventid, hv_actseqnbr, leid );
       END IF;
       COMMIT;
END AddRmaNote;
/
.
.
.
create or replace PROCEDURE wkgrp_usr (
       hv_wrkgrp      IN COORG.COORGCD%TYPE,
       hv_grptyp      IN COORG.COORGTYPCD%TYPE,
       hv_status      IN OUT NUMBER
       )
AS
   CURSOR wkgrpusr_cur is
        SELECT indiv.CUIDNBR, indiv.INDIVLASTNM, indiv.INDIVFRSTNM
   FROM orgemp, emp, afflnrol, le, indiv
   WHERE (
        indiv.FK_LELEID = le.leid AND
        le.letypcd = 'EMP' AND
        afflnrol.FK_LELEID = le.leid AND
        emp.FK_AFFLNROLAFFLNRO = afflnrol.AFFLNROLCD AND
        emp.FK_AFFLNROLFK_AFFL = afflnrol.FK_AFFLNAFFLNID AND
        emp.FK_AFFLNROLFK_LELE = afflnrol.FK_LELEID AND
        orgemp.FK_EMPEMPSSN = emp.EMPSSN AND
        orgemp.FK_COORGCOORGTYPCD = hv_grptyp AND
        orgemp.FK_COORGCOORGCD = hv_wrkgrp
        );
   id   indiv.CUIDNBR%TYPE;
```

```
   fnm indiv.INDIVFRSTNM%TYPE;
   lnm indiv.INDIVLASTNM%TYPE;
BEGIN
   hv_status := 0;
   BEGIN
   OPEN wkgrpusr_cur;
   FETCH wkgrpusr_cur INTO id, lnm, fnm;
   WHILE NOT wkgrpusr_cur%NOTFOUND LOOP
   INSERT INTO user_table (cuid,lastnm,frstnm) VALUES (id,lnm,fnm );
   FETCH wkgrpusr_cur INTO id, lnm, fnm;
   END LOOP;
   CLOSE wkgrpusr_cur;
   END;
   COMMIT;
END wkgrp_usr;
```

The crt_fprc.sql script, with minor editing if the developers tended to string commands over several lines, can then be executed to rebuild whichever object or objects it was invoked to recreate. If the DBA doesn't mind editing the crt_fprc.sql the code for the packages, package bodies, procedures or functions can be modified to be more readable and to adhere to any coding standards. In shops where many developers may be working on a single project, the diversity of coding techniques can be astounding, this allows the DBA to enforce some semblance of order on the code in the database.

Soft documentation on object status can be generated using the db_obj.sql script in Appendix A.

Documenting Triggers

Triggers are used to enforce referential integrity constraints, enforce snapshot logic, provide updates on calculated table values and a number of other database functions. The triggers required for constraint enforcement and snapshot upkeep are automatically generated by the Oracle kernel. The following script will recreate the triggers in the database. It is suggested that all triggers created by other than Oracle processes be preceded by some unique set of characters so that the script can be made more selective to ignore the Oracle created triggers.

```
REM  trig_rct.sql
REM
REM    FUNCTION: SCRIPT FOR RE-CREATING DATABASE TRIGGERS
REM
REM           This script can be run by anyone with access to dba_ views
REM
REM              This script is intended to run with Oracle7.
REM
REM              Running this script will in turn create a script to
REM              build all the triggers in the database.  This created
REM              script is called 'crt_trig.sql'.
REM
REM              Only preliminary testing of this script was performed.
REM              Be sure to test it completely before relying on it.
REM
REM M. Ault 3/29/96 TRECOM
REM
set verify off feedback off termout off echo off pages 0 long 4000

set termout on
select 'Creating trigger build script...' from dual;
set termout off

create table trig_temp (owner varchar2(30),
                        trigger_name varchar2(30),
                        trigger_type varchar2(16),
                        triggering_event varchar2(26),
                        table_owner varchar2(30),
                        table_name varchar2(30),
                        referencing_names varchar2(87),
                        when_clause varchar2(2000),
                        trigger_body long,
                        trigger_columns varchar2(400)) ;

DECLARE
   CURSOR trig_cursor IS select owner,
                                trigger_name,
                                trigger_type ,
                                triggering_event,
                                'on '||table_owner,
                                table_name,
                                referencing_names,
                                'when '||when_clause,
                                trigger_body
                        from dba_triggers
                        where owner not in ('SYS','SYSTEM')
                        order by owner;
       cursor trig_col is select trigger_owner,
                                trigger_name,
                                column_name
                        from dba_trigger_cols
                        where trigger_owner not in ('SYS','SYSTEM')
                        order by trigger_owner, trigger_name;
       trig_owner               dba_triggers.owner%TYPE;
       trig_name                dba_triggers.trigger_name%TYPE;
       trig_type                dba_triggers.trigger_type%TYPE;
       trig_event               dba_triggers.triggering_event%TYPE;
       trig_towner              dba_triggers.table_owner%TYPE;
```

```
        trig_tname              dba_triggers.table_name%TYPE;
        trig_rnames             dba_triggers.referencing_names%TYPE;
        trig_wclause            dba_triggers.when_clause%TYPE;
        trig_body               dba_triggers.trigger_body%TYPE;
        trig_col_own            dba_trigger_cols.trigger_owner%TYPE;
        trig_col_nam            dba_trigger_cols.trigger_name%TYPE;
        trig_column             dba_trigger_cols.column_name%TYPE;
        all_columns             varchar2(400);
        counter                 integer:=0;

BEGIN
   OPEN trig_cursor;
   LOOP
      FETCH trig_cursor INTO    trig_owner,
                              trig_name,
                              trig_type,
                              trig_event,
                              trig_towner,
                              trig_tname,
                              trig_rnames,
                              trig_wclause,
                              trig_body;
      EXIT WHEN trig_cursor%NOTFOUND;
        all_columns :='';
        counter := 0;
        open trig_col;
        loop
                fetch trig_col into     trig_col_own,
                                      trig_col_nam,
                                      trig_column;
        exit when trig_col%NOTFOUND;
                if trig_owner = trig_col_own and trig_name=trig_col_nam
                then
                        counter := counter+1;
                        if counter = 1 then
                        all_columns := ' of '||all_columns||trig_column;
                        else
                        all_columns := all_columns||', '||trig_column;
                        end if;
                end if;
        end loop;
        close trig_col;
        if trig_rnames = 'REFERENCING NEW AS NEW OLD AS OLD' then
                trig_rnames := '';
        end if;
        if trig_wclause = 'when ' then
                trig_wclause := '';
        end if;
        insert into trig_temp values (trig_owner,
                              trig_name,
                              trig_type,
                              trig_event,
                              trig_towner,
                              trig_tname,
                              trig_rnames,
                              trig_wclause,
                              trig_body,
                              all_columns);
```

```
   END LOOP;
   CLOSE trig_cursor;
   commit;
END;
/

spool rep_out\crt_trgs.sql
set heading off
set recsep off

select 'create trigger '||owner||'.'||trigger_name||'
'||decode(trigger_type,'BEFORE EACH ROW','BEFORE ',
               'AFTER EACH ROW','AFTER
',trigger_type)||triggering_event||'
'||trigger_columns||'
'||table_owner||'.'||table_name||'
'||referencing_names||'
'||decode(trigger_type,'BEFORE EACH ROW','ON EACH ROW',
               'AFTER EACH ROW','ON EACH ROW','')||'
'||when_clause,
trigger_body,' '||'
/'||'
'
from trig_temp
order by owner;
spool off
drop table trig_temp;
exit
```

The output from this script may require more editing than the other scripts shown thus far. An example of its output follows:

```
create trigger CSPCDBA.ACTPER_BEFORE_INSERT
BEFORE INSERT OR UPDATE
 of FK_AUTOSYSAUTOSYSC, FK_AUTOSYSAUTOSYSV
on CSPCDBA.ACTPER
ON EACH ROW
when     NEW.FK_AUTOSYSAUTOSYSV IS NOT NULL
        AND NEW.FK_AUTOSYSAUTOSYSC IS NOT NULL
DECLARE
    DUMMY INTEGER;
/
  BEGIN
    DUMMY := 0;
    SELECT COUNT(*) INTO DUMMY
       FROM AUTOSYS
       WHERE   AUTOSYS.AUTOSYSVRSNNBR = :NEW.FK_AUTOSYSAUTOSYSV
               AND AUTOSYS.AUTOSYSCD = :NEW.FK_AUTOSYSAUTOSYSC;
    IF DUMMY = 0 THEN
        RAISE_APPLICATION_ERROR(-20000, 'INVALID FOREIGN KEY');
    END IF;
END;

create trigger CSPCDBA.ACTRSLTRSPTYP_BEFORE_INSERT
BEFORE INSERT OR UPDATE
 of     FK_AUTSYSRSLTTYFK, FK_AUTSYSRSLTTYFK0,
```

```
        FK_AUTSYSRSLTTYFK1, FK_AUTSYSRSLTTYFK2
on CSPCDBA.ACTRSLTRSPTYP
ON EACH ROW
when    NEW.FK_AUTSYSRSLTTYFK IS NOT NULL
        AND NEW.FK_AUTSYSRSLTTYFK0 IS NOT NULL
        AND NEW.FK_AUTSYSRSLTTYFK1 IS NOT NULL
        AND NEW.FK_AUTSYSRSLTTYFK2 IS NOT NULL
DECLARE
    DUMMY INTEGER;
/
  BEGIN
    DUMMY := 0;
    SELECT COUNT(*) INTO DUMMY
       FROM AUTSYSRSLTTYP
       WHERE
        AUTSYSRSLTTYP.FK_RSLTTYPRSLT_CD = :NEW.FK_AUTSYSRSLTTYFK
        AND AUTSYSRSLTTYP.FK_RSLTTYPRSLTTYPC = :NEW.FK_AUTSYSRSLTTYFK0
        AND AUTSYSRSLTTYP.FK_AUTOSYSAUTOSYSC = :NEW.FK_AUTSYSRSLTTYFK1
        AND AUTSYSRSLTTYP.FK_AUTOSYSAUTOSYSV = :NEW.FK_AUTSYSRSLTTYFK2;
    IF DUMMY = 0 THEN
        RAISE_APPLICATION_ERROR(-20000, 'INVALID FOREIGN KEY');
    END IF;
END;
```

The script should be run after all other trigger building events in the database have completed. use of the CREATE rather than the CREATE OR REPLACE command should provide some measure of safety that existing triggers will not be harmed by running the script.

Soft documentation of database objects can be generated with db_obj.sql in Appendix A.

Documenting Database Views

The final set of stored objects that will be documented are database views. Views are used to pre-join tables, provide security or make it easier for non-sql literate users to get at complex data. Often times, a developer may create a view and then forget to document it. This script should provide some measure of documentation and allow recreation of database views:

```
REM
REM NAME              :rct_vw.sql
REM FUNCTION   :recreate database views by owner
REM USE                       :Generate ddl for database views
REM Limitations :If your views are greater than 5000 characters
```

```
REM                    then increase the set long. This can be determined
by
REM                    querying the DBA_VIEWS table's text_length column
REM                    for the max value: select max(text_length) from
REM                                       dba_views;
REM
set pages 59 lines 79 feedback off echo off
column text            format a80
column view_name       format a20
set long 5000 heading off
column dbname new_value db noprint
select value dbname from v$parameter where name='db_name';
spool rep_out\&db\crt_view
select
       'REM Code for view: '||v.view_name||'
CREATE OR REPLACE VIEW '||v.owner||'.'||v.view_name||' as
 ',
       v.text
from dba_views v
where v.owner like upper('%&&owner_name')
order by
       v.view_name;
spool off
Pause Press enter to continue
exit
```

Example output for two of the ALL_ views:

```
REM Code for view: ALL_CATALOG

CREATE OR REPLACE VIEW SYS.ALL_CATALOG as
select u.name, o.name,
decode(o.type, 0, 'NEXT OBJECT', 1, 'INDEX', 2, 'TABLE', 3, 'CLUSTER',
4, 'VIEW', 5, 'SYNONYM', 6, 'SEQUENCE', 'UNDEFINED')
from sys.user$ u, sys.obj$ o
where o.owner# = u.user#
and o.type in (2, 4, 5, 6)
and o.linkname is null
and (o.owner# in (userenv('SCHEMAID'), 1)   /* public objects */
or
obj# in ( select obj#  /* directly granted privileges */
from sys.objauth$
where grantee# in ( select kzsrorol
from x$kzsro
)
)
or
(
o.type in (7, 8, 9) /* prc, fcn, pkg */
and
exists (select null from v$enabledprivs
where priv_number = -144 /* EXECUTE ANY PROCEDURE */)
)
or
(
o.type in (2, 4, 5) /* table, view, synonym */
```

```
and
exists (select null from v$enabledprivs
where priv_number in (-45 /* LOCK ANY TABLE */,
-47 /* SELECT ANY TABLE */,
-48 /* INSERT ANY TABLE */,
-49 /* UPDATE ANY TABLE */,
-50 /* DELETE ANY TABLE */))
)
or
( o.type = 6 /* sequence */
and
exists (select null from v$enabledprivs
where priv_number = -109 /* SELECT ANY SEQUENCE */)))

REM Code for view: ALL_COL_COMMENTS

CREATE OR REPLACE VIEW SYS.ALL_COL_COMMENTS as
select u.name, o.name, c.name, co.comment$
from sys.obj$ o, sys.col$ c, sys.user$ u, sys.com$ co
where o.owner# = u.user#
and o.type in (2, 4, 5)
and o.obj# = c.obj#
and c.obj# = co.obj#(+)
and c.col# = co.col#(+)
and (o.owner# = userenv('SCHEMAID')
or o.obj# in
(select obj#
from sys.objauth$
where grantee# in ( select kzsrorol
from x$kzsro

)
)
or
exists (select null from v$enabledprivs
where priv_number in (-45 /* LOCK ANY TABLE */,
-47 /* SELECT ANY TABLE */,
-48 /* INSERT ANY TABLE */,
-49 /* UPDATE ANY TABLE */,
-50 /* DELETE ANY TABLE */))
)
```

Try running the above documentation script against the DBA_ series of views, a DBA can learn a lot from looking at the way those views are constructed.

Soft documentation on view status can be obtained by use of the db_obj.sql and db_view.sql scripts in Appendix A.

Snap Shot and Snap Shot Log Documentation

At the current time there are no scripts to re-generate existing snapshots, there are however report scripts that will document snapshots and snapshot logs in the instance.

Script to Document Snapshot logs:

```
rem
rem Name:             snap_log_rep.sql
rem FUNCTION:  Report on database Snapshot Logs
rem Use:              From an account that accesses DBA_ views
rem M. Ault TRECOM
rem
set pages 56 lines 130 feedback off
start title132 "Snapshot Log Report"
spool rep_out\&db\db_snplg
rem
column log_owner       format a10     heading "Owner"
column master  format a20     heading "Master"
column log_table       format a20     heading "Snapshot"
column log_trigger     format a20     heading "Trigger Name"
column current_snapshots              heading "Last Refresh"
rem
select
        log_owner,
        master,
        log_table,
        log_trigger,
        current_snapshots
from
dba_snapshot_logs
order by 1;
rem
spool off
pause Press enter to continue
exit
```

Script to Document Snapshots:

```
rem
rem Name:      snap_rep.sql
rem FUNCTION:  Report on database Snapshots
rem Use:       From an account that accesses DBA_ views
rem
set pages 56 lines 130 feedback on
```

```
rem
column owner   format a10     heading "Owner"
column snapshot        format a20     heading "Snapshot"
column source  format a20     heading "Source Table"
column link    format a20     heading "Link"
column log                    heading "Use Log?"
column refreshed              heading "Refreshed?"
column error   format a20     heading "Error?"
column type    format a10     heading "Refresh Type"
column started         format a13     heading "Start Refresh"
column next    format a13     heading "Next Refresh"
rem
start title132 "Snapshot Report"
spool rep_out\&db\db_snp
rem
select owner, name||'.'||table_name Snapshot, master_view,
       master_owner||'.'||master Source, master_link Link,
       can_use_log Log, last_refresh Refreshed, start_with started,
       error, type , next,
       query
from dba_snapshots
order by 1,3,5;
rem
spool off
pause Press enter to continue
exit
```

Documenting Database Links

Many databases will be distributed in nature. This distributed attribute requires that the database have pre-defined linkages to other databases. These linkages are known as database links. The following script allows re-creation of database links for a specific database:

```
REM
REM link_rct.sql
REM
REM    FUNCTION: SCRIPT FOR RE-CREATING DATABASE LINKS
REM
REM              This script must be run by 'SYS'.
REM
REM              This script is intended to run with Oracle7.
REM
REM Running this script will in turn create a script to build all the
REM database links in the database.  This created script is called
REM 'crt_dbls.sql'.
REM
REM Since a DBA cannot create a private database link for a user,
REM this script will contain various connect clauses before each create
REM statement.  In order for the database links to be created under
REM the correct schema, it must connect as that individual.  Therefore,
```

```
REM before using the script, you must add each user's password to the
REM connect clause.  Duplicate connect clauses can be eliminated being
REM sure that the database link is created under the correct schema.
REM
REM The PUBLIC database links will cause connect as 'SYS'. However, this
REM username can be changed to any user with the DBA role or with the
REM 'CREATE PUBLIC DATABASE LINK' system privilege.
REM
REM The spooled output is ordered by the link owner, a PUBLIC database
REM link has 'PUBLIC' as it's owner.
REM
REM Only preliminary testing of this script was performed. Be sure to
REM test it completely before relying on it.
REM

set verify off
set feedback off
set termout off
set echo off
set pagesize 0

set termout on
select 'Creating database link build script...' from dual;
set termout off

create table dl_temp (lineno NUMBER, grantor_owner varchar2(20),
text VARCHAR(255))
TABLESPACE TEMP01;

DECLARE
   CURSOR link_cursor IS select   u.name,
                                  l.name,
                                  l.userid,
                                  l.password,
                                  l.host
                          from    sys.link$ l, sys.user$ u
                          where   l.owner# = u.user#
                          order by l.name;
   lv_owner    sys.user$.name%TYPE;
   lv_db_link  sys.link$.name%TYPE;
   lv_username sys.link$.userid%TYPE;
   lv_password sys.link$.password%TYPE;
   lv_host     sys.link$.host%TYPE;
   lv_string     VARCHAR2(255);
   lv_user       VARCHAR2(255);
   lv_connect    VARCHAR2(255);
   lv_text       VARCHAR2(500);

   procedure write_out(p_string VARCHAR2) is
   begin
      insert into dl_temp (grantor_owner,text)
                       values (lv_owner,p_string);
   end;

BEGIN
   OPEN link_cursor;
   LOOP
      FETCH link_cursor INTO lv_owner,
```

```
                              lv_db_link,
                              lv_username,
                              lv_password,
                              lv_host;
      EXIT WHEN link_cursor%NOTFOUND;
if (lv_owner = 'PUBLIC') then
      lv_string := ('CREATE PUBLIC DATABASE LINK '||
                lower(replace(lv_db_link,'.WORLD','')));
else
      lv_string := ('CREATE DATABASE LINK '||
                lower(replace(lv_db_link,'.WORLD','')));
end if;
      if (lv_username is not null) then
         lv_user := ('CONNECT TO '||lower(lv_username)||
                  ' IDENTIFIED BY '||lower(lv_password));
      end if;
      if (lv_host is not null) then
         lv_connect := ('USING '''||lv_host||''''||';');
      end if;
   lv_text := lv_string || ' ' || lv_user || ' ' || lv_connect;
   write_out(lv_text);
   lv_user := ' ';
   lv_connect := ' ';
   END LOOP;
   CLOSE link_cursor;
END;
/
 column dbname new_value db noprint
select value dbname from v$parameter where name='db_name';
spool rep_out\&db\crt_dbls.sql
break on downer skip 1
col text format a60 word_wrap

select   'connect ' || decode (grantor_owner, 'PUBLIC', 'SYS',
grantor_owner)
|| '/'|| decode (grantor_owner, 'PUBLIC', 'SYS', grantor_owner) downer,'
'||rtrim(text)
from     dl_temp
order by downer
/
spool off

drop table dl_temp;

exit
```

An edited example of the output from this script is shown below.

```
connect SYSTEM/SYSTEM

CREATE DATABASE LINK orcspcd2 CONNECT TO cspcdba2 IDENTIFIED BY notit
USING 't:90.37.190.32:ORCSPCD2';
CREATE DATABASE LINK orcspcd3 CONNECT TO dsptdba IDENTIFIED BY notit USING
't:90.32.44.48:ORCSPCD3';
CREATE DATABASE LINK ordsptt1 CONNECT TO system IDENTIFIED BY notit USING
't:90.93.232.17:ORDSPTT1';
```

In Conclusion

In this presentation techniques for documenting existing Oracle 7 Instances have been presented. Virtually every aspect of the Oracle database has been covered with the exception of items relating to recreation of snapshots and snapshot logs. By using these techniques a DBA can create a series of scripts that allow documentation and if required, recreation of their Instance or Instances.

Appendix A
Soft Documentation Scripts

```
rem
rem datafile.sql
rem FUNCTION: Report on Database datafiles
rem USE: From SQLPLUS
rem
rem M. Ault 11/23/95 TRECOM
rem
column file_name              format a65            heading 'File|Name'
column file_id                format 999999  heading 'File|Number'
column tablespace_name        format a10            heading 'Tablespace'
column bytes          format 9,999.99       heading 'Megs'
column status                               heading 'File|Status'
column value   new_value block_size   noprint
rem
select
       value
from
       sys.v_$parameter
where
       name='db_block_size';
rem
column  blocks  format 99,999 heading '&block_size byte|Blocks'
set feedback off echo off lines 132 pages 59
start title132 'File Storage Statistics Report'
spool rep_out\&db\db_files
select
       file_name,
       file_id,
       status,
       tablespace_name,
       bytes/(1024*1024) bytes,
       blocks
from
       sys.dba_data_files
order by
       file_id
/
spool off
pause press enter/return to continue
clear columns

rem db_logs.sql
rem FUNCTION: Report on Redo Logs Physical files
rem MRA TRECOM 13-Oct-1995
rem
column group#  format 999999
column member  format a50
set lines 80 pages 60 feedback off verify off
```

