<A>APPENDIX C

<B>SQL AND PL/SQL SCRIPTS FROM BOOK

DBAs must know how to write and use SQL and PL/SQL scripts. The tools available to modern DBAs are all wonderful, including the Q product which is on the enclosed CD, however, there comes a time when the DBA needs a specific bit of information, a different cut at the same data, or, heaven forbid, a paper report they can show management or use in a report. To accomplish this a DBA must know SQL and PL/SQL and the SQL*Plus formatting commands.

A general warning: Each of these scripts has been test run and they should execute properly, however, sometimes when switching from notepad to word to disk and back things like single quotes and other "minor" format items may get set for standard written text instead of what the computer expects to see. If you get the error saying the SQLPLUS or other interface doesn't understand or has a bad character, look to the single quotes first. If you have problems with some on UNIX due to control m characters (you shouldn't but it may happen) try reading the script from the CD into word and then save it as text only, this usually solves that problem. A final word, Oracle likes to add columns, remove columns and change tables, views and even statistics names, I have tried to ensure all of these are current, up to 8.0.2, but, I can't say that in 8.0.3 or later things won't change.

The scripts in this book should provide a firm foundation from which the DBA can build an excellent toolbox of ready made reports, scripot builders and other items that all DBAs (at least with Oracle) should have. The key to scripts and their building and usage are the SQLPLUS, SVRMGR and OEM-SQL Worksheet interfaces. I prefer the SQL*Plus interface since every user can use it and it recognizes (naturally) all SQL*PLus formatting commands. This is followed by SVRMGR for the same reasons, bringing up the rear is the SQL Worksheet simply because it han't been taught to recognize SQL*Plus formatting commands. Every platform has at least the first two tools, SQL*Plus and SVRMGR, usually both in GUI and command line form.

<B>To get into SQL*Plus:

SQLPLUS username/password@connect string @command file

Where:

username/password - This is the user's Oracle username and password, which is usually different from the Operating System username and password. If the user is assigned an autologin type of account, only the / is required.

@connect string - This is a connect string that connects the user to other databases than the default database. It can be used with SQL*NET or NET8 over networks to access other systems.

@command file - This allows the user to specify a SQL command file that is run automatically.

If the DBA account is what is known as an OPS$ account (not recommended), the format would be as follows.

SQLPLUS /

Since an OPS$ account allows the user to get into the Oracle system without specifying a password if they are logging in from their normal account, the use of OPS$ accounts should be restricted to "captive" type users, that is, users who can only access the system via an appropriate secure menu system. Under ORACLE7 and ORACLE8 the OPS$ format is the default but the system manager can assign whatever prefix they desire by use of the OS_AUTHENT_PREFIX parameter in the INIT.ORA file.

To exit SQL*Plus use “exit” on most platforms.

On my WINDOWS, WIN(5 ans WINNT platforms I usually create multiple shortcuts, one for each database, from the SQLPLUS program icon. If you right mouse click on the shortcut you can get to the properties listing, from there on WINDOWS it is easy to edit the command line, from NT and WIN95 switch to the Shortcut tab and edit the target line to include a username and password as well as connect string, for example on my NY platform one of my shortcuts has the target line:

D:\ORANT\RDBMS80\BIN\PLUS40W.EXE\ system/system_test@beq-test

This brings up the SQLPLUS program against the TEST (ORTEST1) instance. I also set the Start In: setting to the location of my SQL and PL/SQL scripts (usually C:\SQL_SCRIPTS). Off of the SQL_SCRIPTS directory I hang a “rep_out” directory with sub-directories for all instances. This is why you will see a majority of the scripts spooling out to “rep_out/&db/list_name” or “rep_out\&db\list_name”, suprizingly it doesn’t seem to make a diffeene which slash is used, the reports seem to get to the right place. It makes it easy to find reports and I use the same format on all platforms.

<B>To Get into SVRMGR:

$ svrgmr -- or -- svrmgrl (line mode)-- or -- svrmgr30 (NT, WIN95) -- or -- svrmgrm (motif mode)

You will get a normal prompt for svrmgr:

SVRMGR>

just enter your connect command:

SVRMGR> connect internal/password

To exit SVRMGR just type ‘exit’

The OEM SQL Worksheet has a self explanatory GUI interface.

Enough general verbage, here are the scripts from the book, I hope you find them useful, they are also included on the CD in the Dictionary Lite product in a form I hope you find easy to use.

<B>Script to build a database creation script:

```
REM FUNCTION: SCRIPT FOR CREATING DB
REM           This script must be run by a user with the DBA role.
REM           This script is intended to run with Oracle7 or 8.
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           Only preliminary testing of this script was performed.
REM           Be sure to test it completely before relying on it.
REM M. Ault 3/29/96 TRECOM, REVELNET
REM
SET VERIFY OFF FEEDBACK OFF ECHO OFF PAGES 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 grp_cursor IS
      SELECT
```

```
            group#
      FROM
            v$log;
   CURSOR mem_cursor (grp_num number) IS
      SELECT
            a.member, b.bytes from v$logfile a, v$log b
      WHERE
            a.group#=grp_num
            AND a.group#=b.group#
      ORDER BY
            member;
   grp_member           v$logfile.member%TYPE;
   bytes                v$log.bytes%TYPE;
   db_name              VARCHAR2(8);
   db_string            VARCHAR2(255);
   db_lineno            NUMBER := 0;
   thrd                 NUMBER;
   grp                  NUMBER;
   filename             dba_data_files.file_name%TYPE;
   sz                   NUMBER;
   begin_count          NUMBER;
   max_group            NUMBER;
   PROCEDURE write_out(p_line INTEGER,
                  p_string VARCHAR2) IS
      BEGIN
         INSERT INTO db_temp (lineno,text)
                VALUES (db_lineno,db_string);
      END;
BEGIN
      SELECT MAX(group#) INTO max_group FROM v$log;
      db_lineno:=db_lineno+1;
   SELECT 'CREATE DATABASE '||name INTO db_string
      FROM v$database;
     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 grp_cursor%ISOPEN
THEN
      CLOSE grp_cursor;
      OPEN grp_cursor;
ELSE
      OPEN grp_cursor;
END IF;
LOOP
      FETCH grp_cursor INTO grp;
      EXIT WHEN grp_cursor%NOTFOUND;
      db_lineno:=db_lineno+1;
      db_string:= ' 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, bytes;
            EXIT when mem_cursor%NOTFOUND;
            IF begin_count=db_lineno THEN
                  db_string:=chr(39)||grp_member||chr(39);
                  write_out(db_lineno,db_string);
                  db_lineno:=db_lineno+1;
            ELSE
                  db_string:=','||chr(39)||grp_member||chr(39);
                  write_out(db_lineno,db_string);
                  db_lineno:=db_lineno+1;
            END IF;
      END LOOP;
      db_lineno:=db_lineno+1;
      IF grp=max_group
      THEN
            db_string:=' ) SIZE '||bytes;
            write_out(db_lineno,db_string);
      ELSE
            db_string:=' ) SIZE '||bytes||',';
            write_out(db_lineno,db_string);
      END IF;
END LOOP;
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 '||chr(39)||filename||chr(39)||' SIZE '||sz||'
REUSE';
      ELSE
      db_string:=','||chr(39)||filename||chr(39)||' 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 grp_cursor;
COMMIT;
END;
/
rem The next section could be converted to use
rem UTLFILE so the entire anonymous PL/SQL section
rem and this report section would become a stored
rem procedure, but to keep it generic I will leave as
rem is.
COLUMN dbname NEW_VALUE db NOPRINT
SELECT name dbname FROM v$database;
SET HEADING OFF PAGES 0 VERIFY OFF RECSEP OFF
SPOOL rep_out\&db\crt_db.sql
COLUMN 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 continue
SET VERIFY ON FEEDBACK ON PAGES 22 TERMOUT ON
CLEAR COLUMNS
```

<B>Script to Recreate the database initialization file entries for an instance

```
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 a80 WORD_WRAPPED
COLUMN dbname NEW_VALUE db NOPRINT
SELECT name dbname FROM v$database;
DEFINE OUTPUT = 'rep_out\&db\init.ora'
DEFINE cr = chr(10)
SPOOL &OUTPUT
SELECT '# Init.ora file from v$parameter'||&&cr||
'# generated on:'||sysdate||&&cr||
'# script by MRA 11/7/95 REVEALNET'||&&cr||
'#' name FROM dual
UNION
SELECT name||' = '||value name  FROM V$PARAMETER
WHERE value IS NOT NULL;
SPOOL OFF
CLEAR COLUMNS
SET NEWPAGE 0 VERIFY OFF
SET TERMOUT ON PAGES 22 LINES 80 HEADING ON
SET TERMOUT ON
UNDEF OUTPUT
PAUSE Press enter to continue
```

<B>Example of the output from ALTER SSYSTEM BACKUP CONTROL FILE TO TRACE

```
# 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;
```

<B>Example PL/SQL procedure to execute a set of process kill commands:

```
 CREATE OR REPLACE PROCEDURE kill_session ( session_id in varchar2,
serial_num in varchar2)
AS
cur INTEGER;
ret INTEGER;
string VARCHAR2(100);
BEGIN
 string :=
        'ALTER SYSTEM KILL SESSION
'||''''||session_id||','||serial_num||'''';
   cur := dbms_sql.open_cursor;
   dbms_sql.parse(cur,string,dbms_sql.v7);
   ret := dbms_sql.execute(cur)  ;
   dbms_sql.close_cursor(cur);
```

```
EXCEPTION
   WHEN OTHERS THEN
      raise_application_error(-20001,'Error in execution',TRUE);
      IF dbms_sql.is_open(cur) THEN
        dbms_sql.close_cursor(cur);
      END IF;
END;
/
```

<B>Example script to invoke the kill procedure

```
REM
REM ORA_KILL.SQL
REM FUNCTION: Kills non-essential Oracle sessions (those that aren't
owned
REM         : by SYS or "NULL"
REM DEPENDANCIES: Depends on kill_session procedure
REM MRA 9/12/96
REM
SET HEADING OFF TERMOUT OFF VERIFY OFF ECHO OFF
SPOOL kill_all.sql
SELECT 'EXECUTE kill_session('||chr(39)||sid||chr(39)||','||
chr(39)||serial#||chr(39)||');' FROM v$session
WHERE username IS NOT NULL
OR username <> 'SYS'
/
SPOOL OFF
START kill_all.sql
```

<B>Scripts for getting undocumented initialization parameters from 7.x, 7.3 and 8.x

Databases

```
REM Script for getting undocumented init.ora
REM parameters from a 7.2 instance
REM MRA - Revealnet 2/23/97
REM
COLUMN parameter FORMAT a40
COLUMN value FORMAT a30
COLUMN ksppidf HEADING 'Is|Default'
SET FEEDBACK OFF VERIFY OFF PAGES 55
START title80 'Undocumented Init.ora Parameters'
SPOOL rep_out/&db/undoc
SELECT  ksppinm "Parameter",
  ksppivl "Value",
  ksppidf
FROM x$ksppi
WHERE ksppinm LIKE '/_%' escape '/'
/
SPOOL OFF
TTITLE OFF
```

```
REM Script for getting undocumented init.ora
REM parameters from a 7.3 or 8.0.2 instance
REM MRA - Revealnet 4/23/97
REM
COLUMN parameter        FORMAT a37
COLUMN description      FORMAT a30 WORD_WRAPPED
COLUMN "Session Value"  FORMAT a10
COLUMN "Instance Value" FORMAT a10
SET LINES 100
SET PAGES 0
SPOOL undoc.lis
SELECT
      a.ksppinm  "Parameter",
      a.ksppdesc "Description",
      b.ksppstvl "Session Value",
      c.ksppstvl "Instance Value"
FROM
      x$ksppi a,
      x$ksppcv b,
      x$ksppsv c
WHERE
      a.indx = b.indx
      AND a.indx = c.indx
      AND a.ksppinm LIKE '/_%' escape '/'
/
SPOOL OFF
SET LINES 80 PAGES 20
CLEAR COLUMNS
```

<B>Anonymous PL/SQL--SQL--SQLPLUS Script to generate a tablespace rebuild script

```
REM TBSP_RCT.SQL
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 or 8.
REM
REM FUNCTION: Running this script will in turn create a script to build
REM           all the tablespaces in the database.  This created script,
REM           crt_tbls.sql, can be run by any user with the DBA role
REM           or with the 'CREATE TABLESPACE' system privilege.
REM
REM Only preliminary testing of this script was performed.  Be sure to
REM test it completely before relying on it.
REM
REM
SET VERIFY OFF TERMOUT OFF FEEDBACK OFF ECHO OFF PAGESIZE 0
SET TERMOUT ON
PROMPT 'Creating tablespace build script...'
SET TERMOUT OFF;
rem
rem The following view needs to be created in SYS with a public
```

```
rem  select grant and synonym or this script will not work
rem create or replace view dba_file_data as
rem select
rem a.name tablespace,a.dflminext min_extents, a.dflmaxext max_extents,
rem a.dflinit init,a.dflincr next,a.dflextpct pct_increase, d.name
rem datafile,
rem b.blocks datafile_size, c.maxextend max_extend, c.inc ext_incr
rem from sys.ts$ a, sys.file$ b, sys.filext$ c, v$dbfile d
rem where
rem a.ts#=b.ts# and
rem b.file#=c.file#(+) and
rem b.file#=d.file#(+)
rem /
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,
                     0,
                     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;
   CURSOR get_auto (df_nm VARCHAR2) IS
            SELECT
                     max_extend, ext_incr
                     ext_incr
            FROM
                     dba_file_data
            WHERE
                     datafile=df_nm;
   lv_max_extend        dba_file_data.max_extend%TYPE;
   lv_ext_incr          dba_file_data.ext_incr%TYPE;
   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;
   lv_min_extlen        NUMBER := 0;
   sub_strg             VARCHAR2(20);
   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_min_extlen,
                           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';
         OPEN get_auto(lv_file_name);
         FETCH get_auto INTO lv_max_extend, lv_ext_incr;
         IF lv_max_extend=0 THEN
                        sub_strg:='MAXSIZE UNLIMITED';
             ELSE
                        sub_strg:=' MAXSIZE '||TO_CHAR(lv_max_extend);
             END IF;
         IF lv_ext_incr != 0 THEN
                 lv_string:=lv_string||chr(10)||' AUTOEXTEND ON NEXT '||
                 to_char(lv_ext_incr)||sub_strg;
             CLOSE get_auto;
         END IF;
      IF get_auto%ISOPEN THEN
            CLOSE get_auto;
      END IF;
      IF lv_min_extlen != 0 THEN
            lv_string:=lv_string||chr(10)||
                        'MINIMUM EXTENT '||TO_CHAR(lv_min_extlen);
      END IF;
```

```
      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;
/
COLUMN dbname NEW_VALUE db NOPRINT
SELECT name dbname FROM v$database;
SPOOL rep_out\&db\crt_tbsp.sql
SET HEADING OFF
COLUMN text FORMAT a80 WORD_WRAP
SELECT   text
FROM     ts_temp
ORDER BY ts_name, lineno;
SPOOL OFF;
DROP TABLE ts_temp;
SET VERIFY ON RECSEP ON TERMOUT ON HEADING ON FEEDBACK ON
SET PAGESIZE 22 LINES 80
CLEAR COLUMNS
```