```
start title80 'Redo Log Physical Files'
break on group#
spool rep_out\&db\db_redo
select * from sys.v_$logfile
order by group#
/
spool off
pause Press enter to continue
exit
```

```
rem Name: log_stat.sql
rem FUNCTION: Provide a current status for redo logs
rem MRA TRECOM 13-Oct-95
rem
column first_change#   format 9999999 heading Change#
column group#          format 9,999 heading Grp#
column thread#                format 999 heading Th#
column sequence#              format 9,999 heading Seq#
column members                format 999 heading Mem
column archived               format a4 heading Arc?
break on thread#
set pages 60 lines 80 feedback off
start title80 'Current Redo Log Status'
spool rep_out\&db\db_logst
select
       thread#,
       group#,
       sequence#,
       bytes,
       members,
       archived,
       status,
       first_change#,
       first_time
 from
       sys.v_$log
order by
       thread#,
       group#;
pause Press Enter to continue
exit
```

```
rem FUNCTION: Provide data on Redo Log Threads
rem
rem name: db_thrd.sql
rem MRA 10-20-95 TRECOM
rem
column current_group#         heading Current|Group#
column Checkpoint_change#     heading Checkpoint|Change#
column checkpoint_time                heading Checkpoint|Time
column open_time                      heading Open|Time
column thread#                        heading Thread#
column status                 heading Status
column enabled                        heading Enabled
column groups                 heading Groups
column Instance                       heading Instance
column sequence#                      heading Sequence#
set lines 132 pages 59
start title132 'Redo Thread Report'
spool rep_out\&db\db_thds
select *
from
       sys.v_$thread
order by
       thread#;
spool off
pause Press enter to continue
exit
```

```
REM
REM NAME               : DB_PARM.SQL
REM FUNCTION   : GENERATE LIST OF DB PARAMETERS
REM USE        : GENERAL
REM Limitations        : None
REM Revisions:
REM            Date           Modified By           Reason For change
REM            11/24/93       M. Ault        Initial Creation
SET NEWPAGE 0 VERIFY OFF
SET ECHO OFF termout off PAGES 58 lines 131
column num             format 999
column type    format 9999
column name    format a37
column value   format a30
COLUMN ISDEFAULT       FORMAT A8 HEADING "Default"
start title132 "INIT.ORA PARAMETER LISTING"
DEFINE OUTPUT = rep_out\&db\db_parm
SPOOL &OUTPUT
SELECT * FROM V$PARAMETER;
SPOOL OFF
CLEAR COLUMNS
pause Press enter to continue
exit
```

```
rem Name: free_spc.sql
rem FUNCTION: Provide data on tablespace extent status
rem REQUIREMENTS: The free_space view must exist
SET FEED OFF
SET FLUSH OFF
SET VERIFY OFF
set pages 58 LINES 130
column tablespace      heading Name   format a30
column file_id                heading File#  format 99999
column pieces          heading Frag   format 9999
column free_bytes      heading 'Free Byte'
column free_blocks     heading 'Free Blk'
column largest_bytes   heading 'Biggest Bytes'
column largest_blks    heading 'Biggest Blks'
column ratio           heading 'Percent' format 999.999
start title132 "FREE SPACE REPORT"
define 1 = rep_outp\&db\db_fspc
spool &1
select
        tablespace,
        count(*) files,
        sum(pieces) pieces,
        sum(free_bytes) free_bytes,
        sum(free_blocks) free_blocks,
        sum(largest_bytes) largest_bytes,
        sum(largest_blks) largest_blks,
        sum(largest_bytes)/sum(free_bytes)*100 ratio
from
        free_space
group by
        tablespace;
spool off
clear columns
pause Press Enter to Continue
exit
```

```
rem Name       : fs_view.sql
rem FUNCTION: Create free_space view for use by freespc reports
create view free_space
           (tablespace,
              file_id,
              pieces,
              free_bytes,
              free_blocks,
              largest_bytes,
              largest_blks) as
select
       tablespace_name,
       file_id, count(*),
       sum(bytes),
       sum(blocks),
       max(bytes),
       max(blocks)
from
       sys.dba_free_space
group by
       tablespace_name,
       file_id;
```

```
REM
REM db_tbsp.sql
REM
REM FUNCTION: Generate a report of Tablespace Defaults
REM USE: From SQLPLUS
REM
REM M. Ault TRECOM 4/1/96
REM
column tablespace name format a15 heading 'Tablespace|Name'
column initial format 9,999,999,999 heading 'Initial|Extent(Bytes)'
column next format 9,999,999,999 heading 'Next|Extent(Bytes)'
column minextents format 999 heading 'Minimum|Extents'
column maxextents format 999 heading 'Maximum|Extents'
column pctincrease format 999 heading 'Percent|Increase'
column status format a15 heading 'Status'
set pages 60 lines 80 feedback off verify off
start title80 "Tablespace Defaults Report"
spool rep_out/&db/db_tbsp
select
       tablespace_neme,
       initial,next ,
       minextents,
       maxextents,
       pctincrease,
       status
from
       dba_tablespaces
order by
       tablespace_name;
spool off
set feedback on verify on
pause Press enter to continue
Exit
```

```
REM NAME               :RBK1.SQL
REM FUNCTION   :REPORT ON ROLLBACK SEGMENT STORAGE
REM USE                :FROM SQLPLUS
REM Limitations        : None
REM
COLUMN NAME            FORMAT A10            HEADING 'ROLLBACK'
COLUMN EXTENTS                FORMAT 9999999        HEADING 'CUR EXTENTS'
COLUMN OPTSIZE                FORMAT 9999999        HEADING 'OPTL SIZE'
COLUMN SHRINKS                FORMAT 9999999        HEADING 'SHRINKS'
COLUMN AVESHRINK              FORMAT 999999999      HEADING 'AVE SHRINK'
COLUMN AVEACTIVE              FORMAT 999999999      HEADING 'AVE TRANS'
rem
SET FEEDBACK OFF VERIFY OFF lines 80 pages 58
@title80 "ROLLBACK SEGMENT STORAGE"
SPOOL rep_out\&db\db_rbks1
rem
SELECT *
FROM ROLLBACK1
ORDER BY NAME;
SPOOL OFF
pause Press enter to continue
exit
```

```
REM NAME               :RBK2.SQL
REM FUNCTION   :REPORT ON ROLLBACK SEGMENT STATISTICS
REM USE                :FROM SQLPLUS
REM Limitations : None
REM
COLUMN NAME    FORMAT A10            HEADING 'ROLLBACK'
COLUMN EXTENTS         FORMAT 9999999         HEADING 'EXTENTS'
COLUMN XACTS   FORMAT 9999999         HEADING 'TRANS'
COLUMN HWMSIZE         FORMAT 99999999        HEADING 'LARGEST TRANS'
COLUMN RSSIZE  FORMAT 99999999        HEADING 'CUR SIZE'
COLUMN WAITS   FORMAT 99999   HEADING 'WAITS'
COLUMN WRAPS   FORMAT 99999   HEADING 'WRAPS'
COLUMN EXTENDS         FORMAT 9999999         HEADING 'EXTENDS'
rem
SET FEEDBACK OFF VERIFY OFF lines 80 pages 58
rem
@title80 "ROLLBACK SEGMENT STATISTICS"
SPOOL rep_out\&db\db_rbks2
rem
SELECT *
FROM
       ROLLBACK2
ORDER BY
       NAME;
SPOOL OFF
pause Press enter to continue
exit
```

```
REM FUNCTION: create views required for rbk1 and rbk2 reports.
REM
CREATE OR REPLACE VIEW ROLLBACK1 AS
SELECT
        NAME,
        EXTENTS,
        OPTSIZE,
        SHRINKS,
        AVESHRINK,
        AVEACTIVE
FROM
        V$ROLLNAME N,
        V$ROLLSTAT S
WHERE
        N.USN=S.USN;

CREATE OR REPLACE VIEW ROLLBACK2 AS
SELECT
        NAME,
        EXTENTS,
        XACTS,
        HWMSIZE,
        RSSIZE,
        WAITS,
        WRAPS,
        EXTENDS
FROM
        V$ROLLNAME N,
        V$ROLLSTAT S
WHERE
        N.USN=S.USN;

rem
rem db_rbks.sql
rem FUNCTION: Generate a report on database rollback segments
rem USE: From SQLPLUS
rem M. Ault 11/28/95 TRECOM
rem
column rollback_seg    format a10            heading 'Rollback|Segment'
column tablespace      format a10            heading 'Rbks|Tablespace'
column pct                    format 9990    heading 'Pct|Inc'
column init            format 99,999  heading 'Init|Ext'
column next            format 99,999  heading 'Next|Ext'
column min                    format 999            heading 'Min|Ext'
column max                    format 9,999   heading 'Max|Ext'
column instance               format a3             heading 'Int|Num'
column type                                  heading 'Rbks|Type'
column status                                heading 'Rbks|Status'
set pages 59 lines 132 feedback off echo off
start title132 'Rollback Storage Parameter Report'
spool rep_out\&db\db_rbks
select
        segment_name rollback_seg,
        tablespace_name tablespace,
        initial_extent init,
        next_extent next,
        min_extents min,
        max_extents max,
```

```
        pct_increase pct,
        status,
        instance_num Instance,
        decode(owner,'SYS','PRIVATE') type
from
        dba_rollback_segs;
spool off
pause press enter/return to continue
clear columns
exit
```

```
REM NAME              : DB_PROF.SQL
REM FUNCTION   : GENERATE USER PROFILES REPORT
REM USE        : From SQLPLUS
REM
set flush off term off pagesize 58 linesize 78
column profile                format A20     heading Profile
column resource_name                  heading 'Resource:'
column limit           format A15     heading Limit
break on profile
start title80 'ORACLE PROFILES REPORT'
define output = rep_out\&db\db_profs
spool &output
select
       profile,
       resource_name,
       limit
from
       sys.dba_profiles
order by
       profile;
spool off
pause Press enter to continue
exit
```

```
REM
REM NAME               : DB_ROLE.SQL
REM FUNCTION   : GENERATE ROLES REPORT
REM USE        : FROM SQLPLUS
REM Limitations        : None
REM M. Ault TRECOM
REM
rem set flush off term off pagesize 58  linesize 78
column grantee                heading 'User or Role'
column admin_option    heading 'Admin?'
break on grantee
start title80 'ORACLE ROLES REPORT'
define output = rep_out\&db\db_roles
spool &output
select
       grantee,
       privilege,
       admin_option
from
       sys.dba_sys_privs
order by
       grantee;
spool off
pause Press enter to continue
exit
```

```
rem************************************************************************
rem  NAME: db_tgnts.sql
rem
rem  HISTORY:
rem  Date      Who                          What
rem  --------          ------------------- --------------------------------
-
rem  05/24/91          Gary Dodge                    Creation
rem  12/12/92          Michael Brouillette           Allow specification of
a
rem                                                  owner.
rem  05/27/93  Mike Ault                     Updated to ORACLE7
rem
rem  FUNCTION: Produce report of table grants showing GRANTOR, GRANTEE
rem            and specific GRANTS.
rem
rem  INPUTS: Owner name
rem ************************************************************************
rem
COLUMN GRANTEE                FORMAT A18     heading "Grantee"
COLUMN OWNER           FORMAT A18     heading "Owner"
COLUMN TABLE_NAME       FORMAT A30    heading "Table"
COLUMN GRANTOR                FORMAT A18     heading "Grantor"
COLUMN PRIVILEGE              FORMAT A10     heading "Privilege"
COLUMN GRANTABLE       FORMAT A19     HEADING "With Grant Option?"
rem
BREAK ON OWNER SKIP 4 ON TABLE_NAME SKIP 1 on grantee on grantor ON REPORT
REM
SET LINESIZE 130 PAGES 56 VERIFY OFF FEEDBACK OFF
start title132 "TABLE GRANTS BY OWNER AND TABLE"
DEFINE OUTPUT = rep_out\&db\db_tgnts
SPOOL &output
REM
SELECT
        OWNER,
        TABLE_NAME,
        GRANTEE,
        GRANTOR,
        PRIVILEGE,
        GRANTABLE
FROM
        DBA_TAB_PRIVS
WHERE
        OWNER NOT IN ('SYS','SYSTEM')
ORDER BY
        OWNER,
        TABLE_NAME,
        GRANTOR,
        GRANTEE;
REM
SPOOL OFF
pause Press enter to continue
exit
```

```
REM
REM NAME               : DB_USRS.SQL
REM FUNCTION   : GENERATE USER_REPORT
REM USE        : From SQLPLUS
REM Limitations        : None
REM Revisions
REM Date               Modified By    Reason For change
REM 08-Apr-1993        MIKE AULT      INITIAL CREATE
REM
set flush off  term off  pagesize 58  linesize 131
rem
column username                        format a10     heading User
column default_tablespace     format a20     heading "Default Tablespace"
column temporary_tablespace   format a20     heading "Temporary
Tablespace"
column granted_role           format a20     heading Roles
column default_role           format a9      heading Default?
column admin_option           format a7      heading Admin?
column profile                         format a15     heading 'Users
Profile'
rem
start :title132 'ORACLE USER REPORT'
define output = rep_out\&db\db_usrs
break on username skip 1 on default_tablespace on temporary_tablespace on
profile
spool &output
rem
select
        username,
        default_tablespace,
        temporary_tablespace,
        profile,
        granted_role,
        admin_option,
        default_role
from
        sys.dba_users a,
        sys.dba_role_privs b
where
        a.username = b.grantee
order by
        username,
        default_tablespace,
        temporary_tablespace,
        profile,
        granted_role;
rem
spool off
pause Press enter to continue
exit
```

```
REM NAME               : DB_TABS.SQL
REM FUNCTION   : GENERATE TABLE REPORT
REM USE        : FROM SQLPLUS
REM Limitations   : None
clear columns
column table_name                              heading Table
column tablespace_name        format A15       heading Tablespace
column minextents                              heading "Min|Extents"
column maxextents                              heading "Max|Extents"
column pctfree                                        heading "Percent|Free"
column pctused                                        heading "Percent|Used"
column pctincrease                             heading "Percent|Increase"
set lines 132 pages 60 feedback off verify off
START TITLE132 "ORACLE TABLE REPORT"
spool rep_out\&db\db_tabs
select
       owner,
       table_name,
       tablespace_name,
       initial, next,
       minextents,maxextents,
       pctfree, pctused,
       pctincrease
from
       sys.dba_tables
where
       OWNER NOT IN ('SYSTEM', 'SYS')
order by
       tablespace_name, owner;
spool off
clear columns
pause Press enter to continue
EXIT
```