<B>Script to build free_space view for use with free_space and consolidation scripts (this view is included in the Dictionary Lite crea_tab.sql script)

```
rem Name:     view.sql
rem FUNCTION: Create free_space view for use by freespc reports
rem
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;
```

<B>Script which uses fre_space view and set events command to consolidate extents for pre-7.2 databases:

```
rem
rem NAME: defrag7.sql
rem FUNCTION: Uses the "set events" command to manually coalesce
rem FUNCTION: any tablespace with greater than 1 fragment. You
rem FUNCTION: may wish to alter to exclude the temporary tablespace.
rem FUNCTION: The procedure uses the FREE_SPACE view which is a
rem FUNCTION: summarized version of the DBA_FREE_SPACE view.
rem FUNCTION: This procedure must be run from a DBA user id.
rem HISTORY:
rem WHO           WHAT        WHEN
rem Mike Ault           Created           1/4/96
rem
SET HEADING OFF FEEDBACK OFF ECHO OFF TERMOUT OFF
SPOOL def.sql
SELECT
      'ALTER SESSION SET EVENTS '||
      ''''||'IMMEDIATE TRACE NAME COALESCE LEVEL '||ts#||''''||';'
FROM
      sys.ts$,
      free_space
WHERE
      ts#=file_id-1 AND pieces>1;
SPOOL OFF
@def.sql
HOST rm def.sql
```

<B>Script to use free_space view and ALTER TABLESPACE command to consolidate extents in post-7.2/7.3 databases:

```
rem
rem NAME: defrg73.sql
rem FUNCTION: Uses the coalesce command to manually coalesce
rem FUNCTION: any tablespace with greater than 1 fragment. You
rem FUNCTION: may wish to alter to exclude the temporary tablespace.
rem FUNCTION: The procedure uses the FREE_SPACE view which is a
rem FUNCTION: summarized version of the DBA_FREE_SPACE view.
rem FUNCTION: This procedure must be run from a DBA user id.
rem HISTORY:
rem WHO           WHAT              WHEN
rem Mike Ault     Created          1/4/96
rem
CLEAR COLUMNS
```

```
CLEAR COMPUTES
DEFINE cr='chr(10)'
TTITLE OFF
SET HEADING OFF FEEDBACK OFF ECHO OFF TERMOUT OFF
SPOOL def.sql
SELECT
      'ALTER TABLESPACE '||tablespace||' COALESCE;'||&&cr||
      'COMMIT;'
FROM
      free_space
WHERE
      pieces>1;
SPOOL OFF
@def.sql
HOST rm def.sql
SET HEADING ON FEEDBACK ON TERMOUT ON
TTITLE OFF
```

<B>Script to find objects bounded by free space (bound objects)

```
rem  ****************************************************************
rem  NAME:     BOUND_OB.sql
rem  FUNCTION: Report on objects with extents bounded by freespace
rem ****************************************************************
START title80 "Objects With Extents Bounded by Free Space"
SPOOL rep_out\&db\b_ob..lis
COLUMN e FORMAT a15        HEADING "TABLE SPACE"
COLUMN a FORMAT a6         HEADING "OBJECT|TYPE"
COLUMN b FORMAT a30        HEADING "OBJECT NAME"
COLUMN c FORMAT a10        HEADING "OWNER ID"
COLUMN d FORMAT 99,999,999 HEADING "SIZE|IN BYTES"
BREAK ON e SKIP 1 ON c
SET FEEDBACK OFF
SET VERIFY OFF
SET TERMOUT OFF
COLUMN bls NEW_VALUE block_size NOPRINT
SELECT blocksize bls
FROM sys.ts$
WHERE name='SYSTEM';
SELECT h.name e, g.name c, f.object_type a, e.name b,
b.length*&&block_size d
 FROM sys.uet$ b, sys.fet$ c, sys.fet$ d, sys.obj$ e, sys.sys_objects f,
       sys.user$ g, sys.ts$ h
 WHERE b.block# = c.block# + c.length
   AND b.block# + b.length = d.block#
   AND f.header_file = b.segfile#
   AND f.header_block = b.segblock#
   AND f.object_id = e.obj#
   AND g.user# = e.owner#
   AND b.ts# = h.ts#
 ORDER BY 1,2,3,4
/
CLEAR COLUMNS
SET FEEDBACK ON
SET VERIFY ON
SET TERMOUT ON
```

```
TTITLE ''
TTITLE OFF
SPOOL OFF
CLEAR BREAKS
```

<B>PL/SQL--SQL--SQLPLUS script to create a tablespace level export script:

```
rem********** RevealNet Oracle Administration **********
rem  File: tbsp_exp.sql
rem  This is a part of the RevealNet Oracle Administration library.
rem  Copyright (C) 1996-97 RevealNet, Inc.
rem  All rights reserved.
rem  For more information, call RevealNet at 1-800-REVEAL4
rem  or check out our Web page: www.revealnet.com
rem  Modifications (Date, Who, Description)
rem FUNCTION: Creates a basic shell script to perform tablespace level
rem FUNCTION: exports for a database
rem FUNCTION: Each tablespace is given its own export that handles
rem FUNCTION: its tables and their related indexes, grants and contraints
rem NOTE: Only preliminary testing of this script has been done, you
rem NOTE: should test this script throughly before production use.
rem ************************************************************************
rem
SET VERIFY OFF ECHO OFF TERMOUT ON FEEDBACK OFF
PROMPT ...creating tablespace level export script
SET TERMOUT OFF
DROP TABLE exp_temp;
CREATE TABLE exp_temp (file# NUMBER, line_no NUMBER, line_txt long);
DECLARE
CURSOR count_tabs (tbsp IN VARCHAR2) IS
        SELECT count(*)
        FROM dba_tables
        WHERE tablespace_name=tbsp;
CURSOR get_tbsp IS
        SELECT tablespace_name
        FROM dba_tablespaces
        WHERE tablespace_name != 'SYSTEM';
CURSOR get_owners ( tbsp IN VARCHAR2 ) IS
        SELECT DISTINCT(owner)
        FROM dba_tables
        WHERE tablespace_name=tbsp;
cursor get_tabs ( tbsp IN VARCHAR2, owner in VARCHAR2 ) IS
        SELECT table_name
        FROM dba_tables
        WHERE tablespace_name=tbsp
                AND owner=owner;
row_cntr                INTEGER:=0;
tablespace_nm           dba_tablespaces.tablespace_name%TYPE;
owner                   dba_tables.owner%TYPE;
table_nm                dba_tables.table_name%TYPE;
ln_txt                  exp_temp.line_txt%TYPE;
own_cnt                 INTEGER;
tab_cnt                 INTEGER;
file_no                 INTEGER;
tab_count               INTEGER;
```

```
dbname                  v$database.name%TYPE;
PROCEDURE insert_tab (file_no NUMBER, row_cntr NUMBER, ln_txt VARCHAR2) IS
BEGIN
        INSERT INTO exp_temp (file#,line_no, line_txt)
        VALUES (file_no,row_cntr,ln_txt);
END;
BEGIN
/* initialize various counters */
row_cntr        :=0;
tab_count       :=0;
file_no         :=1;
/* Get database name */
SELECT name INTO dbname FROM v$database;
ln_txt:='# Tablespace level export script for instance: '||dbname;
row_cntr:=row_cntr+1;
insert_tab (file_no, row_cntr, ln_txt);
/* Set command in script to set SID */
ln_txt:='ORACLE_SID='||LOWER(dbname);
row_cntr:=row_cntr+1;
insert_tab (file_no, row_cntr, ln_txt);
/* First run to build export script header
 Get all tablespace names other than system */
IF get_tbsp%ISOPEN THEN
        CLOSE get_tbsp;
        OPEN get_tbsp;
ELSE
        OPEN get_tbsp;
END IF;
LOOP
        FETCH get_tbsp INTO tablespace_nm;
        EXIT WHEN get_tbsp%NOTFOUND;
/* See if tablespace has tables */
        IF count_tabs%ISOPEN THEN
                CLOSE count_tabs;
                OPEN count_tabs(tablespace_nm);
        ELSE
                OPEN count_tabs(tablespace_nm);
        END IF;
        FETCH count_tabs INTO tab_count;
        IF tab_count=0 THEN
                GOTO end_loop1;
        END IF;
        row_cntr:=row_cntr+1;
        ln_txt:='#';
        insert_tab (file_no, row_cntr, ln_txt);
        row_cntr:=row_cntr+1;
        ln_txt:='#';
        insert_tab (file_no, row_cntr, ln_txt);
        SELECT '# Tablespace: '||tablespace_nm INTO ln_txt FROM dual;
        row_cntr:=row_cntr+1;
        insert_tab (file_no, row_cntr, ln_txt);
        SELECT '#   Export DMP file name: '||tablespace_nm||'_'||trunc(sysdate)||'.dmp' INTO ln_txt
        FROM dual;
        row_cntr:=row_cntr+1;
        insert_tab (file_no, row_cntr, ln_txt);
        row_cntr:=row_cntr+1;
        ln_txt:='#   Owners for '||tablespace_nm;
        insert_tab (file_no, row_cntr, ln_txt);
```

```
        SELECT '' into ln_txt FROM dual;
        own_cnt:=0;
/* Get tablespace table owners */
        IF get_owners%ISOPEN THEN
                CLOSE get_owners;
                OPEN get_owners(tablespace_nm);
        ELSE
                open get_owners(tablespace_nm);
        END IF;
        tab_cnt:=0;
        LOOP
                FETCH get_owners INTO owner;
                EXIT WHEN get_owners%NOTFOUND;
/* Get tablespace tables */
                ln_txt:='#   Tables for tablespace: '||tablespace_nm;
                row_cntr:=row_cntr+1;
                insert_tab (file_no, row_cntr, ln_txt);
                ln_txt:='';
                IF get_tabs%ISOPEN THEN
                        CLOSE get_tabs;
                        OPEN get_tabs(tablespace_nm,owner);
                ELSE
                        OPEN get_tabs(tablespace_nm, owner);
                END IF;
                LOOP
                        FETCH get_tabs INTO table_nm;
                        EXIT WHEN get_tabs%NOTFOUND;
                        tab_cnt:=tab_cnt+1;
                        IF tab_cnt=1 THEN
                                ln_txt:='/*    '||ln_txt||owner||'.'||table_nm;
                        ELSE
                                ln_txt:=ln_txt||', '||owner||'.'||table_nm;
                        END IF;
                END LOOP;
        CLOSE get_tabs;
        row_cntr:=row_cntr+1;
        ln_txt:=ln_txt||' */';
        insert_tab (file_no, row_cntr, ln_txt);
        END LOOP;
        CLOSE get_owners;
<<end_loop1>>
NULL;
END LOOP;
close get_tbsp;
ln_txt:='####### End of Header -- Start of actual export script ########';
row_cntr:=row_cntr+1;
insert_tab (file_no, row_cntr, ln_txt);
ln_txt:='set -x ';
row_cntr:=row_cntr+1;
insert_tab (file_no, row_cntr, ln_txt);
select 'script tablespace_exp_'||sysdate||'.log' into ln_txt
from dual;
row_cntr:=row_cntr+1;
insert_tab (file_no, row_cntr, ln_txt);
/* Now build actual export command sets */
/* Get all tablespace names other than system */
IF get_tbsp%ISOPEN THEN
        CLOSE get_tbsp;
```

```
        OPEN get_tbsp;
ELSE
        OPEN get_tbsp;
END IF;
LOOP
        FETCH get_tbsp INTO tablespace_nm;
        EXIT WHEN get_tbsp%NOTFOUND;
/* See if tablespace has tables */
        IF count_tabs%ISOPEN THEN
                CLOSE count_tabs;
                OPEN count_tabs(tablespace_nm);
        ELSE
                OPEN count_tabs(tablespace_nm);
        END IF;
        FETCH count_tabs into tab_count;
        IF tab_count=0 THEN
                GOTO end_loop;
        END IF;
        row_cntr:=row_cntr+1;
        ln_txt:='#';
        insert_tab (file_no, row_cntr, ln_txt);
        row_cntr:=row_cntr+1;
        ln_txt:='#';
        insert_tab (file_no, row_cntr, ln_txt);
        SELECT '# Export script for tablespace '||tablespace_nm INTO ln_txt FROM dual;
        row_cntr:=row_cntr+1;
        insert_tab (file_no, row_cntr, ln_txt);
        SELECT '# created on '||sysdate into ln_txt FROM dual;
        row_cntr:=row_cntr+1;
        insert_tab (file_no, row_cntr, ln_txt);
        ln_txt:='if ( -r  '||tablespace_nm||'.par'||' ) then';
        row_cntr:=row_cntr+1;
        insert_tab (file_no, row_cntr, ln_txt);
        ln_txt:='   rm '||tablespace_nm||'.par';
        row_cntr:=row_cntr+1;
        insert_tab (file_no, row_cntr, ln_txt);
        ln_txt:='end if';
        row_cntr:=row_cntr+1;
        insert_tab (file_no, row_cntr, ln_txt);
        ln_txt:='touch '||tablespace_nm||'.par';
        row_cntr:=row_cntr+1;
        insert_tab (file_no, row_cntr, ln_txt);
/* Set up basic export commands */
        SELECT
        'echo '||chr(39)||'grants=y indexes=y constraints=y
compress=y'||chr(39)||'>>'||tablespace_nm||'.par'
        INTO ln_txt
        FROM dual;
        row_cntr:=row_cntr+1;
        insert_tab (file_no, row_cntr, ln_txt);
        SELECT '' INTO ln_txt FROM dual;
        own_cnt:=0;
        ln_txt:='echo '||chr(39)||'tables=('||chr(39)||'>>'||tablespace_nm||'.par';
        row_cntr:=row_cntr+1;
        insert_tab (file_no, row_cntr, ln_txt);
/*  Get tablespace table owners */
        IF get_owners%ISOPEN THEN
                CLOSE get_owners;
```

```
                OPEN get_owners(tablespace_nm);
        ELSE
                OPEN get_owners(tablespace_nm);
        END IF;
        tab_cnt:=0;
        LOOP
                FETCH get_owners INTO owner;
                EXIT WHEN get_owners%NOTFOUND;
/* Get tablespace tables */
                IF get_tabs%ISOPEN THEN
                        CLOSE get_tabs;
                        OPEN get_tabs(tablespace_nm,owner);
                ELSE
                        OPEN get_tabs(tablespace_nm,owner);
                END IF;
                LOOP
                        FETCH get_tabs INTO table_nm;
                        EXIT WHEN get_tabs%NOTFOUND;
                        tab_cnt:=tab_cnt+1;
                        IF tab_cnt=1 THEN
                        ln_txt:='echo
'||chr(39)||owner||'.'||table_nm||chr(39)||'>>'||tablespace_nm||'.par';
                        ELSE
                        ln_txt:='echo '||chr(39)||',
'||owner||'.'||table_nm||chr(39)||'>>'||tablespace_nm||'.par';
                        END IF;
                        row_cntr:=row_cntr+1;
                        insert_tab (file_no, row_cntr, ln_txt);
                END LOOP;
                CLOSE get_tabs;
        END LOOP;
        CLOSE get_owners;
        ln_txt:='echo '||chr(39)||')'||chr(39)||'>>'||tablespace_nm||'.par';
        row_cntr:=row_cntr+1;
        insert_tab (file_no, row_cntr, ln_txt);
/*Set file name for export file*/
        SELECT
                'echo '||chr(39)||'file='||tablespace_nm||'_'||TRUNC(sysdate)||'.dmp'||chr(39)||'>>'||
                tablespace_nm||'.par'
        INTO ln_txt
        FROM dual;
        row_cntr:=row_cntr+1;
        insert_tab (file_no, row_cntr, ln_txt);
        SELECT
                'exp system/angler parfile='||tablespace_nm||'.par'
        INTO ln_txt
        FROM dual;
        row_cntr:=row_cntr+1;
        insert_tab (file_no, row_cntr, ln_txt);
        SELECT
                'compress '||tablespace_nm||'_'||TRUNC(sysdate)||'.dmp '
        INTO ln_txt
        FROM dual;
        row_cntr:=row_cntr+1;
        insert_tab (file_no, row_cntr, ln_txt);
        file_no:=file_no+1;
<<end_loop>>
NULL;
```

```
END LOOP;
CLOSE get_tbsp;
COMMIT;
END;
/
SET HEADING OFF FEEDBACK OFF LONG 4000 LINES 80 PAGES 0 VERIFY OFF
SET RECSEP OFF EMBEDDED ON ECHO OFF TERMOUT OFF
COLUMN        file#     NOPRINT
COLUMN        line_no NOPRINT
COLUMN        line_txt FORMAT a80    WORD_WRAPPED
SPOOL tablespace_export.sh
SELECT *  FROM exp_temp
ORDER BY file#,line_no;
SPOOL OFF
SET HEADING ON FEEDBACK ON LONG 2000 LINES 80 PAGES 22 VERIFY ON
SET RECSEP ON EMBEDDED OFF ECHO OFF TERMOUT ON
CLEAR COLUMNS
PROMPT  Procedure completed
```