```
rem
rem  NAME: tab_stat.sql
rem  HISTORY:
rem  Date      Who                          What
rem  --------  -------------------    ------------------------------
rem  5/27/93   Mike Ault                    Initial creation
rem
rem  FUNCTION:  Will show table statistics for a user's tables or all rem
           tables.
rem               Statistics must have been run on tables
rem
set pages 56 lines 130 newpage 0 verify off echo off feedback off
rem
column owner           format a12            heading "Table Owner"
column table_name      format a20            heading "Table"
column tablespace_name        format a20            heading "Tablespace"
column num_rows               format 999,999,999 heading "Rows"
column blocks          format 999,999        heading "Blocks"
column empty_blocks    format 999,999        heading "Empties"
column space_full      format 999.99  HEADING "% Full"
column chain_cnt              format 999,999        HEADING "Chains"
column avg_row_len format 99,999,999,999     HEADING "Avg Length (Bytes)"
rem
start title132 "Table Statistics Report"
DEFINE OUTPUT = rep_out\&db\tab_stat
spool &output
rem
BREAK ON OWNER SKIP 2 ON TABLESPACE_NAME SKIP 1;
select
       owner,
       table_name,
       tablespace_name,
       num_rows,
       blocks,
       empty_blocks,
100*((NUM_ROWS*AVG_ROW_LEN)/((GREATEST(blocks,1)+empty_blocks)*2048))
space_full,
       chain_cnt,
       avg_row_len
from
       dba_tables
WHERE
       OWNER NOT IN ('SYS','SYSTEM')
order by
       owner,
       tablespace_name;
spool off
pause Press enter to continue
exit

rem Name: db_clus.sql
rem FUNCTION: Generate a report on database clusters
rem FUNCTION: showing cluster, tablespace, tables, and column data
rem MRA TRECOM 10/23/95
rem
column owner           format a10
column cluster_name    format a15 heading "Cluster"
column tablespace_name        format a20 heading "Tablespace"
```

```
column table_name      format a20 heading "Table"
column tab_column_name         format a20 heading "Table Column"
column clu_column_name         format a20 heading "Cluster Column"
set pages 56 lines 130 feedback off
start title132 "Cluster Report"
break on owner on tablespace_name on cluster_name on table_name
spool rep_out\&db\db_clus
select
       a.owner,
       tablespace_name,
       a.cluster_name,
       table_name,
       tab_column_name,
       clu_column_name
from
       dba_clusters a,
       dba_clu_columns b
where
       a.cluster_name=b.cluster_name
order by 1,2,3,4
/
spool off
pause Press enter to continue
exit
rem Name               :       clu_typ.sql
rem Purpose            :       Report on new DBA_CLUSTER columns
rem Use        :       From an account that accesses DBA_ views
rem
column owner                   format a10     heading "Owner"
column cluster_name            format a15     heading "Cluster"
column tablespace_name         format a10     heading "Tablespace"
column avg_blocks_per_key      format 999999  heading "Blocks per Key"
column cluster_type            format a8      Heading "Type"
column function                format 999999  heading "Function"
column hashkeys                format 99999   heading "# of Keys"
set pages 56 lines 79 feedback off
start title80 "Cluster Type Report"
spool rep_out\&db\clu_type
select
       owner,
       cluster_name,
       tablespace_name,
       avg_blocks_per_key,
       cluster_type,
       function,
       hashkeys
from
       dba_clusters
order by 2
/
spool off
exit

rem Name: clus_siz.sql
rem
rem FUNCTION: Generate a cluster sizing report
rem
column owner                           format a10
```

```
column cluster_name                    format a15                    heading
"Cluster"
column tablespace_name         format a15                    heading
"Tablespace"
column pct_free                        format 999999         heading "%
Free"
column pct_used                        format 999999         heading "%
Used"
column key_size                        format 999999         heading "Key
Size"
column ini_trans                       format 999                    heading
"IT"
column max_trans                       format 999999         heading "Max
Tran"
column initial_extent                  format 999999999      heading
"Initial Ext"
column next_extent                     format 999999999      heading "Next
Ext"
column max_extents                     format 9999                   heading
"Max Ext"
column pct_increase                    format 9999                   heading
"% Inc"
set pages 56 lines 130 feedback off
start title132 "Cluster Sizing Report"
break on owner on tablespace_name
spool rep_out\&db\cls_sze
select
        owner,
        tablespace_name,
        cluster_name,
        pct_free,
        pct_used,
        key_size,
        ini_trans,
        max_trans,
        initial_extent,
        next_extent,
        min_extents,
        max_extents,
        pct_increase
from
        dba_clusters
order by
        1,2,3
/
spool off
pause Press enter to continue
exit
```

```
rem
rem NAME:     pk_fk.sql
rem
rem FUNCTION: Report on all primary key - foreign key relationships
rem           in the database
rem
rem
column for_owner   format a10  heading "Foreign|Owner"
column for_table   format a25  heading "Foreign|Table"
column pri_owner   format a10  heading "Primary|Owner"
column pri_table   format a25  heading "Primary|Table"
column for_col     format a25  heading "Foreign|Column"
column pri_col     format a25  heading "Primary|Column"
accept owner prompt 'Enter Table Owner:'
break on for_owner on for_table on pri_owner
start title132 "Primary-Foreign Key Report"
set lines 131 pages 59 verify off feedback off
spool rep_out\&db\pk_fk
select
       a.owner for_owner,
       a.table_name for_table,
       c.column_name for_col,
       b.owner pri_owner,
       b.table_name pri_table,
       d.column_name pri_col
from
       dba_constraints a,
       dba_constraints b,
       dba_cons_columns c,
       dba_cons_columns d
where
       a.r_constraint_name=b.constraint_name
       and a.constraint_type = 'R'
       and b.constraint_type = 'P'
       and a.owner=upper('&owner')
       and a.r_owner = b.owner
       and a.constraint_name = c.constraint_name
       and b.constraint_name = d.constraint_name
       and a.owner = d.owner
       and a.table_name = c.table_name
       and b.owner = d.owner
       and b.table_name = d.table_name
order by
       1,2,4;
spool off
pause press enter to continue
exit
```

```
rem  ****************************************************************
rem  NAME:     db_inx.sql
rem  FUNCTION: Generate report on Indexes.
rem  INPUTS:
rem          1 = Oracle Account
rem          2 = Table name
rem  ****************************************************************
SET HEADING OFF VERIFY OFF PAUSE OFF
PROMPT **  Index Column Report  **
PROMPT
PROMPT Percent signs are wild
ACCEPT OWNER CHAR PROMPT 'Enter Oracle account to report on (or wild): ';
ACCEPT TABLE_NAME CHAR PROMPT 'Enter table name to report on (or wild): ';
PROMPT
PROMPT Report file name is  DB_INX.LIS
SET HEADING ON
SET LINESIZE 130 PAGESIZE 56 NEWPAGE 0 SPACE 1 TAB OFF
SET TERMOUT OFF
BREAK ON TABLE_OWNER SKIP PAGE ON TABLE_TYPE ON TABLE_NAME
ON UNIQUENESS ON INDEX_NAME SKIP 1
COLUMN TABLE_OWNER            FORMAT A30     HEADING 'Object Owner'
COLUMN TABLE_TYPE      FORMAT A6      HEADING 'Type'
COLUMN TABLE_NAME             FORMAT A30     HEADING 'Object Name'
COLUMN INDEX_NAME             FORMAT A30     HEADING 'Index Name'
COLUMN UNIQUENESS             FORMAT A1      HEADING 'U|N|I|Q|U|E'
COLUMN COLUMN_NAME     FORMAT A30     HEADING 'Column Name'
START TITLE132 "Index Columns by Owner and Table Name"
SPOOL  rep_out\&db\db_inx
SELECT
       I.TABLE_OWNER
      , DECODE(TABLE_TYPE,'CLUSTER','CLUSTR','TABLE') TABLE_TYPE
      , I.TABLE_NAME
        , I.INDEX_NAME
      , SUBSTR(UNIQUENESS,1,1) UNIQUENESS
        , IC.COLUMN_NAME
    FROM
       DBA_INDEXES I,
       DBA_IND_COLUMNS IC
   WHERE
       I.INDEX_NAME = IC.INDEX_NAME
       AND OWNER = INDEX_OWNER
       AND I.OWNER      LIKE upper('&OWNER')
       AND I.TABLE_NAME LIKE upper('&TABLE_NAME')
ORDER BY
       1,2,3,5 DESC,4,COLUMN_POSITION;
SPOOL OFF
EXIT
```

```
rem  NAME: INX_STAT.sql
rem  HISTORY:
rem  Date             Who                    What
rem  05/27/93         Mike Ault              Initial creation
rem
rem  FUNCTION: Report on index statistics
rem  INPUTS:    1 = Index owner    2 = Index name
rem
def iowner= '&OWNER'
def iname  = '&INDEX'
set pages 56 lines 130 verify off feedback off
column owner           format a20            heading "Owner"
column index_name      format a20            heading "Index"
column status          format a11            heading "Status"
column blevel          format 9,999,999,999 heading "Tree Level"
column leaf_blocks     format 999,999,999 heading "Leaf Blk"
column distinct_keys   format 999,999        heading "# Keys"
column avg_leaf_blocks_per_key format 999,999,999 heading "Avg. LB/Key"
column avg_data_blocks_per_key format 999,999,999 heading "Avg. DB/Key"
column clustering_factor format 999,999,999 heading "Clstr Factor"
rem
start title132 "Index Statistics Report"
spool report_outpur/ind_stat&db
rem
select
       owner,
       index_name,
       status,
       blevel,
       leaf_blocks,
       distinct_keys,
       avg_leaf_blocks_per_key,
       avg_data_blocks_per_key,
       clustering_factor
from
       dba_indexes
where
       owner like upper('&&iowner')
       and index_name like upper('&&iname')
order by 1,2;
rem
spool off
exit
```

```
rem  ************************************************************************
rem  NAME: PKEY.sql
rem
rem  FUNCTION:  This routine prints a report of all primary keys defined
rem             in the data dictionary (for owners other than SYS and rem
           SYSTEM).
rem       As written, it must be run by a DBA.  By changing the query to
rem       run against ALL_CONSTRAINTS, etc. it could be used by any user
rem       to see the primary keys which are "available" to them.
rem
rem  INPUTS: Owner for tables
rem  ************************************************************************
ACCEPT OWNER PROMPT 'ENTER OWNER NAME OR "ALL" '
rem
COLUMN OWNER           FORMAT A15  NOPRINT NEW_VALUE OWNER_VAR
COLUMN TABLE_NAME              FORMAT A25     HEADING 'Table Name'
COLUMN CONSTRAINT_NAME         FORMAT A20  HEADING 'Constraint Name'
COLUMN COLUMN_NAME     FORMAT A25     HEADING' Column Name'
rem
BREAK ON OWNER SKIP PAGE ON TABLE_NAME SKIP 1 ON
       CONSTRAINT_NAME
rem
start title80 "Primary Keys For Database Tables"
SPOOL rep_out\&db\pkey
rem
SELECT
       C.OWNER,
       C.TABLE_NAME,
       C.CONSTRAINT_NAME,
       CC.COLUMN_NAME
 FROM
       DBA_CONSTRAINTS C,
       DBA_CONS_COLUMNS CC
 WHERE
       C.CONSTRAINT_NAME = CC.CONSTRAINT_NAME
       AND C.OWNER = CC.OWNER
       AND C.CONSTRAINT_TYPE = 'P'
       AND C.OWNER <> 'SYS'
       AND C.OWNER <> 'SYSTEM'
       AND C.OWNER = UPPER('&&OWNER')
       OR UPPER('&&OWNER') = 'ALL'
ORDER BY
       C.OWNER,
       C.TABLE_NAME,
       CC.POSITION;
SPOOL OFF
UNDEF OWNER
exit
```

```
rem  ****************************************************************
rem  NAME: FKEY.sql
rem
rem  FUNCTION: This routine prints a report of all foreign keys defined
rem            in data dictionary(for owners other than SYS and SYSTEM).
rem            AS written, it must be run by a DBA.By changing the query
rem            against ALL_CONSTRAINTS, etc. it could be used by any
rem            user to see the foreign keys which are "available".
rem  ****************************************************************
SET LINES 130 PAGES 56
rem
COLUMN OWNER                  FORMAT A10  NOPRINT NEW_VALUE OWNER_VAR
COLUMN TABLE_NAME             FORMAT A24  HEADING'TABLE NAME'
COLUMN REF_TABLE              FORMAT A24  HEADING'REF TABLE'
COLUMN R_OWNER                FORMAT A10  NOPRINT
COLUMN CONSTRAINT_NAME        FORMAT A30  HEADING 'CONST NAME'
COLUMN R_CONSTRAINT_NAME      FORMAT A30  NOPRINT
COLUMN COLUMN_NAME                           FORMAT A20  HEADING'COLUMN'
COLUMN REF_COLUMN                            FORMAT A20  HEADING'REF
COLUMN'
rem
start title132 "FOREIGN KEY REPORT"
rem
BREAK ON OWNER SKIP PAGE ON TABLE_NAME SKIP 1 -
            ON CONSTRAINT_NAME ON REF_TABLE
SPOOL rep_out\&db\fkey
rem
SELECT
       C.OWNER,
       C.TABLE_NAME,
       C.CONSTRAINT_NAME,
       CC.COLUMN_NAME,
      R.TABLE_NAME REF_TABLE,
       RC.COLUMN_NAME REF_COLUMN
  FROM
       DBA_CONSTRAINTS C,
      DBA_CONSTRAINTS R,
      DBA_CONS_COLUMNS CC,
      DBA_CONS_COLUMNS RC
 WHERE
       C.CONSTRAINT_TYPE = 'R'
       AND C.OWNER NOT IN ('SYS','SYSTEM')
       AND C.R_OWNER = R.OWNER
       AND C.R_CONSTRAINT_NAME = R.CONSTRAINT_NAME
       AND C.CONSTRAINT_NAME = CC.CONSTRAINT_NAME
       AND C.OWNER = CC.OWNER AND
       AND R.CONSTRAINT_NAME = RC.CONSTRAINT_NAME
       AND R.OWNER = RC.OWNER
       AND CC.POSITION = RC.POSITION
ORDER BY
       C.OWNER,
       C.TABLE_NAME,
       C.CONSTRAINT_NAME,
       CC.POSITION;
SPOOL OFF
EXIT
```