<C>SQL Fragment to determine duplicate rows in the emp table, modify for use on other tables as needed:

```
DELETE FROM emp E
WHERE E.rowid > ( SELECT MIN (x.rowid)
                  FROM emp X
                  WHERE X.emp_no = E.emp_no );
```

<C>Script to get row size from an existing table:

```
rem  ***************************************************
rem  NAME: TB_RW_SZ.sql
rem  HISTORY:
rem  Date               Who                     What
rem  --------           -------------------     ------------
rem  01/20/93           Michael Brouillette   Creation
rem  FUNCTION:  Compute the average row size for a table.
rem  NOTES:  Currently requires DBA.
rem  INPUTS:
rem         tname  = Name of table.
rem         towner = Name of owner of table.
rem             cfile  = Name of output SQL Script file
rem  ***************************************************
COLUMN dum1             NOPRINT
COLUMN rsize            FORMAT 99,999.99
COLUMN rcount           FORMAT 999,999,999 newline
ACCEPT tname  PROMPT 'Enter table name: '
ACCEPT towner PROMPT 'Enter owner name: '
ACCEPT cfile  PROMPT 'Enter name for output SQL file: '
SET PAGESIZE 999  HEADING OFF  VERIFY OFF  TERMOUT OFF
SET FEEDBACK OFF  SQLCASE UPPER  NEWPAGE 3
```

```
SPOOL &cfile..sql
SELECT 0 dum1,
       'SELECT Table '||'&towner..&tname'||
      ' has '',COUNT(*) rcount,'' rows of '', ('
FROM dual
UNION
SELECT column_id,
      'SUM(NVL(VSIZE('||column_name||'),0)) + 1 +'
FROM dba_tab_columns
 WHERE table_name = '&tname' AND owner = '&towner'
   AND column_id <> (SELECT MAX(column_id)
                     FROM dba_tab_columns
                            WHERE table_name = '&tname'
                              AND owner = '&towner')
UNION
SELECT column_id,
      'SUM(NVL(VSIZE('||column_name||'),0)) + 1)'
FROM  dba_tab_columns
WHERE table_name = '&tname' AND owner = '&towner'
      AND column_id = (SELECT MAX(column_id)
                        FROM dba_tab_columns
                        WHERE table_name = '&tname'
                              AND owner = '&towner')
UNION
SELECT 997,  '/ COUNT(*) + 5 rsize, '' bytes each.'''
FROM dual
UNION
SELECT 999,  'from &towner..&tname.;'  FROM dual;
SPOOL OFF
SET TERMOUT ON  FEEDBACK 15   PAGESIZE 20
SET SQLCASE MIXED   NEWPAGE 1
START &cfile
CLEAR COLUMNS
UNDEF cfile
UNDEF tname
UNDEF towner
```

<C>PL/SQL--SQL--SQPLUS Script to generate a table rebuild script:

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

```
REM
SET VERIFY OFF FEEDBACK OFF TERMOUT OFF ECHO OFF PAGESIZE 0
SET TERMOUT ON
SELECT 'Creating table build script...' FROM dual;
SET TERMOUT OFF

CREATE TABLE t_temp
     (lineno NUMBER, tb_owner VARCHAR2(30), tb_name VARCHAR2(30),
      text VARCHAR2(2000))
/

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,freelists,
                       freelist_groups
            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_id;
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_freelists         user_tables.freelists%TYPE;
   lv_freelist_groups   user_tables.freelist_groups%TYPE;
   lv_first_rec         BOOLEAN;
   lv_lineno            NUMBER := 0;
   lv_string            VARCHAR2(2000);
   nul_cnt              NUMBER;
   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,lv_freelists,
                                lv_freelist_groups;
      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);
      nul_cnt:=0;
      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;
            IF ((lv_data_type = 'NUMBER') AND (lv_data_precision>0))
            THEN
             lv_string := lv_string || lower(lv_column_name) ||
                 ' ' || lv_data_type ||'('||lv_data_precision||','||
                           NVL(lv_data_scale,0)||')';
            ELSIF ((lv_data_type = 'FLOAT') AND (lv_data_precision>0))
            THEN
             lv_string := lv_string || lower(lv_column_name) ||
                 ' ' || lv_data_type ||'('||lv_data_precision||')';
            ELSE
             lv_string := lv_string || lower(lv_column_name) ||
                                ' ' || lv_data_type;
            END IF;
             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
                nul_cnt:=nul_cnt+1;
              lv_string := lv_string||
            ' CONSTRAINT ck_'||lv_table_name||'_'||nul_cnt||' 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 := 'FREELISTS ' || TO_CHAR(lv_freelists) ||
              ' FREELIST GROUPS ' || TO_CHAR(lv_max_trans);
      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;
SET VERIFY ON FEEDBACK ON TERMOUT ON
SET PAGESIZE 22
```

Code fragment to create a temporary table based on index_stats view and populate it with multiple indexe's statistics:

```
ACCEPT owner PROMPT 'Enter table owner name: '
ACCEPT table PROMPT 'Enter table name: '
SET HEADING OFF FEEDBACK OFF VERIFY OFF ECHO OFF RECSEP OFF PAGES 0
DEFINE cr = ‘chr(10)’
SPOOL index_sz.sql
SELECT 'CREATE TABLE stat_temp AS SELECT * FROM index_stats;'||&&cr||
'TRUNCATE TABLE stat_temp;'
FROM dual;
SELECT
'ANALYZE INDEX '||owner||'.'||index_name||' VALIDATE STRUCTURE;'||&&cr||
'INSERT INTO stat_temp SELECT * FROM index_stats;'||&&cr||
'COMMIT;'
FROM dba_indexes
WHERE owner=upper('&owner')
AND table_name=upper(‘&table’);
SPOOL OFF
@index_sz.sql
```

<C>Script to detect index browning:

```
rem
rem bowning.sql
rem purpose: Generate browing report from stat_temp
rem
rem MRA RevealNet/TreCom  5/28/97
rem
COLUMN del_lf_rows_len FORMAT 999,999,999 HEADING 'Deleted Bytes'
COLUMN lf_rows_len FORMAT 999,999,999 HEADING 'Filled Bytes'
COLUMN browning FORMAT 999.90 HEADING 'Percent|Browned'
SPOOL browning.lst
SELECT name,del_lf_rows_len,lf_rows_len,
(del_lf_rows_len/
((lf_rows_len+del_lf_rows_len),0,1,lf_rows_len+del_lf_rows_len))*100
browning
FROM stat_temp;
SPOOL OFF
```

<C>Script to check a proposed indexes size:

```
rem  ************************************************
rem  NAME:  IN_ES_SZ.sql
rem  HISTORY:
rem  Date               Who                    What
rem  --------           -------------------    ------------
rem  01/20/93           Michael Brouillette    Creation
rem  FUNCTION:  Compute the space used by an entry for an
rem   existing index.
rem  NOTES:  Currently requires DBA.
rem  INPUTS:
rem         tname  = Name of table.
rem         towner = Name of owner of table.
```

```
rem         clist  = List of columns enclosed in quotes.
rem                  i.e 'ename', 'empno'
rem              cfile  = Name of output SQL Script file
rem  ************************************************
COLUMN name NEW_VALUE   db NOPRINT
COLUMN dum1             NOPRINT
COLUMN isize            FORMAT 99,999.99
COLUMN rcount           FORMAT 999,999,999 newline
ACCEPT tname  PROMPT 'Enter table name: '
ACCEPT towner PROMPT 'Enter table owner name: '
ACCEPT clist  PROMPT 'Enter column list: '
ACCEPT cfile  PROMPT 'Enter name for output SQL file: '
SET PAGESIZE 999 HEADING OFF VERIFY OFF TERMOUT OFF
SET FEEDBACK OFF SQLCASE UPPER
SET NEWPAGE 3
SELECT name FROM v$database;
SPOOL rep_out/&db/propindx
SELECT -1 dum1,
       'SELECT ''Proposed Index on table ''||' FROM dual
UNION
SELECT 0,
       '''&towner..&tname'||' has '',COUNT(*)
rcount,'' entries of '', ('  FROM dual UNION
SELECT column_id,
      'SUM(NIL(vsize('||column_name||'),0)) + 1 +'
FROM dba_tab_columns
WHERE table_name = '&tname'
   AND owner = '&towner'
   AND column_name in (&clist)
   AND column_id <> (SELECT MAX(column_id)
                     FROM dba_tab_columns
                     WHERE table_name = '&tname'
                        AND owner = '&towner'
                        AND column_name IN (&clist))
UNION
SELECT column_id,
      'SUM(NIL(VSIZE('||column_name||'),0)) + 1)'
 FROM dba_tab_columns
 WHERE table_name = '&tname'
   AND owner = '&towner' AND column_name IN (&clist)
   AND column_id = (SELECT MAX(column_id)
                    FROM dba_tab_columns
                    WHERE table_name = '&tname'
                    AND owner = '&towner'
                    AND column_name IN (&clist)) UNION
SELECT 997, '/ COUNT(*) + 11 isize, '' bytes each.'''
FROM dual UNION
SELECT 999,
       'FROM &towner..&tname.;'  FROM dual;
SPOOL OFF
SET TERMOUT ON FEEDBACK 15 PAGESIZE 20 SQLCASE MIXED
SET NEWPAGE 1
START &cfile
CLEAR COLUMNS
```

<C>Script to calculate average size of existing index:

```
rem  ****************************************************
rem
rem  NAME:  IN_CM_SZ.sql
rem
rem  HISTORY:
rem  Date             Who              What
rem  --------  ------------------  ------------------------
rem  01/20/93  Michael Brouillette  Creation
rem
rem  FUNCTION:  Compute the space used by an entry for an
rem     existing index.
rem
rem  NOTES:  Currently requires DBA.
rem
rem  INPUTS:
rem         tname  = Name of table.
rem         towner = Name of owner of table.
rem         iname  = Name of index.
rem         iowner = Name of owner of index.
rem             cfile  = Name of output file SQL Script.
rem  ****************************************************
COLUMN dum1             NOPRINT
COLUMN isize            FORMAT 99,999.99
COLUMN rcount           FORMAT 999,999,999 newline
ACCEPT tname  PROMPT 'Enter table name: '
ACCEPT towner PROMPT 'Enter table owner name: '
ACCEPT iname  PROMPT 'Enter index name: '
ACCEPT iowner PROMPT 'Enter index owner name: '
ACCEPT cfile  PROMPT 'Enter name for output SQL file: '
SET PAGESIZE 999 HEADING OFF VERIFY OFF TERMOUT OFF
SET FEEDBACK OFF
SET SQLCASE UPPER NEWPAGE 3
SPOOL &cfile..sql
SELECT -1 dum1,
       'SELECT ''Index '||'&iowner..&iname'||' on table '
  FROM dual
UNION
SELECT 0,
       '&towner..&tname'||' has '',
       COUNT(*) rcount,'' entries of '', ('
  FROM dual
UNION
SELECT column_id,
      'SUM(NIL(vsize('||column_name||'),0)) + 1 +'
  FROM dba_tab_columns
 WHERE table_name = '&tname'
   AND owner = '&towner' AND column_name IN
                  (SELECT column_name FROM dba_ind_columns
                    WHERE table_name = '&tname'
                  AND table_owner = '&towner'
                  AND index_name = '&iname'
                  AND index_owner = '&iowner')
                  AND column_id <> (select max(column_id)
         FROM dba_tab_columns
    WHERE table_name = '&tname'
      AND owner = '&towner'
```

```
      AND column_name IN
               (SELECT column_name FROM dba_ind_columns
                 WHERE table_name = '&tname'
                   AND table_owner = '&towner'
                   AND index_name = '&iname'
                   AND index_owner = '&iowner'))
UNION
SELECT column_id,
       'SUM(NIL(vsize('||column_name||'),0)) + 1)'
  FROM dba_tab_columns
 WHERE table_name = '&tname' AND owner = '&towner'
   AND column_name IN
                   (SELECT column_name FROM dba_ind_columns
                     WHERE table_name = '&tname'
                       AND table_owner = '&towner'
                       AND index_name = '&iname'
                       AND index_owner = '&iowner')
   AND column_id = (SELECT MAX(column_id)
                      FROM dba_tab_columns
                     WHERE table_name = '&tname'
                           AND owner = '&towner'
       AND column_name IN
               (SELECT column_name FROM dba_ind_columns
                 WHERE table_name = '&tname'
                   AND table_owner = '&towner'
                   AND index_name = '&iname'
                   AND index_owner = '&iowner'))
UNION
SELECT 997,
       '/ COUNT(*) + 11 isize, '' bytes each.'''  from dual
UNION
SELECT 999,  'FROM &towner..&tname.;'  FROM dual;
SPOOL OFF
SET TERMOUT ON FEEDBACK 15 PAGESIZE 20 SQLCASE MIXED
SET NEWPAGE 1
START &cfile
CLEAR columns
UNDEF tname
UNDEF towner
UNDEF iname
UNDEF iowner
UNDEF cfile
```