```
rem  NAME: db_seqs.sql
rem
rem  FUNCTION: Generate a report on sequences for an account
rem
rem  INPUTS:
rem                   1 - Sequence Owner or Wild Card
rem                   2 - Sequence Name or Wild Card
rem  ****************************************************************
SET HEADING OFF VERIFY OFF PAUSE OFF
rem
PROMPT ** Sequence Report **
PROMPT
PROMPT Percent signs are wild
ACCEPT sequence_owner prompt 'Enter Oracle account for report:';
ACCEPT sequence_name prompt 'Enter object name for report: ';
PROMPT
PROMPT Report file name is db_seqs.lis
rem
SET HEADING ON
SET LINESIZE 130 PAGESIZE 56 NEWPAGE 0 TAB OFF SPACE 1
SET TERMOUT OFF
BREAK ON SEQUENCE_OWNER SKIP 2
rem
COLUMN SEQUENCE_OWNER  FORMAT A10            HEADING 'Sequence Owner'
COLUMN SEQUENCE_NAME   FORMAT A30            HEADING 'Sequence Name'
COLUMN MIN_VALUE                                    HEADING 'Minimum'
COLUMN MAX_VALUE                                    HEADING 'Maximum'
COLUMN INCREMENT_BY    FORMAT 9999           HEADING 'Incr.'
COLUMN CYCLE_FLAG                                   HEADING 'Cycle'
COLUMN ORDER_FLAG                                   HEADING 'Order'
COLUMN CACHE_SIZE             FORMAT 99999   HEADING 'Cache'
COLUMN LAST_NUMBER                           HEADING 'Last Value'
rem
START title132 "SEQUENCE REPORT"
SPOOL rep_out\&db\db_seqs
rem
SELECT
       SEQUENCE_OWNER,
       SEQUENCE_NAME,
       MIN_VALUE,
       MAX_VALUE,
       INCREMENT_BY,
       DECODE(CYCLE_FLAG,'Y','YES','N','NO') CYCLE_FLAG,
       DECODE(ORDER_FLAG,'Y','YES','N','NO') ORDER_FLAG,
       CACHE_SIZE,
       LAST_NUMBER
FROM
       DBA_SEQUENCES
WHERE
       SEQUENCE_OWNER LIKE UPPER('&SEQUENCE_OWNER') AND
       SEQUENCE_NAME LIKE UPPER('&SEQUENCE_NAME')
ORDER BY
       1,2;
SPOOL OFF
pause Press Enter to continue
EXIT
```

```
rem  ****************************************************************
rem  NAME              :  db_obj.sql
rem  FUNCTION  :  Report on all objects and their status
rem  HISTORY:
rem  Date              Who                    What
rem  ____              _______                ____________________
rem  02/08/94  Michael Ault   Created
rem  ****************************************************************
column  owner          format a17     heading "Owner"
column  object_name    format a23     heading "Object"
column object_type     format a20     heading "Type of Object"
column status          format a10     heading "Status"
rem
set verify off  feedback off lines 80 pages 58 heading off
rem
break on owner on  object_type
start  title80 "Database Object Status"
spool rep_out\&db\db_obj
rem
prompt Enter % for all objects
rem
select
       owner,
       object_type,
       object_name,
       status
from
       dba_objects
where
       object_type like upper('&type')
order by
       1,2;
spool off
pause Press enter to continue
exit
```

```
rem  ***************************************************************
rem  NAME              :  db_trig.sql
rem  FUNCTION  :  Report on all triggers and their status
rem  HISTORY:
rem  Date              Who                    What
rem  ____              _______                ____________________
rem  02/08/94  Michael Ault   Created
rem  ***************************************************************
column owner           format a10
column trigger_name    format a20     heading "Trigger"
column trigger_type    format a20     heading "Type"
column triggering_event format a10    heading "Event"
column table           format a40     heading "Trigger Table"
rem
break on owner
set verify off  feedback off lines 130 pages 58
start title132 "Trigger Status Report"
spool  trigger_status&db
rem
Select
       owner,
       trigger_name,
       trigger_type,
       triggering_event,
       table_owner||'.'||table_name "Table",
       status
from dba_triggers
order by 1,5,2;
spool off
exit
```

```
REM
REM NAME               :db_view.sql
REM FUNCTION:     :report on database views by owner
REM USE                :Generate a report on database views
REM Limitations        :If your view definitions are greater than 5000
REM                     characters then increase the set long. This can be
REM                     determined by querying the DBA_VIEWS table's
REM                     text_length column for the max value:
REM                      select max(text_length) from dba_views;
REM
set pages 59 lines 131 feedback off echo off
column text            format a75
column owner           format a20
column view_name              format a20
column status          format a7
set long 5000
start title132 "Database Views Report"
spool rep_out\&db\db_views
select
       v.owner,
       v.view_name,
       v.text,
       o.status
from
       dba_views v,
       dba_objects o
where
       v.owner like upper('%&owner_name') and
       o.owner = v.owner and
       o.object_type = 'VIEW' and
       o.object_name = v.view_name
order by
       o.status,
       v.view_name;
spool off
Pause Press enter to continue
exit
```

```
REM NAME              : DB_LINKS.SQL
REM FUNCTION:  GENERATE REPORT OF DATABASE LINKS
REM USE        : FROM SQLPLUS
REM Limitations       : None
set pages 58 lines 130 verify off term off
start title132 "Db Links Report"
spool rep_out\&db\db_links
column host           format a60     heading "Connect String Used"
column owner          format a15     heading "Creator of DB Link"
column db_link                format a10     heading " DB Link Name"
column username               format a15     heading "Connecting User"
column password               format a15     heading "Password"
column create                        heading "Date Created"
select
       host,
       owner,
       db_link,
       username,
       created
from
       dba_db_links
order by
       owner,
       host;
spool off
pause Press enter to continue
exit
```

```
rem TITLE132.SQL - FUNCTION:    This SQL*Plus script builds a standard
rem                             heading for 132 col database reports
rem
column  TODAY          NEW_VALUE      CURRENT_DATE   NOPRINT
column  TIME                   NEW_VALUE       CURRENT_TIME  NOPRINT
column  DATABASE               NEW_VALUE       DATA_BASE     NOPRINT
column  PASSOUT        NEW_VALUE       DBNAME        NOPRINT
define COMPANY = "Insert Company Name"
define HEADING = "&1"
rem
TTITLE LEFT "Date: " current_date CENTER company col 118 "Page:" format
999 -
       SQL.PNO SKIP 1 LEFT "Time: " current_time CENTER heading RIGHT -
       format a15 SQL.USER SKIP 1 CENTER format a20 data_base SKIP 2
set heading off
set pagesize 0
SELECT
       TO_CHAR(SYSDATE,'MM/DD/YY') TODAY,
       TO_CHAR(SYSDATE,'HH:MI AM') TIME,
       value||' database' DATABASE,
       rtrim(value) passout
FROM
       sys.v_$parameter
WHERE
       name = 'db_name';
set heading on
set pagesize 58
set newpage 0
DEFINE DB = '_&DBNAME'

rem TITLE80.SQL - FUNCTION:    This SQL*Plus script builds a standard rem
                                report heading for database reports that
rem                                    are 80 columns
column  TODAY          NEW_VALUE      CURRENT_DATE   NOPRINT
column  TIME                   NEW_VALUE       CURRENT_TIME   NOPRINT
column  DATABASE               NEW_VALUE       DATA_BASE         NOPRINT
column  PASSOUT        NEW_VALUE       DBNAME              NOPRINT
define COMPANY = "Insert Company Name"
define HEADING = "&1"
rem
TTITLE LEFT "Date: " current_date CENTER company col 66 "Page:" format 999
-
       SQL.PNO SKIP 1 LEFT "Time: " current_time CENTER heading RIGHT -
       format a15 SQL.USER SKIP 1 CENTER format a20 data_base SKIP 2
rem
set heading off
set pagesize 0
SELECT
       TO_CHAR(SYSDATE,'MM/DD/YY') TODAY,
       TO_CHAR(SYSDATE,'HH:MI AM') TIME,
       value||' database' DATABASE,
       rtrim(value) passout
FROM   v$parameter
where name = 'db_name';
set heading on
set pagesize 58
set newpage 0
DEFINE DB = '_&DBNAME'
```