<C>PL/SQL -- SQL -- SQLPLUS Script to create an index rebuild script:

```
REM
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 or Oracle8.
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' or partitioned indexes.
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 TERMOUT OFF FEEDBACK OFF ECHO OFF PAGES 0
SET TERMOUT ON
SELECT 'Creating index build script...' FROM dual;
SET TERMOUT OFF;

CREATE table i_temp
     (lineno NUMBER, id_name VARCHAR2(30),
      text VARCHAR2(2000)) STORAGE (INITIAL 100k NEXT 100k)
/


DECLARE
   CURSOR ind_cursor IS
    SELECT
             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
             user_indexes
     ORDER BY
             index_name;
   CURSOR col_cursor ( c_ind VARCHAR2, c_tab VARCHAR2) IS
      SELECT
            column_name
      FROM
            user_ind_columns
      WHERE
            index_name = c_ind
            AND table_name = c_tab
      ORDER BY
            column_position;
```

```
   lv_index_name        user_indexes.index_name%TYPE;
   lv_table_owner       user_indexes.table_owner%TYPE;
   lv_table_name        user_indexes.table_name%TYPE;
   lv_uniqueness        user_indexes.uniqueness%TYPE;
   lv_tablespace_name   user_indexes.tablespace_name%TYPE;
   lv_ini_trans         user_indexes.ini_trans%TYPE;
   lv_max_trans         user_indexes.max_trans%TYPE;
   lv_initial_extent    user_indexes.initial_extent%TYPE;
   lv_next_extent       user_indexes.next_extent%TYPE;
   lv_min_extents       user_indexes.min_extents%TYPE;
   lv_max_extents       user_indexes.max_extents%TYPE;
   lv_pct_increase      user_indexes.pct_increase%TYPE;
   lv_pct_free          user_indexes.pct_free%TYPE;
   lv_column_name       user_ind_columns.column_name%TYPE;
   lv_first_rec         BOOLEAN;
   lv_string            VARCHAR2(2000);
   lv_lineno            NUMBER := 0;

   PROCEDURE write_out(p_line INTEGER, p_name VARCHAR2,
                               p_string VARCHAR2) IS
   BEGIN
      INSERT INTO i_temp (lineno,id_name,text)
                  VALUES  (p_line,p_name,p_string);
   END;

BEGIN
   OPEN ind_cursor;
   LOOP
      FETCH ind_cursor INTO
            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;
      if (lv_uniqueness = 'UNIQUE') THEN
             lv_string:= 'CREATE UNIQUE INDEX '||LOWER(lv_index_name);
             write_out(lv_lineno, lv_index_name, lv_string);
             lv_lineno := lv_lineno + 1;
      ELSE
             lv_string:= 'CREATE INDEX ' || LOWER(lv_index_name);
             write_out(lv_lineno,  lv_index_name, lv_string);
             lv_lineno := lv_lineno + 1;
      END IF;
      OPEN col_cursor(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_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_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_name, lv_string);
      lv_lineno := lv_lineno + 1;
      lv_string := 'TABLESPACE ' || lv_tablespace_name || ' STORAGE (';
      write_out(lv_lineno,  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_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_name, lv_string);
      lv_lineno := lv_lineno + 1;
      lv_string := '/';
      write_out(lv_lineno,  lv_index_name, lv_string);
      lv_lineno := lv_lineno + 1;
      lv_lineno := lv_lineno + 1;
      lv_string:='                                                    ';
      write_out(lv_lineno,  lv_index_name, lv_string);
   END LOOP;
   CLOSE ind_cursor;
END;
/
COLUMN dbname NEW_VALUE db NOPRINT;
SELECT name dbname FROM v$database;
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_name, lineno;
rem
```

```
SPOOL OFF
rem
DROP TABLE i_temp;
SET VERIFY ON TERMOUT ON FEEDBACK ON ECHO ON PAGES 22
CLEAR COLUMNS
```

<C>SQL -- PL/SQL Script to create a synonym rebuild script:

```
REM FUNCTION: SCRIPT FOR CREATING SYNONYMS
REM           This script must be run by a user with the DBA role.
REM           This script is intended to run with Oracle7 or Oracle8.
REM           Running this script will in turn create a script to build
REM           all the synonyms in the database. The created script,
REM           create_synonyms.sql, can be run by any user with the DBA
REM           role or with the 'CREATE ANY SYNONYM' and 'CREATE PUBLIC
REM           SYNONYM' system privileges.
REM NOTE:    This script does not capture synonyms for tables owned
REM          by the 'SYS' user.
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
SET TERMOUT ON
SELECT 'Creating synonym build script...' FROM dual;
SET TERMOUT OFF
COLUMN dbname NEW_VALUE db NOPRINT
SELECT name dbname FROM v$database;
DEFINE cr='chr(10)'
SPOOL rep_out\&db\crt_syns.sql

SELECT 'CREATE '|| DECODE(owner,'PUBLIC','PUBLIC ',NULL) ||
       'SYNONYM ' || DECODE(owner,'PUBLIC',NULL, owner || '.') ||
        LOWER(synonym_name) || ' FOR ' || LOWER(table_owner) ||
        '.' || LOWER(table_name) ||
        DECODE(db_link,NULL,NULL,'@'||db_link) || ';'
  FROM sys.dba_synonyms
  WHERE table_owner != 'SYS'
  ORDER BY owner
/
SPOOL OFF
SET VERIFY ON FEEDBACK ON TERMOUT ON PAGESIZE 22
CLEAR COLUMNS
UNDEF cr
```

<C>PL/SQL--SQL-SQLPLUS Script to create a Sequence Rebuild Script

```
REM
REM    FUNCTION: SCRIPT FOR RE-CREATING DATABASE SEQUENCES
REM
REM              This script must be run by a user with select
REM              grant on DBA_SEQUENCES
```

```
REM              This script is intended to run with Oracle7 or 8.
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. Test
REM it completely before relying on it.
REM

SET VERIFY OFF FEEDBACK OFF TERMOUT OFF ECHO OFF PAGESIZE 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 name dbname FROM v$database;
SPOOL rep_out\&db\crt_seq.sql
BREAK ON downer SKIP 1
COLUMN text FORMAT a60 WORD_WRAP
COLUMN downer NOPRINT
SELECT
      grantor_owner downer,
      text
FROM
      seq_temp
ORDER BY
      downer
/
SPOOL OFF
rem
DROP TABLE seq_temp;
SET TERMOUT ON VERIFY ON FEEDBACK ON
CLEAR COLUMNS
CLEAR BREAKS
PROMPT Finished build
```

<C>PL/SQL--SQL--SQLPLUS Script to create a Database Link Rebuild script:

```
REM
REM NAME:        link_rct.sql
REM
REM FUNCTION:    SCRIPT FOR RE-CREATING DATABASE LINKS
```

```
REM
REM              This script must be run by users with select
REM              grant on dba_db_links.
REM
REM              This script is intended to run with Oracle7 or 8.
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 on 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 executing the script, you must add each user's password to
REM the connect clause.  Duplicate connect clauses can be eliminated by
REM being sure that the database link is being created under the correct
REM schema.
REM
REM The PUBLIC database links will require a connect as 'SYS'.  However,
REM this username can be changed to any user with the DBA role or with
REM the '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. Test
REM it completely before relying on it.
REM

SET VERIFY OFF FEEDBACK OFF TERMOUT OFF ECHO OFF PAGES 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));

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;
/
DEFINE cr = CHR(10)
COLUMN dbname NEW_VALUE db NOPRINT
SELECT name dbname FROM v$database;
SPOOL rep_out\&db\crt_dbls.sql
BREAK ON downer SKIP 1
COLUMN text FORMAT A60 WORD_WRAP

SELECT
   'CONNECT '||DECODE(grantor_owner,'PUBLIC','SYS',grantor_owner)
||'/'||DECODE(grantor_owner,'PUBLIC','SYS',grantor_owner) downer||&&cr||
RTRIM(text)
FROM
     dl_temp
ORDER BY
     downer
```

```
/
SPOOL OFF
SET VERIFY ON FEEDBACK ON TERMOUT ON PAGES 22
CLEAR COLUMNS
DROP TABLE dl_temp;
```

<C>SQL Script to create a View Recreate script:

```
REM
REM NAME        :view_rct.sql
REM FUNCTION:recreate 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: select
REM               max(text_length) from dba_views;
REM
SET PAGES 59 LINES 79 FEEDBACK OFF ECHO OFF VERIFY OFF
DEFINE cr='chr(10)'
COLUMN text       FORMAT a80 word_wrapped
COLUMN view_name  FORMAT a20
COLUMN dbname NEW_VALUE db NOPRINT
UNDEF owner_name
UNDEF view_name
SELECT name dbname from v$database;
SET LONG 5000 HEADING OFF
SPOOL rep_out\&db\cre_view.sql
SELECT
'rem Code for view: '||v.view_name||'Instance: '||&&db||&&cr||
'CREATE OR REPLACE VIEW '||v.owner||'.'||v.view_name||' AS '||&&cr,
      v.text
FROM
      dba_views v
WHERE
      v.owner LIKE UPPER('%&&owner_name%')
      AND view_name LIKE UPPER('%&&view_name%')
ORDER BY
      v.view_name;
SPOOL OFF
SET HEADING ON PAGES 22 LINES 80 FEEDBACK ON
CLEAR COLUMNS
TTITLE OFF
PAUSE Press enter to continue
```

<C>PL/SQL--SQL--SQLPLUS Script to Rebuild Triggers

```
REM  trig_rct.sql
REM
```

```
REM    FUNCTION: SCRIPT FOR RE-CREATING DATABASE TRIGGERS
REM       This script can be run by anyone with access to dba_ triggers
REM              This script is intended to run with Oracle7.
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 'create_triggers.sql'.
REM              Only preliminary testing of this script was performed.
REM              Be sure to test it completely before relying on it.
SET VERIFY OFF FEEDBACK OFF TERMOUT OFF ECHO OFF PAGES 0 LONG 4000
SET TERMOUT ON ARRAYSIZE 1
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 ( owner VARCHAR2, name VARCHAR2 ) IS
      SELECT
            trigger_owner,
            trigger_name,
            column_name
      FROM
            dba_trigger_cols
      WHERE
            trigger_owner = owner AND
            trigger_name = 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(trig_owner,trig_name);
      LOOP
            FETCH trig_col INTO
                              trig_col_own,
                              trig_col_nam,
                              trig_column;
            EXIT WHEN trig_col%NOTFOUND;
                  counter := counter+1;
                  IF counter = 1 THEN
                  all_columns := ' OF '||all_columns||trig_column;
                  ELSE
                  all_columns := all_columns||', '||trig_column;
            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;
```

```
/
COLUMN dbname NEW_VALUE db NOPRINT
SELECT name dbname FROM v$database;
DEFINE cr='CHR(10)'
SPOOL rep_out\&db\crt_trgs.sql
SET HEADING OFF
SET RECSEP OFF PAGES 0

SELECT '/'||&&cr||&&cr||'CREATE OR REPLACE TRIGGER
'||owner||'.'||trigger_name||&&cr||
DECODE(trigger_type,'BEFORE EACH ROW','BEFORE ',
      'AFTER EACH ROW','AFTER ',trigger_type)||
triggering_event||&&cr||
trigger_columns||&&cr||
table_owner||'.'||table_name||' '||referencing_names||&&cr||
DECODE(trigger_type,'BEFORE EACH ROW','FOR EACH ROW',
            'AFTER EACH ROW','FOR EACH ROW','')||&&cr||
when_clause,
trigger_body
FROM trig_temp
ORDER BY owner;
SPOOL OFF
DROP TABLE trig_temp;
SET VERIFY ON FEEDBACK ON TERMOUT ON PAGESIZE 22
SET HEADING ON RECSEP ON
CLEAR COLUMNS
```

<C>SQL Script to show invalid objects in database:

```
rem Name: inv_obj.sql
rem Purpose: Show alll invalid objects in database
rem Mike Ault 7/2/96 TreCom/RevealNet
rem
COLUMN object_name      FORMAT A30 HEADING 'Object|Name'
COLUMN owner            FORMAT a10 HEADING 'Object|Owner'
COLUMN last_time        FORMAT a20 HEADING 'Last Change|Date'
SET LINES 80 FEEDBACK OFF PAGES 0 VERIFY OFF
START title80 'Invalid Database Objects'
SPOOL rep_out/&db/inv_obj
SELECT
      owner,
      object_name,
      object_type,
      TO_CHAR(last_ddl_time,'DD-MON-YY hh:mi:ss') Last_time
FROM
      dba_objects
WHERE
      status='INVALID'
/
PAUSE Press enter to continue
SET LINES 80 FEEDBACK ON PAGES 22 VERIFY ON
CLEAR COLUMNS
TTITLE OFF
```

<C>SQL Script to Monitor Table Structure

```
REM
REM NAME        : TABLE.SQL
REM FUNCTION    : GENERATE TABLE REPORT
REM Limitations : None
REM MRA 6/12/97 Updated to ORACLE8
REM
CLEAR COLUMNs
COLUMN owner                        FORMAT a15  HEADING 'Table|Owner'
COLUMN table_name                               HEADING Table
COLUMN tablespace_name              FORMAT A15  HEADING Tablespace
COLUMN pct_increase                             HEADING 'Pct|Increase'
COLUMN init                                     HEADING 'Initial|Extent'
COLUMN next                                     HEADING 'Next|Extent'
COLUMN partitioned                  FORMAT a15  HEADING 'Partitioned?'
BREAK ON owner ON tablespace_name
SET PAGES 48 LINES 132
START title132 "ORACLE TABLE REPORT"
SPOOL rep_out\&db\tab_rep
SELECT
            owner,
            tablespace_name,
            table_name,
            initial_extent Init,
            next_extent Next,
            pct_increase,
            partitioned
FROM
            sys.dba_tables
WHERE
            owner NOT IN ('SYSTEM', 'SYS')
ORDER BY
            owner,
            tablespace_name,
            table_name;
SPOOL OFF
CLEAR COLUMNS
PAUSE Press enter to continue
SET PAGES 22 LINES 80
TTITLE OFF
CLEAR COLUMNS
CLEAR BREAKS
```

<C>SQL Script to Monitor Extents

```
REM
REM NAME                : EXTENTS.SQL
REM FUNCTION            : GENERATE EXTENTS REPORT
REM USE                 : FROM SQLPLUS OR OTHER FRONT END
REM LIMITATIONS         : NONE
REM
CLEAR COLUMNS
COLUMN segment_name           HEADING 'Segment'     FORMAT A15
COLUMN tablespace_name        HEADING 'Tablespace'  FORMAT A10
COLUMN owner                  HEADING 'Owner'       FORMAT A10
COLUMN segment_type           HEADING 'Type'        FORMAT A10
COLUMN size                   HEADING 'Size'        FORMAT 999,999,999
COLUMN extents                HEADING 'Current|Extents'
COLUMN max_extents            HEADING 'Max|Extents'
COLUMN bytes                  HEADING 'Size|(Bytes)'
SET PAGESIZE 58 NEWPAGE 0 LINESIZE 130 FEEDBACK OFF
SET ECHO OFF VERIFY OFF
ACCEPT extents PROMPT 'Enter max number of extents: '
BREAK ON tablespace_name SKIP PAGE ON owner
START TITLE132 "Extents Report"
DEFINE output = rep_out\&db\extent
SPOOL &output
SELECT  tablespace_name,
      segment_name,
      extents,
      max_extents,
      bytes,
      owner "owner",
      segment_type
FROM    dba_segments
WHERE extents >= &extents AND owner LIKE UPPER('%&owner%')
ORDER BY tablespace_name,owner,segment_type,segment_name;
SPOOL OFF
CLEAR COLUMNS
CLEAR BREAKS
SET TERMOUT ON FEEDBACK ON VERIFY ON
UNDEF extents
UNDEF owner
TTITLE OFF
UNDEF OUTPUT
PAUSE Press enter to continue
```

<C>SQL Script to Generate an Actual Size Report for Indexes and Tables

```
rem  ************************************************************
rem
rem  NAME: ACT_SIZE.sql
rem
rem  HISTORY:
rem  Date             Who              What
rem  --------  ------------------  ----------------------------------
rem  09/??/90  Maurice C. Manton    Creation for IOUG
rem  12/23/92  Michael Brouillette  Assume TEMP_SIZE_TABLE exists.Use
rem  DBA info.
rem  Prompt for user name. Spool file = owner.
```

```
rem   07/15/96  Mike Ault Updated for Oracle 7.x, added indexes
rem   06/12/97  Mike Ault Updated for Oracle 8.x (use DBMS_ROWID)
rem  FUNCTION:  Will show actual blocks used vs allocated for all tables
rem  for a user.
rem  INPUTS:  owner = Table owner name.
rem  ********************************************************
ACCEPT owner PROMPT 'Enter table owner name: '
SET HEADING OFF FEEDBACK OFF VERIFY OFF ECHO OFF RECSEP OFF PAGES 0
COLUMN db_block_size NEW_VALUE blocksize NOPRINT
TTITLE OFF
DEFINE cr='chr(10)'
DEFINE qt='chr(39)'
TRUNCATE TABLE temp_size_table;
SELECT value db_block_size FROM v$parameter WHERE name='db_block_size';
SPOOL fill_sz.sql
SELECT
 'INSERT INTO temp_size_table'||&&cr||
 'SELECT '||&&qt||segment_name||&&qt||&&cr||
 ',COUNT(DISTINCT(dbms_rowid.rowid_block_number(rowid))) blocks'||&&cr||
 'FROM &&owner..'||segment_name, ';'
FROM
  dba_segments
WHERE
  segment_type ='TABLE'
  AND owner = UPPER('&owner');
SPOOL OFF
SPOOL index_sz.sql
SELECT
    'CREATE TABLE stat_temp AS SELECT * FROM index_stats;'||&&cr||
    'TRUNCATE TABLE stat_temp;'
FROM
    dual;
SELECT
'ANALYZE INDEX '||owner||'.'||index_name||' VALIDATE STRUCTURE;'||&&cr||
'INSERT INTO stat_temp SELECT * FROM index_stats;'||&&cr||
'COMMIT;'
FROM
   dba_indexes
WHERE
   owner=UPPER('&owner');
SPOOL OFF
SET FEEDBACK ON TERMOUT ON LINES 132
START index_sz.sql
INSERT INTO temp_size_table SELECT name,trunc(used_space/&&blocksize)
FROM stat_temp;
DROP TABLE stat_temp;
DEFINE temp_var = &&qt;
START fill_sz
HOST rm fill_size_table.sql
DEFINE bs = '&&blocksize K'
COLUMN t_date           NOPRINT NEW_VALUE t_date
COLUMN user_id          NOPRINT NEW_VALUE user_id
COLUMN segment_name     FORMAT A25         HEADING "SEGMENT|NAME"
COLUMN segment_type     FORMAT A7          HEADING "SEGMENT|TYPE"
COLUMN extents          FORMAT 999         HEADING "EXTENTS"
COLUMN kbytes           FORMAT 999,999,999 HEADING "KILOBYTES"
COLUMN blocks           FORMAT 9,999,999   HEADING "ALLOC.|&&bs|BLOCKS"
COLUMN act_blocks       FORMAT 9,999,990   HEADING "USED|&&bs|BLOCKS"
```

```
COLUMN pct_block        FORMAT 999.99      HEADING "PCT|BLOCKS|USED"
START title132 "Actual Size Report for &owner"
SET PAGES 55
BREAK ON REPORT ON segment_type SKIP 1
COMPUTE SUM OF kbytes ON segment_type REPORT
SPOOL rep_out\&db\&owner
SELECT
            segment_name,
            segment_type,
            SUM(extents) extents,
            SUM(bytes)/1024 kbytes,
            SUM(a.blocks) blocks,
            NVL(MAX(b.blocks),0) act_blocks,
            (MAX(b.blocks)/SUM(a.blocks))*100 pct_block
 FROM
            sys.dba_segments a,
            temp_size_table b
 WHERE
            segment_name = UPPER( b.table_name )
 GROUP BY
            segment_name,
            segment_type
 ORDER BY
            segment_type,
            segment_name;
SPOOL OFF
TRUNCATE TABLE temp_size_table;
SET TERMOUT ON FEEDBACK 15 VERIFY ON PAGESIZE 20 LINESIZE 80 SPACE 1
UNDEF qt
UNDEF cr
TTITLE OFF
CLEAR COLUMNS
CLEAR COMPUTES
PAUSE press enter to continue
```

The script shown above to calculate the actual size of a table or index uses the TEMP_SIZE_TABLE which is created with the script shown below. As shown the act_size script will only work with ORACLE8. To use act_size with ORACLE7 replace the call to the dbms_rowid.rowid_block_number procedure with : SUBSTR(ROWID,1,8).

```
rem
rem Create temp_size_table for use by actsize.sql
rem
CREATE TABLE temp_size_table (
      table_name VARCHAR2(64),
      blocks NUMBER);
```

<C>SQL Script to document table statistics

```
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
rem  FUNCTION:  tables or all tables.
rem
SET PAGES 56 LINES 132 NEWPAGE 0 VERIFY OFF ECHO OFF
SET 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 "Percent|Full"
COLUMN chain_cnt         FORMAT 999,999 HEADING "Chains"
COLUMN avg_row_len       FORMAT 999,999 HEADING "Avg|Length (Bytes)"
rem
START title132 "Table Statistics Report"
SPOOL report_output/&&db/tab_stat
rem
SELECT
            owner,
            table_name,
            tablespace_name,
            num_rows,
            blocks,
            empty_blocks,
            1-((blocks * avg_space)/(blocks * 2048)) space_full,
            chain_cnt,
            avg_row_len
FROM
            dba_tables
WHERE
            owner = UPPER('&owner')
            AND tablespace_name = UPPER('&tablespace')
ORDER BY
            owner,
            tablespace_name;
SPOOL OFF
```

<C>SQL Script to document new table features in ORACLE8:

```
REM
REM     Name:      tab_rep.sql
REM     FUNCTION:  Document table extended parameters
REM     Use:       From SQLPLUS
REM   MRA 6/13/97 Created for ORACLE8
REM
COLUMN owner            FORMAT a10 HEADING 'Owner'
COLUMN table_name       FORMAT a15 HEADING 'Table'
COLUMN tablespace_name  FORMAT a12 HEADING 'Tablespace'
COLUMN table_type_owner FORMAT a10 HEADING 'Type|Owner'
COLUMN table_type       FORMAT a13 HEADING 'Type'
COLUMN iot_name         FORMAT a10 HEADING 'IOT|Overflow'
COLUMN iot_type         FORMAT a12 HEADING 'IOT or|Overflow'
COLUMN nested           FORMAT a6  HEADING 'Nested'
SET LINES 130 VERIFY OFF FEEDBACK OFF PAGES 58
START title132 'Extended Table Report'
SPOOL rep_out\&&db\ext_tab.lis
SELECT
      owner,
      table_name,
      tablespace_name,
      iot_name,
      logging,
      partitioned,
      iot_type,
      table_type_owner,
      table_type,
      packed,
      temporary,
      nested
FROM
      dba_tables
WHERE
      owner LIKE UPPER('%&owner%');
SPOOL OFF
SET VERIFY ON LINES 80 PAGES 22 FEEDBACK ON
```

<C>SQL Script to document a tables columns:

```
rem
rem tab_col.sql
rem
rem FUNCTION: Report on Table and View Column Definitions
rem
rem MRA 9/18/96
rem MRA 6/14/97 Added table level selectivity
rem
COLUMN owner            FORMAT a10  HEADING Owner
COLUMN table_name       FORMAT a30  HEADING "Table or View Name"
COLUMN COLUMN_name      FORMAT a32  HEADING "Table or View Column"
COLUMN data_type        FORMAT a15  HEADING "Data|Type"
COLUMN data_length                  HEADING Length
COLUMN nullable         FORMAT a5   HEADING Null?
BREAK ON owner ON table_name SKIP 1
SET LINES 132 PAGES 48 FEEDBACK OFF VERIFY OFF
```

```
START title132 "Table Columns Report"
SPOOL rep_out/&db/tab_col
SELECT
      a.owner,
      table_name||' '||object_type table_name,
      column_name,
      data_type,
      data_length,
      DECODE(nullable,'N','NO','YES') nullable
FROM
      dba_tab_columns a, dba_objects b
WHERE
      a.owner NOT IN ('SYS','SYSTEM') AND
      a.owner=UPPER('&owner') AND
      a.owner=b.owner AND
      a.table_name LIKE UPPER('%&table%') AND
      a.table_name=b.object_name AND
      object_type IN ('TABLE','VIEW','CLUSTER')
ORDER BY
      owner,
      object_type,
      table_name,
      column_id
/
SPOOL OFF
TTITLE OFF
SET LINES 80 PAGES 22 FEEDBACK ON VERIFY ON
```

<C>PL/SQL-SQL-SQLPLUS Script to report on Primary Key - Foreign Key

relationships:

```
REM FUNCTION: SCRIPT FOR DOCUMENTING 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 or Oracle8.
REM
REM FUNCTION: Running this script will document the
REM FUNCTION: primary key - foreign key
REM FUNCTION: constraints in the database
REM
REM
REM Only preliminary testing of this script was performed.
REM Be sure to test
REM it completely before relying on it.
REM
REM MRA 6/14/97 Verified for Oracle8
REM
SET ARRAYSIZE 1 VERIFY OFF  FEEDBACK OFF TERMOUT OFF ECHO OFF PAGESIZE 0
SET LONG 4000
SET TERMOUT ON
SELECT 'Creating constraint documentation script...' FROM dual;
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);
TRUNCATE TABLE cons_temp;

DECLARE

   CURSOR cons_cursor IS
     SELECT
            owner,
            constraint_name,
            DECODE(constraint_type,'P','Primary Key',
                                   'R','Foreign Key',
                                   'U','Unique',
                                   'C','Check',
                                   'D','Default'),
            search_condition,
            table_name,
            r_owner,
            r_constraint_name,
            delete_rule
       FROM
            user_constraints
       WHERE
            owner NOT IN ('SYS','SYSTEM')
       ORDER BY
            owner;

     CURSOR cons_col (cons_name in VARCHAR2) IS
       SELECT
            owner,
            constraint_name,
            column_name
       FROM
            user_cons_columns
       WHERE
            owner NOT IN ('SYS','SYSTEM') AND
            constraint_name = UPPER(cons_name)
       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 (cons_name);
            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_sc,
                                          cons_tname,
                                          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;
/
CREATE INDEX pk_cons_temp ON cons_temp(constraint_name);
CREATE INDEX lk_cons_temp2 ON cons_temp(referenced_constraint);
SET FEEDBACK OFF TERMOUT OFF ECHO OFF
```

```
SET VERIFY OFF
SET PAGES 48 LINES 132
COLUMN pri_own FORMAT a10 HEADING 'Pri Table|Owner'
COLUMN for_own FORMAT a10 HEADING 'For Table|Owner'
COLUMN pri_tab FORMAT a25 HEADING 'Pri Table|Name'
COLUMN for_tab FORMAT a25 HEADING 'For Table|Name'
COLUMN pri_col FORMAT a25 HEADING 'Pri Key|COLUMNs' word_wrapped
COLUMN for_col FORMAT a25 HEADING 'For Key|COLUMNs' word_wrapped
START title132 'Primary Key - Foreign Key Report'
SPOOL rep_out\&db\pk_fk
BREAK ON pri_own ON pri_tab ON for_own ON for_tab
SELECT
            b.owner pri_own,
            b.table_name pri_tab,
            RTRIM(b.constraint_columns) pri_col,
            a.owner for_own,
            a.table_name for_tab,
            RTRIM(a.constraint_columns) for_col
FROM
            cons_temp a,
            cons_temp b
WHERE
            a.referenced_constraint=b.constraint_name
ORDER BY
            b.owner,b.table_name,a.owner,a.table_name;
SPOOL OFF
DROP TABLE cons_temp;
SET VERIFY ON FEEDBACK ON TERMOUT ON PAGESIZE 22 LINES 80
CLEAR COLUMNS
CLEAR BREAKS
TTITLE OFF
```

<C>SQL Script to Determine Chained Rows without doing ANALYZE

```
rem  ****************************************************************
rem
rem  NAME:     CHAINING.sql
rem
rem  FUNCTION: Report on the number of CHAINED rows within a named table
rem
rem  NOTES:  Requires DBA priviledges.
rem      The target table must have a column that is the leading portion
rem        of an index and is defined as not null.
rem        Uses the V$SESSTAT table where USERNAME is the current user.
rem        A problem if > 1 session active with that USERID.
rem        The statistics in V$SESSTAT may change between releases and
rem        platforms.  Make sure that 'table fetch continued row' is
rem        a valid statistic. Table must have primary or unique index
rem        This routine can be run by AUTO_CHN.sql by remarking the two
rem        accepts and un-remarking the two defines.
rem
rem  INPUTS: obj_own = the owner of the table.
rem          obj_nam = the name of the table.
```

```
rem
rem  ***************************************************************

ACCEPT obj_own PROMPT 'Enter the table owner''s name: '
ACCEPT obj_nam PROMPT 'Enter the name of the table: '

rem DEFINE obj_own = &1   <-- Remove comment to use with auto_chain
rem DEFINE obj_nam = &2   <-- Remove comment to use with auto_chain

SET TERMOUT OFF FEEDBACK OFF VERIFY OFF ECHO OFF HEADING OFF EMBEDDED ON
COLUMN statistic# NEW_VALUE stat_no NOPRINT
SELECT
            statistic#
FROM
            v$statname

WHERE
            n.name = 'table fetch continued row'
/
rem  Find out who we are in terms of sid
COLUMN sid NEW_VALUE user_sid
SELECT
            distinct sid
FROM
            v$session
WHERE
            audsid = USERENV('SESSIONID')
/

rem  Find the last col of the table and a not null indexed column
COLUMN column_name      NEW_VALUE last_col
COLUMN name             NEW_VALUE indexed_column
COLUMN value            NEW_VALUE before_count
SELECT
            column_name
  FROM
            dba_tab_columns
 WHERE
            table_name = upper('&&obj_nam')
            and owner = upper('&&obj_own')
 ORDER BY
            column_id
/
SELECT
            c.name
  FROM
            sys.col$ c,
            sys.obj$ idx,
            sys.obj$ base,
            sys.icol$ ic
 WHERE
            base.obj#       = c.obj#
            and ic.bo#      = base.obj#
            and ic.col#     = c.col#
            and base.owner# = (SELECT user# FROM sys.user$
                               WHERE name = UPPER('&&obj_own'))
            and ic.obj#     = idx.obj#
            and base.name   = UPPER('&&obj_nam')
```

```
            and ic.pos#     = 1
            and c.null$     > 0
/
SELECT value
  FROM v$sesstat
 WHERE v$sesstat.sid = &user_sid
   AND v$sesstat.statistic# = &stat_no
/
rem  Select every row from the target table
SELECT &last_col xx
  FROM &obj_own..&obj_nam
 WHERE &indexed_column <= (SELECT MAX(&indexed_column)
                                     FROM &obj_own..&obj_nam)
/
COLUMN value NEW_VALUE after_count
SELECT value
  FROM v$sesstat
 WHERE v$sesstat.sid = &user_sid
   AND v$sesstat.statistic# = &stat_no
/
SET TERMOUT ON

SELECT
'Table '||UPPER('&obj_own')||'.'||UPPER('&obj_nam')||' contains '||
       (TO_NUMBER(&after_count) - TO_NUMBER(&before_count))||
       ' chained row'||
       DECODE(to_NUMBER(&after_count) -
TO_NUMBER(&before_count),1,'.','s.')
  FROM dual
 WHERE RTRIM('&indexed_column') IS NOT NULL
/

rem If we don't have an indexed column this won't work so say so
SELECT 'Table '||
       UPPER('&obj_own')||'.'||UPPER('&obj_nam')||
      ' has no indexed, not null columns.'
  FROM dual
 WHERE RTRIM('&indexed_column') IS NULL
/

SET TERMOUT ON FEEDBACK 15 VERIFY ON PAGESIZE 20 LINESIZE 80 SPACE 1
SET HEADING ON
UNDEF obj_nam
UNDEF obj_own
UNDEF before_count
UNDEF after_count
UNDEF indexed_column
UNDEF last_col
UNDEF stat_no
UNDEF user_sid
CLEAR COLUMNS
CLEAR COMPUTES
```

<C>SQL Script to automate generation of chained row reports for a user:

```
rem  ****************************************************************
rem
rem  NAME: AUTO_CHN.sql
rem
rem  FUNCTION: Run CHAINING.sql for all of a specified users tables.
rem
rem  NOTES:Requires a minor mod to CHAINING.sql. See CHAINING.sql header
rem
rem  INPUTS:
rem            tabown = Name of owner.
rem
rem  ****************************************************************
rem
ACCEPT tabown PROMPT 'Enter table owner: '
rem
SET TERMOUT OFF FEEDBACK OFF VERIFY OFF ECHO OFF HEADING OFF PAGES 999
SET EMBEDDED ON
COLUMN name NEW_VALUE db NOPRINT
SELECT name FROM v$database;
SPOOL rep_out\auto_chn.gql
rem
SELECT 'start chaining &tabown '||table_name
  FROM dba_tables
 WHERE owner = UPPER('&tabown')
/

SPOOL OFF
SPOOL rep_out\&db\chaining
START rep_out\auto_chn.gql
SPOOL OFF
UNDEF tabown
SET TERMOUT ON FEEDBACK 15 VERIFY ON PAGESIZE 20 LINESIZE 80 SPACE 1
SET EMBEDDED OFF
HO del rep_out\auto_chn.gql
PAUSE Press enter to continue
```

<C>SQL Script to report on table grants

```
rem****************************************************************
rem  NAME: db_tgnts.sql
rem
rem  FUNCTION: Produce report of table or procedure grants showing
rem  GRANTOR, GRANTEE or ROLE and specific GRANTS.
rem
rem  INPUTS: Owner name
rem ****************************************************************
rem
COLUMN grantee          FORMAT A18  HEADING "Grantee|or Role"
COLUMN owner            FORMAT A18  HEADING "Owner"
COLUMN table_name       FORMAT A30  HEADING "Table|or Proc"
COLUMN grantor          FORMAT A18  HEADING "Grantor"
COLUMN privilege        FORMAT A10  HEADING "Privilege"
COLUMN grantable        FORMAT A19  HEADING "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 = report_output/&&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
```

<C>SQL Script to Report on Partitioned Tables

```
rem
rem Name: tab_part.sql
rem Function : Report on partitioned table structure
rem History: MRA 6/13/97 Created
rem
COLUMN table_owner      FORMAT a10 HEADING 'Owner'
COLUMN table_name       FORMAT a15 HEADING 'Table'
COLUMN partition_name   FORMAT a15 HEADING 'Partition'
COLUMN tablespace_name  FORMAT a15 HEADING 'Tablespace'
COLUMN high_value       FORMAT a10 HEADING 'Partition|Value'
SET LINES 78
START title80 'Table Partition Files'
BREAK ON table_owner ON table_name
SPOOL rep_out/&&db/tab_part.lis
SELECT
      table_owner,
      table_name,
      partition_name,
      high_value,
      tablespace_name,
      logging
FROM sys.dba_tab_partitions
ORDER BY table_owner,table_name
/
```

```
SPOOL OFF
```

<C>SQL Script to report on partitioned table storage characteristics:

```
rem
rem NAME:   Tab_pstor.sql
rem FUNCTION: Provide data on partitioned table storage charcacteristics
rem HISTORY: MRA 6/13/97 Created
rem
COLUMN table_owner      FORMAT a6         HEADING 'Owner'
COLUMN table_name       FORMAT a14        HEADING 'Table'
COLUMN partition_name   FORMAT a9         HEADING 'Partition'
COLUMN tablespace_name  FORMAT a11        HEADING 'Tablespace'
COLUMN pct_free         FORMAT 9999       HEADING '%|Free'
COLUMN pct_used         FORMAT 999        HEADING '%|Use'
COLUMN ini_trans        FORMAT 9999       HEADING 'Init|Tran'
COLUMN max_trans        FORMAT 9999       HEADING 'Max|Tran'
COLUMN initial_extent   FORMAT 9999999    HEADING 'Init|Extent'
COLUMN next_extent      FORMAT 9999999    HEADING 'Next|Extent'
COLUMN max_extent                         HEADING 'Max|Extents'
COLUMN pct_increase     FORMAT 999        HEADING '%|Inc'
COLUMN partition_position     FORMAT 9999 HEADING 'Part|Nmbr'
SET LINES 130
START title132 'Table Partition File Storage'
BREAK ON table_owner on table_name
SPOOL rep_out/&&db/tab_pstor.lis
SELECT
      table_owner,
      table_name,
      tablespace_name,
      partition_name,
      partition_position,
      pct_free,
      pct_used,
      ini_trans,
      max_trans,
      initial_extent,
      next_extent,
      max_extent,
      pct_increase
FROM sys.dba_tab_partitions
ORDER BY table_owner,table_name
/
SPOOL OFF
```

<C>SQL Script to report on Nested Tables

```
rem
rem NAME: tab_nest.sql
rem PURPOSE: Report on Nested Tables
rem HISTORY: MRA 6/14/97 Created
rem
COLUMN owner                  FORMAT a10 HEADING 'Owner'
COLUMN table_name             FORMAT a20 HEADING 'Store Table'
COLUMN table_type_owner       FORMAT a10 HEADING 'Type|Owner'
COLUMN table_type_name        FORMAT a15 HEADING 'Type|Name'
COLUMN parent_table_name      FORMAT a25 HEADING 'Parent|Table'
COLUMN parent_table_column    FORMAT a15 HEADING 'Parent|Column'
SET PAGES 58 LINES 132 VERIFY OFF FEEDBACK OFF
START title132 'Nested Tables'
BREAK ON owner
SPOOL rep_out\&db\tab_nest.lis
SELECT
      owner,
      table_name,
      table_type_owner,
      table_type_name,
      parent_table_name,
      parent_table_column
FROM sys.dba_nested_tables
ORDER BY owner;
SPOOL OFF
```

<C>SQL Script to report on indexes:

```
rem
rem NAME: ind_rep.sql
rem FUNCTION: Report on indexes
rem HISTORY: MRA 6/14/97 Creation
rem
COLUMN owner                        FORMAT a8   HEADING 'Index|Owner'
COLUMN index_name                   FORMAT a27  HEADING 'Index'
COLUMN index_type                   FORMAT a6   HEADING 'Type|Index'
COLUMN table_owner                  FORMAT a8   HEADING 'Table|Owner'
COLUMN table_name                   FORMAT a24  HEADING 'Table Name'
COLUMN table_type                   FORMAT a10  HEADING 'Table|Type'
COLUMN uniqueness                   FORMAT a1   HEADING 'U|n|i|q|u|e'
COLUMN tablespace_name  FORMAT a13  HEADING 'Tablespace'
COLUMN column_name                  FORMAT a25  HEADING 'Col. Name'
SET PAGES 58 LINES 130 FEEDBACK OFF VERIFY OFF
BREAK ON owner
START title132 'Expandeded Index Report'
SPOOL rep_out\&db\ind_exp.lis
SELECT
            a.owner,
            a.index_name,
            a.index_type,
            a.table_owner,
            a.table_name,
            a.table_type,
            DECODE
            (a.uniqueness, 'UNIQUE', 'U','NONUNIQUE','N') uniqueness,
```

```
            a.tablespace_name,
            b.column_name
FROM
            dba_indexes a, dba_ind_columns b
WHERE
            owner LIKE UPPER('%&owner%')
            AND a.owner=b.index_owner(+)
            AND a.index_name=b.index_name(+)
ORDER BY
            owner, index_type;
SPOOL OFF
```

<C>SQL Script to report on index browning (the script to generate and store index statistics must be run first)

```
rem
rem NAME: brown.sql
rem FUNCTION: Analyze indexes and produce stat report
rem FUNCTION: Including browning indicator
rem
rem HISTORY: MRA 6/15/97 Created
rem
COLUMN del_lf_rows_len  FORMAT 999,999,999      HEADING 'Deleted Bytes'
COLUMN lf_rows_len      FORMAT 999,999,999      HEADING 'Filled Bytes'
COLUMN browning         FORMAT 999.90     HEADING 'Percent|Browned'
COLUMN height           FORMAT 999999     HEADING 'Height'
COLUMN blocks           FORMAT 999999     HEADING 'Blocks'
COLUMN distinct_keys    FORMAT 999999999  HEADING '#|Keys'
COLUMN most_repeated_key FORMAT 999999999 HEADING 'Most|Repeated|Key'
COLUMN used_space       FORMAT 999999999  HEADING 'Used|Space'
COLUMN rows_per_key     FORMAT 999999     HEADING 'Rows|Per|Key'
ACCEPT owner PROMPT 'Enter table owner name: '
SET HEADING OFF FEEDBACK OFF VERIFY OFF ECHO OFF RECSEP OFF PAGES 0
TTITLE OFF
DEFINE cr='chr(10)'
SPOOL index_sz.sql
SELECT
  'CREATE TABLE stat_temp AS SELECT * FROM index_stats;'||&&cr||
  'TRUNCATE TABLE stat_temp;'
FROM dual;
SELECT
'ANALYZE INDEX '||owner||'.'||index_name||' VALIDATE STRUCTURE;'||&&cr||
      'INSERT INTO stat_temp SELECT * FROM index_stats;'||&&cr||
      'COMMIT;'
FROM
      dba_indexes
WHERE
      owner=UPPER('&owner');
SPOOL OFF
PROMPT ‘Analyzing Indexes’
SET FEEDBACK OFF TERMOUT OFF LINES 132 VERIFY OFF
START index_sz.sql
```

```
SET TERMOUT ON FEEDBACK ON VERIFY ON LINES 132 PAGES 58
START title132 "Index Statistics Report"
SPOOL rep_out/&db/browning.lst
SELECT
      name,
      del_lf_rows_len,
      lf_rows_len,
(del_lf_rows_len/DECODE((lf_rows_len+del_lf_rows_len),0,1,lf_rows_len+
del_lf_rows_len))*100 browning,
      height,
      blocks,
      distinct_keys,
      most_repeated_key,
      used_space,
      rows_per_key
FROM
      stat_temp
WHERE rows_per_key>0;
SPOOL OFF
SET FEEDBACK ON TERMOUT ON LINES 80 VERIFY ON
HOST del stat_temp
```

Figure 12.31 Script to produce index statistics reports from ANALYZE INDEX command

<C>SQL Script to report Index Statistics:

```
rem  NAME: IN_STAT.sql
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 a8          HEADING "Owner"
COLUMN index_name              FORMAT a25         HEADING "Index"
COLUMN status                  FORMAT a7          HEADING "Status"
COLUMN blevel                  FORMAT 9,999       HEADING "Tree|Level"
COLUMN leaf_blocks             FORMAT 999,999,999 HEADING "Leaf Blk"
COLUMN distinct_keys           FORMAT 999,999,999 HEADING "# Keys"
COLUMN avg_leaf_blocks_per_key FORMAT 9,999       HEADING "Avg.|LB/Key"
COLUMN avg_data_blocks_per_key FORMAT 9,999       HEADING "Avg.|DB/Key"
COLUMN clustering_factor       FORMAT 999,999     HEADING "Clstr|Factor"
COLUMN num_rows                FORMAT 999,999,999 HEADING "Number|Rows"
COLUMN sample_size             FORMAT 99,999      HEADING "Sample|Size"
COLUMN last_analyzed           HEADING 'Analysis|Date'
rem
BREAK ON owner
START title132 "Index Statistics Report"
SPOOL rep_out\&db\ind_stat
rem
SELECT
            owner, index_name, status, blevel, leaf_blocks,
            distinct_keys, avg_leaf_blocks_per_key,
            avg_data_blocks_per_key, clustering_factor,
```

```
            num_rows, sample_size, last_analyzed
FROM
            dba_indexes
WHERE
            owner LIKE UPPER('&&iowner')
            AND index_name LIKE UPPER('&&iname')
            AND num_rows>0
ORDER BY
            1,2;
rem
SPOOL OFF
SET PAGES 22 LINES 80 VERIFY ON FEEDBACK ON
CLEAR COLUMNS
UNDEF iowner
UNDEF iname
UNDEF owner
UNDEF name
TTITLE OFF
```

<C>SQL Script to monitor partitioned indexes:

```
rem
rem Name: ind_part.sql
rem Function : Report on partitioned index structure
rem History: MRA 6/14/97 Created
rem
COLUMN index_owner            FORMAT a10 HEADING 'Owner'
COLUMN index_name             FORMAT a15 HEADING 'Index'
COLUMN partition_name         FORMAT a15 HEADING 'Partition'
COLUMN tablespace_name        FORMAT a15 HEADING 'Tablespace'
COLUMN high_value             FORMAT a10 HEADING 'Partition|Value'
SET LINES 78
START title80 'Index Partition Files'
BREAK ON index_owner ON index_name
SPOOL rep_out/&&db/ind_part.lis
SELECT
      index_owner,
      index_name,
      partition_name,
      high_value,
      tablespace_name,
      logging
FROM sys.dba_ind_partitions
ORDER BY index_owner,index_name
/
SPOOL OFF
```

<C>SQL Script to report on partitioned index storage characteristics:

```
rem
rem NAME:   ind_pstor.sql
rem FUNCTION: Provide data on partitioned index storage charcacteristics
rem HISTORY: MRA 6/13/97 Created
rem
COLUMN owner                    FORMAT a6         HEADING 'Owner'
COLUMN index_name               FORMAT a14        HEADING 'Table'
COLUMN partition_name           FORMAT a9         HEADING 'Partition'
COLUMN tablespace_name          FORMAT a11        HEADING 'Tablespace'
COLUMN pct_free                 FORMAT 9999       HEADING '%|Free'
COLUMN ini_trans                FORMAT 9999       HEADING 'Init|Tran'
COLUMN max_trans                FORMAT 9999       HEADING 'Max|Tran'
COLUMN initial_extent           FORMAT 9999999    HEADING 'Init|Extent'
COLUMN next_extent              FORMAT 9999999    HEADING 'Next|Extent'
COLUMN max_extent                                 HEADING 'Max|Extents'
COLUMN pct_increase             FORMAT 999        HEADING '%|Inc'
COLUMN distinct_keys            FORMAT 9999999    HEADING '#Keys'
COLUMN clustering_factor        FORMAT 999999     HEADING 'Clus|Fact'
SET LINES 130
START title132 'Index Partition File Storage'
BREAK ON index_owner on index_name
SPOOL rep_out/&&db/ind_pstor.lis
SELECT
      index_owner,
      index_name,
      tablespace_name,
      partition_name,
      pct_free,
      ini_trans,
      max_trans,
      initial_extent,
      next_extent,
      max_extent,
      pct_increase,
      distinct_keys,
      clustering_factor
FROM sys.dba_ind_partitions
ORDER BY index_owner,index_name
/
SPOOL OFF
```

<C>SQL Script to document clusters:

```
rem
rem File:   CLU_REP.SQL
rem Purpose:            Document Cluster Data
rem Use:    From user with access to DBA_ views
rem
rem When    Who         What
rem ------- -----       -------
rem 5/27/93 Mike Ault   Initial Creation
rem 6/15/97 Mike Ault   Verified against Oracle8
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 SKIP 1 ON cluster ON tablespace
SPOOL rep_out\&db\cluster
SELECT
            a.owner,a.cluster_name,tablespace_name,
            table_name,tab_column_name,clu_column_name
FROM
            dba_clusters a,dba_clu_columns b
WHERE
            a.owner = b.owner and
            a.cluster_name=b.cluster_name
ORDER BY 1,2,3,4
/
SPOOL OFF
```

<C>SQL Script to document Cluster Sizing characteristics:

```
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 999          HEADING "%|Fre"
COLUMN pct_used             FORMAT 999          HEADING "%|Use"
COLUMN key_size             FORMAT 999999       HEADING "Key Size"
COLUMN ini_trans            FORMAT 999          HEADING "Ini|Trn"
COLUMN max_trans            FORMAT 999          HEADING "Max|Trn"
COLUMN initial_extent       FORMAT 999999999    HEADING "Init Ext"
COLUMN next_extent          FORMAT 999999999    HEADING "Next Ext"
COLUMN min_extents          FORMAT 999          HEADING “Min|Ext”
COLUMN max_extents          FORMAT 999          HEADING "Max|Ext"
COLUMN pct_increase         FORMAT 999          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
CLEAR COLUMNS
CLEAR BREAKS
SET PAGES 22 LINES 80 FEEDBACK ON
PAUSE Press enter to continue
```

<C>SQL Script to document cluster types:

```
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 report_output/&db/clu_type
SELECT
            owner,
            cluster_name,
            tablespace_name,
            avg_blocks_per_key,
            cluster_type,
            function,
            hashkeys
FROM
            dba_clusters
ORDER BY 2
GROUP BY  owner, tablespace, type
/
SPOOL OFF
SET PAGES 22 LINES 80 FEEDBACK ON
CLEAR COLUMNS
TTITLE OFF
```

<C>SQL Script to Document Snapshots:

```
rem
rem Name:   snap_rep.sql
rem Purpose:Report on database Snapshots
rem Use:    From an account that accesses DBA_ views
rem
rem   When       Who         What
rem   --------   ---------  ------------------
rem   5/27/93    Mike Ault  Initial Creation
rem
SET PAGES 56 LINES 130 FEEDBACK OFF VERIFY OFF
rem
COLUMN snapshot         FORMAT a30  HEADING "Snapshot"
COLUMN source           FORMAT a30  HEADING "Source Table"
COLUMN link             FORMAT a20  HEADING "Link"
COLUMN log                          HEADING "Use|Log?"
COLUMN refreshed                    HEADING "Refreshed?"
COLUMN type             FORMAT a10  HEADING "Ref|Type"
COLUMN refreshed                    HEADING "Last Refresh"
COLUMN start            FORMAT a13  HEADING "Start Refresh"
COLUMN error                        HEADING "Error"
COLUMN next             FORMAT a13  HEADING "Next Refresh"
rem
PROMPT Percent signs are wild card
ACCEPT snap_owner PROMPT Enter the snapshot owner
START title132 "Snapshot Report for &snap_owner"
SPOOL snap_rep&db
rem
SELECT
            name||'.'||table_name Snapshot, master_view,
            master_owner||'.'||master Source,
            master_link Link,
            can_use_log Log, last_refresh Refreshed,
            start_with start,
            DECODE(type,'FAST','F','COMPLETE','C'),
            next,
            start_with Started, query
FROM dba_snapshots
WHERE owner LIKE UPPER('%&snap_owner%')
ORDER BY owner,3,5;
rem
SPOOL OFF
```

<C>SQL Script to document Snapshot Logs:

```
rem
rem Name:   snap_log_rep.sql
rem Purpose:            Report on database Snapshot Logs
rem Use:    From an account that accesses DBA_ views
rem
rem   When       Who         What
rem   --------   ---------  ------------------
rem   5/27/93    Mike Ault  Initial Creation
rem
SET PAGES 56 LINES 130 FEEDBACK OFF
START title132 "Snapshot Log Report"
```

```
SPOOL snap_log_rep&db
rem
COLUMN log_owner        FORMAT a10 HEADING "Owner"
COLUMN master           FORMAT a20 HEADING "Master"
COLUMN log_table        FORMAT a20 HEADING "Snapshot"
COLUMN trigger          FORMAT a20 HEADING "Trigger Text"
COLUMN current          HEADING "Last Refresh"
rem
SELECT
            log_owner, master, log_table table,
            log_trigger trigger, rowids, filter_columns filtered,
             current_snapshots current, snapshot_id id
FROM
            dba_snapshot_logs
ORDER BY 1;
rem
SPOOL OFF
```

<C>SQL Script to document all types:

```
rem
rem NAME: types.sql
rem FUNCTION: Provide basic report of all database types
rem HISTORY : MRA 6/15/97 Created
rem
COLUMN owner      FORMAT a10        HEADING 'Type|Owner'
COLUMN type_name  FORMAT a30        HEADING 'Type|Name'
COLUMN typecode   FORMAT a27        HEADING 'Type|Code'
COLUMN predefined FORMAT a3         HEADING Pre?
COLUMN incomplete FORMAT a3         HEADING Inc?
COLUMN methods    FORMAT 9999999    HEADING '#|Methods'
COLUMN attributes FORMAT 999999     HEADING '#|Attrib'
SET LINES 130 PAGES 58 VERIFY OFF FEEDBACK OFF
BREAK ON owner
START title132 'Database Types Report'
SPOOL rep_out\&db\types.lis
SELECT
      DECODE(owner, null,'SYS-GEN',owner) owner,
      type_name,
      typecode,
      attributes,
      methods,
      predefined,
      incomplete
FROM dba_types
ORDER BY owner, type_name;
SPOOL OFF
```

<C>SQL Script to document collection types:

```
rem
```

```
rem NAME: col_type.sql
rem FUNCTION: Document the collection types in the database
rem HISTORY: MRA 6/15/97 Created
rem
COLUMN owner            FORMAT a10  HEADING 'Collec.|Owner'
COLUMN type_name        FORMAT a16  HEADING 'Type|Name'
COLUMN coll_type        FORMAT a15  HEADING 'Collec.|Type'
COLUMN upper_bound                  HEADING 'VARRAY|Limit'
COLUMN elem_type_owner  FORMAT a10  HEADING 'Elementary|Type|Owner'
COLUMN elem_type_name   FORMAT a11  HEADING 'Elementary|Type|Name'
SET PAGES 58 LINES 78 VERIFY OFF FEEDBACK OFF
START title80 'Collection Type Report'
SPOOL rep_out\&db\col_type.lis
select
      owner,
      type_name,
      coll_type,
      upper_bound,
      elem_type_owner,
      elem_type_name
FROM dba_coll_types
WHERE owner LIKE '%&owner%'
/
SPOOL OFF
```

<C>SQL Script to document type methods:

```
rem
rem NAME typ_meth.sql
rem FUNCTION : Create a report of type methods
rem HISTORY: MRA 6/16/97 Created
rem
COLUMN owner            FORMAT a10        HEADING 'Owner'
COLUMN type_name        FORMAT a13        HEADING 'Type|Name'
COLUMN method_name      FORMAT a17        HEADING 'Method|Name'
COLUMN method_type                        HEADING 'Method|Type'
COLUMN parameters       FORMAT 99999      HEADING '#|Param'
COLUMN results          FORMAT 99999      HEADING '#|Results'
COLUMN method_no        FORMAT 999999     HEADING 'Meth.|Number'
BREAK ON owner ON type_name
SET LINES 80 PAGES 58 VERIFY OFF FEEDBACK OFF
START title80 'Type Methods Report'
SPOOL rep_out\&db\typ_meth.lis
SELECT
      owner,
      type_name,
      method_name,
      method_no,
      method_type,
      parameters,
      results
FROM dba_type_methods
ORDER BY owner, type_name;
SPOOL OFF
```

<C>SQL Script to generate a report on REFs in the database

```
rem
rem NAME: tab_ref.sql
rem FUNCTION: Generate a lit of all REF columns in the database
rem HISTORY: MRA 6/16/97 Created
rem
COLUMN owner              FORMAT a8  HEADING 'Owner'
COLUMN table_name         FORMAT a23 HEADING 'Table|Name'
COLUMN column_name        FORMAT a15 HEADING 'Column|Name'
COLUMN with_rowid         FORMAT a5  HEADING 'With|Rowid'
COLUMN is_scoped          FORMAT a6  HEADING 'Scoped'
COLUMN scope_table_owner  FORMAT a8  HEADING 'Scope|Table|Owner'
COLUMN scope_table_name   FORMAT a15 HEADING 'Scope|Table|Name'
BREAK ON owner
SET PAGES 58 LINES 130 FEEDBACK OFF VERIFY OFF
START title132 'Database REF Report'
SPOOL rep_out\&db\tab_ref.lis
SELECT
      owner,
      table_name,
      column_name,
      with_rowid,
      is_scoped,
      scope_table_owner,
      scope_table_name
FROM
      dba_refs
ORDER BY
      owner;
SPOOL OFF
```

<C>SQL Script to document Database Users:

```
REM
REM NAME        : DB_USER.SQL
REM
REM FUNCTION    : GENERATE USER_REPORT
REM Limitations : None
REM
REM Updates     : MRA 6/10/97 added ORACLE8 account status
REM
SET PAGESIZE 58  LINESIZE 131 FEEDBACK OFF
rem
COLUMN username                 FORMAT a10 HEADING User
COLUMN account_status           FORMAT a10 HEADING Status
COLUMN default_tablespace       FORMAT a15 HEADING Default
COLUMN temporary_tablespace     FORMAT a15 HEADING Temporary
COLUMN granted_role             FORMAT a21 HEADING Roles
```

```
COLUMN default_role             FORMAT a9  HEADING Default?
COLUMN admin_option             FORMAT a7  HEADING Admin?
COLUMN profile                  FORMAT a15 HEADING Profile
rem
START title132 'ORACLE USER REPORT'
DEFINE output = rep_out\&db\db_user
BREAK ON username SKIP 1 ON account_status ON default_tablespace
ON temporary_tablespace ON profile
SPOOL &output
rem
SELECT username,
       account_status,
       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
SET TERMOUT ON FLUSH ON FEEDBACK ON VERIFY ON
CLEAR COLUMNS
CLEAR BREAKS
PAUSE Press enter to continue
```

<C>SQL Script to document User/Role System Grants:

```
REM
REM NAME    : sys_role.SQL
REM PURPOSE: GENERATE SYSTEM GRANTS and ROLES REPORT
REM USE     : CALLED BY SQLPLUS
REM Limitations         : None
REM Revisions:
REM Date           Modified By      Reason For change
REM 08-Apr-1993    MIKE AULT        INITIAL CREATE
REM 10-Jun-1997    Mike Ault        Update to ORACLE8
REM
SET FLUSH OFF TERM OFF PAGESIZE 58  LINESIZE 78
COLUMN grantee          HEADING 'User or Role'
COLUMN admin_option     HEADING Admin?
START title80 'SYSTEM GRANTS AND ROLES REPORT'
DEFINE output = rep_out\&&db\role_report
SPOOL &output
SELECT
            grantee,
            privilege,
            admin_option
```

```
            FROM
                        sys.dba_sys_privs
            GROUP BY
                        grantee;
            SPOOL OFF
```

<C>SQL Script to generate Profiles report

```
REM NAME    : PROFILE_REPORT.SQL
REM PURPOSE : GENERATE USER PROFILES REPORT
REM Revisions:
REM Date                Modified By Reason For change
REM 08-Apr-1993         MIKE AULT   INITIAL CREATE
SET FLUSH OFF TERM OFF PAGESIZE 58 LINESIZE 78
COLUMN profile          HEADING Profile
COLUMN resource_name    HEADING 'Resource:'
COLUMN limit            HEADING Limit
START title80 'ORACLE PROFILES REPORT'
DEFINE output = rep_out/&&db/prof_rep
SPOOL &output
SELECT
            profile,
            resource_name,
            limit
FROM
            sys.dba_profiles
GROUP BY
            profile;
SPOOL OFF
```

<C>SQL Script to generate table grants report:

```
rem  PURPOSE: Produce report of table grants showing
rem               GRANTOR, GRANTEE and
rem               specific GRANTS.
rem  LIMITATIONS: User must have access to DBA_TAB_PRIVS
rem  INPUTS: Owner name
rem  OUTPUTS: Report of table grants
rem
rem  HISTORY:
rem Who:          What:             Date:
rem Mike Ault     Initial Creation  3/2/95
rem Mike Ault     Oracle 8 verified 6/10/97
rem
rem NOTES: Will not report grants to SYS or SYSTEM
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"
SPOOL rep_out\&db\grants
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
SET LINESIZE 80 PAGES 22 VERIFY ON FEEDBACK ON
CLEAR BREAKS
CLEAR COLUMNS
TTITLE OFF
```

<C>SQL Script for documenting column level grants:

```
REM FUNCTION:   SCRIPT FOR CAPTURING TABLE COLUMN GRANTS
REM
REM
REM This script is intended to run with Oracle7 or Oracle8.
REM
REM Running this script will create a script of all the grants
REM on columns
REM
REM Grants must be made by the origianl grantor so the script
REM connects as that user using the username as the password
REM edit the proper password in at time of running
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
DEFINE cr=chr(10);
BREAK ON line1
COLUMN dbname NEW_VALUE db NOPRINT
SELECT name dbname FROM v$database;
SPOOL rep_out\&db\grt_cols.sql
rem
SELECT
  'CONNECT '||grantor||'/'||grantor line1,'GRANT '||&&cr||
  lower(privilege)||'('||column_name||') ON
 '||owner||'.'||table_name||&&cr||
 ' TO '|| lower(grantee) ||&&cr||
 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
SET VERIFY ON FEEDBACK ON TERMOUT ON PAGESIZE 22 EMBEDDED OFF
CLEAR COLUMNS
CLEAR COMPUTES
CLEAR BREAKS
```

<C>SQL Script to generate a list of sids, pids, osusers, terminals, etc. for current processes:

```
rem
rem  Name:     pid.sql
rem
rem  FUNCTION: Generate a list of current oracle sids/pids
rem
COLUMN program    FORMAT a25
COLUMN pid        FORMAT 9999
COLUMN sid        FORMAT 9999
COLUMN osuser                 HEADING Oper|System|User
SET LINES 132 PAGES 58
BREAK ON username
COMPUTE COUNT OF pid ON username
START title132 "Oracle Processes"
SPOOL rep_out\&db\cur_proc
SELECT
      NVL(a.username,'Null') username,
      b.pid,
      a.sid,
      DECODE(a.terminal,'?','Detached',a.terminal) terminal,
      b.program,
      b.spid,
      a.osuser,
```

```
      a.serial# ser#
 FROM
      v$session a,
      v$process b
WHERE
      a.sid=b.pid
ORDER BY
      a.username,
      b.pid
/
SPOOL OFF
CLEAR BREAKS
CLEAR COLUMNS
SET PAGES 22 LINES 80
TTITLE OFF
PAUSE Press enter to continue
```

<C>SQL Script to generate report of users login times (bonus script):

```
rem
rem login.sql
rem FUNCTION: Generate report of session login times
rem
rem MRA/Revealnet
rem
COLUMN sids FORMAT a10 HEADING "Sid,Ser#"
COLUMN username FORMAT a15 HEADING Username
COLUMN ltime FORMAT a20 HEADING "Login Time"
COLUMN program FORMAT a30 HEADING Program
START title80 'User Login Times'
SPOOL rep_out\&db\login_tm
SELECT s.sid||','||n.serial# sids,
       n.username,
       n.status "Status",
       n.program,
       to_char(sysdate-(hsecs-s.value)/(24*3600*100),
          'MM/DD/YYYY HH24:MI:SS') ltime
FROM sys.V_$sesstat s,
     sys.V_$session n,
     sys.v_$timer
WHERE s.statistic# = 13
AND   s.sid = n.sid
AND   s.value != 0
ORDER BY 2,5;
SPOOL OFF
```

<C>SQL Script which uses free_space view (above) to monitor free space in tablespaces:

```
rem
rem Name:     free_spc2.sql
rem
rem FUNCTION: Provide data on tablespace extent status
rem FUNCTION: this report uses the free_space2 view
rem FUNCTION: includes fsfi from DBA Handbook
```

```
rem
SET FEED OFF
SET FLUSH OFF
SET VERIFY OFF
set pages 58 LINES 132
COLUMN tablespace      HEADING Name             FORMAT a30
COLUMN files           HEADING '#Files'         FORMAT 9,999
COLUMN pieces          HEADING Frag             FORMAT 9,999
COLUMN free_bytes      HEADING 'Free|Byte'      FORMAT 9,999,999,999
COLUMN free_blocks     HEADING 'Free|Blk'       FORMAT 999,999
COLUMN largest_bytes   HEADING 'Biggest|Bytes'  FORMAT 9,999,999,999
COLUMN largest_blks    HEADING 'Biggest|Blks'   FORMAT 999,999
COLUMN ratio           HEADING 'Percent'        FORMAT 999.999
COLUMN average_fsfi    HEADING 'Average|FSFI'   FORMAT 999.999
START title132 "FREE SPACE REPORT"
DEFINE 1 = report_output/&&db/free_spc
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,
       SUM(fsfi)/COUNT(*) average_fsfi
FROM
       free_space
GROUP BY
       tablespace;
SPOOL OFF
CLEAR COLUMNS
TTITLE OFF
SET FEED ON
SET FLUSH ON
SET VERIFY ON
SET PAGES 22 LINES 80
PAUSE Press Enter to Continue
```

<C>SQL to create a new view to monitor auto extend features in pre 7.3 Oracle:

```
CREATE VIEW dba_file_data AS
SELECT
a.name tablespace,a.dflminext min_extents, a.dflmaxext max_extents,
a.dflinit init,a.dflincr next,a.dflextpct pct_increase, d.name datafile,
b.blocks datafile_size, c.maxextend max_extend, c.inc ext_incr
FROM ts$ a, file$ b, filext$ c, v$dbfile d
WHERE
a.ts#=b.ts# and b.file#=c.file# and b.file#=d.file#
/
```

<C>SQL Script to document all database datafiles:

```
REM
REM     Name:      datafile.sql
REM     FUNCTION:  Document  file sizes and locations
REM     Use:       From SQLPLUS
REM
CLEAR COMPUTES
COLUMN file_name                FORMAT A50
COLUMN tablespace_name          FORMAT A15
COLUMN meg                      FORMAT 99,999.90
START title80 'DATABASE DATAFILES'
SPOOL rep_out\&db\datafile
BREAK ON tablespace_name SKIP 1 ON REPORT
COMPUTE SUM OF meg ON tablespace_name
COMPUTE SUM OF meg ON REPORT
SELECT
      tablespace_name,
      file_name,
      bytes/1048576 meg
FROM
      dba_data_files
ORDER BY
      tablespace_name
/
SPOOL OFF
CLEAR COLUMNS
CLEAR COMPUTES
PAUSE Press enter to continue
```

<C>SQL Script to map extent usage for a tablespace:

```
rem
rem Name: mapper.sql
rem Function: create an extent map for a specific tablespace
rem  Based on a technique from “DBA Handbook”
rem Mike Ault 7/19/96 Trecom/RevealNet
rem
SET PAGES 47 LINES 132 VERIFY OFF FEEDBACK OFF
COLUMN file_id    HEADING 'File|id'
COLUMN value      NEW_VALUE dbblksiz NOPRINT
COLUMN meg        FORMAT 9,999.99
SELECT value FROM v$parameter WHERE name='db_block_size';
START title132 '&&ts Mapping Report'
SPOOL rep_out/&db/ts_map
SELECT
      'free space' owner, '      ' object,
      file_id, block_id, blocks,
      (blocks*&dbblksiz)/(1024*1024) meg
FROM
      dba_free_space
WHERE
      tablespace_name=UPPER('&&ts')
UNION
SELECT
      SUBSTR(owner,1,20), SUBSTR(segment_name, 1,32),
```

```
      file_id, block_id, blocks,
      (blocks*&dbblksiz)/(1024*1024) meg
FROM
      dba_extents
WHERE
      tablespace_name = UPPER('&&ts')
ORDER BY 3,4;
SPOOL OFF
UNDEF ts
SET PAGES 22 LINES 80 VERIFY ON FEEDBACK ON
CLEAR COLUMNS
TTITLE OFF
```

<C>SQL Script to monitor sequences:

```
rem  NAME: Sequence.sql
rem
rem  HISTORY:
rem  Date               Who                     What
rem  --------           -------------------     -------------------
rem  5/10/93            Mike Ault                Creation
rem  FUNCTION: Generate report on Sequences
rem  INPUTS:
rem
rem         1 - Sequence Owner or Wild Card
rem         2 - Sequence Name or Wild Card
rem
rem  ************************************************************
SET HEADING OFF VERIFY OFF PAUSE OFF
PROMPT ** Sequence Report **
PROMPT
PROMPT Percent signs are wild
ACCEPT sequence_owner char  'Enter account to report on (or pct sign):';
ACCEPT sequence_name char  'Enter sequence to report on (or pct sign):';
PROMPT
PROMPT Report file name is SEQUENCE.LIS
SET HEADING ON
SET LINESIZE 130 PAGESIZE 56 NEWPAGE 0 TAB OFF SPACE 1
SET TERMOUT OFF
BREAK ON sequence_owner SKIP 2
COLUMN sequence_owner   FORMAT A30  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'
START title132 "SEQUENCE REPORT"
SPOOL report_output/&&db/sequence
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
```

<C>SQL Script to monitor synonyms:

```
REM
REM NAME    : SYNONYM.SQL
REM PURPOSE : GENERATE REPORT OF A USERS SYNONYMS
REM USE     : FROM SQLPLUS
REM Limitations         : None
REM Revisions:
REM Date                Modified By Reason For change
REM 12/MAY/93           Mike Ault   Initial Creation
REM 15/Jun/97           Mike Ault   Verifed for Oracle8
REM
PROMPT Percent signs are Wild Cards
PROMPT
ACCEPT own PROMPT 'Enter the user who owns synonym: '
SET PAGES 56 LINES 130 VERIFY OFF FEEDBACK OFF TERM OFF
START title132 "Synonym Report"
SPOOL rep_out/&&db/synonym
COLUMN host             FORMAT a24 HEADING "Connect String"
COLUMN owner            FORMAT a15
COLUMN table            FORMAT a35
COLUMN db_link          FORMAT a6 HEADING Link
COLUMN username         FORMAT a15
SELECT
            a.owner,
            synonym_name ,
            table_owner ||'.'|| table_name "Table" ,
            b.db_link,
            username,
            host
FROM
            dba_synonyms a,
            dba_db_links b
WHERE
            a.db_link = b.db_link(+) AND
            a.owner LIKE UPPER('&own');
SPOOL OFF
```

<C>SQL Script to monitor Database Links:

```
REM
REM NAME         : DBLINK.SQL
REM FUNCTION     : GENERATE REPORT OF DATABASE LINKS
REM USE          : FROM SQLPLUS
REM Limitations  : None
REM
SET PAGES 58 LINES 130 VERIFY OFF TERM OFF
START title132 "Db Links Report"
SPOOL report_output/&db/dblinks
COLUMN host             FORMAT a60      HEADING "Connect|String"
COLUMN owner            FORMAT a15      HEADING "Creator"
COLUMN db_link          FORMAT a10      HEADING " DB Link|Name"
COLUMN username         FORMAT a15      HEADING "Connecting|User"
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
```

<C>SQL to create rollback segment monitoring views:

```
REM
REM FUNCTION: create views required for rbk1 and rbk2 reports.
REM
rem
CREATE OR REPLACE VIEW rollback1 AS
SELECT
            d.segment_name,
            extents,
            optsize,
            shrinks,
            aveshrink,
            aveactive,
            d.status
FROM
            v$rollname n,
            v$rollstat s,
            dba_rollback_segs d
WHERE
            d.segment_id=n.usn(+)
            AND d.segment_id=s.usn(+)
;
```

```
CREATE OR REPLACE VIEW rollback2 AS
SELECT
            d.segment_name,
            extents,
            xacts,
            hwmsize,
            rssize,
            waits,
            wraps,
            extends,
            d.status
FROM
            v$rollname n,
            v$rollstat s,
            dba_rollback_segs d
WHERE
            d.segment_id=n.usn(+)
            AND d.segment_id=s.usn(+);
```

<C>SQL Scripts to monitor rolback segments that use rollback views created above:

```
REM
REM NAME                : RBK1.SQL
REM FUNCTION            : REPORT ON ROLLBACK SEGMENT STORAGE
REM FUNCTION            : USES THE ROLLBACK1 VIEW
REM USE                 : FROM SQLPLUS
REM Limitations         : None
REM
COLUMN tablespace_name  FORMAT a10  HEADING 'TABLESPACE'
COLUMN segment_name     FORMAT A10  HEADING 'ROLLBACK'
COLUMN extents          FORMAT 9,999            HEADING 'CUR EXTENTS'
COLUMN optsize          FORMAT 99,999,999       HEADING 'OPTL SIZE'
COLUMN shrinks          FORMAT 9,999            HEADING 'SHRINKS'
COLUMN aveshrink        FORMAT 99,999,999       HEADING 'AVE SHRINK'
COLUMN aveactive        FORMAT 99,999,999       HEADING 'AVE TRANS'
COLUMN status           FORMAT A8               HEADING 'STATUS'
rem
SET FEEDBACK OFF VERIFY OFF LINES 80 PAGES 58
@title80 "ROLLBACK SEGMENT STORAGE"
SPOOL rep_out\&db\rollbck1
rem
SELECT * FROM rollback1 ORDER BY segment_NAME;
SPOOL OFF
PAUSE Press enter to continue
CLEAR COLUMNS
TTITLE OFF
SET FEEDBACK ON VERIFY ON LINES 80 PAGES 22

REM
REM NAME        : RBK2.SQL
REM FUNCTION    : REPORT ON ROLLBACK SEGMENT STATISTICS
REM FUNCTION    : USES THE ROLLBACK2 VIEW
REM USE         : FROM SQLPLUS
REM Limitations : None
```

```
REM
COLUMN SEGMENT_NAME     FORMAT A10    HEADING 'ROLLBACK'
COLUMN EXTENTS  FORMAT 9,999          HEADING 'EXTENTS'
COLUMN XACTS    FORMAT 9,999          HEADING 'TRANS'
COLUMN HWMSIZE  FORMAT 99,999,999     HEADING 'LARGEST|TRANS'
COLUMN RSSIZE   FORMAT 99,999,999     HEADING 'CUR SIZE'
COLUMN WAITS    FORMAT 9,999          HEADING 'WAITS'
COLUMN WRAPS    FORMAT 9,999          HEADING 'WRAPS'
COLUMN EXTENDS  FORMAT 9,999          HEADING 'EXTENDS'
COLUMN STATUS   FORMAT A7             HEADING 'STATUS'
rem
SET FEEDBACK OFF VERIFY OFF lines 80 pages 58
rem
@title80 "ROLLBACK SEGMENT STATISTICS"
SPOOL rep_out\&db\rollbck2
rem
SELECT * FROM rollback2 ORDER BY segment_name;
SPOOL OFF
SET LINES 80 PAGES 20 FEEDBACK ON VERIFY ON
TTITLE OFF
CLEAR COLUMNS
PAUSE Press enter to continue
```

<C>SQL Script to tell what users are using what rollback segments:

```
rem   Name  : TX_RBS.SQL
rem   Purpose: Generate a report of active rollbacks
rem   Use   : From SQL*Plus
rem   History:
rem   Date              Who             What
rem   Sept 91           Lan Nguyen      Presented in paper at IOUG
rem                     Walter Lindsey
rem   5/15/93           Mike Ault       Added Title80, sets and output
rem   1/4/97            Mike Ault       Verified against 7.3
rem*********************************************************************
COLUMN  name FORMAT a21             HEADING "Rollback Segment Name"
COLUMN  pid  FORMAT 9999999999      HEADING "Oracle PID"
COLUMN  spid FORMAT 9999999999      HEADING "Sys PID"
SET PAGES 56  LINES 130 VERIFY OFF FEEDBACK OFF
START title132 "Rollback Segments in Use"
SPOOL report_output/&db/tx_rbs
SELECT
            r.name, l.Sid, p.spid,
            NVL(p.username, 'no transaction') "Transaction",
            p.terminal "Terminal"
FROM
            v$lock l,
            v$process p,
            v$rollname r
WHERE
            l.Sid = p.pid (+)
            and TRUNC(l.id1(+) / 65536) = r.usn
            and l.type(+) = 'TX'
            and l.lmode(+) = 6
ORDER BY r.name;
```

```
SPOOL OFF
SET PAGES 22  LINES 80 VERIFY ON FEEDBACK ON
CLEAR COLUMNS
TTITLE OFF
```

<C>SQL Script to monitor rollback usage for a single transaction:

```
rem*********************************************************
rem  Name   : UNDO.SQL
rem  Purpose: Document rollback usage for a single
rem           transaction
rem  Use    : Note: You must alter the UNDO script and add a
rem           call to the transaction at the indicated line
rem  Restrictions:     : The database should be placed in DBA mode and
rem           these be the only transaction running.
rem  History:
rem   Date              Who            What
rem   Sept 91           Lan Nguyen     Presented in paper at IOUG
rem                     Walter Lindsey
rem   5/15/93           Mike Ault   Changed to use one table
rem
SET FEEDBACK OFF  TERMOUT OFF
COLUMN name FORMAT a40
DEFINE undo_overhead=54
DROP TABLE undo_data;
CREATE TABLE undo_data
            (
            tran_no number, start_writes number, end_writes number
            );
INSERT INTO undo_data
SELECT 1, SUM(writes),0 from v$rollstat;
SET FEEDBACK ON  TERMOUT ON
rem
rem   INSERT TRANSACTION HERE
rem
SET FEEDBACK OFF  TERMOUT OFF
UPDATE undo_data SET end_writes = SUM(writes) FROM v$rollstat;
 WHERE tran_no=1;
SET FEEDBACK ON  TERMOUT ON
SELECT  ((end-writes - start_writes) - &undo_overhead)
"Number of Rollback Bytes Generated"
FROM undo_data;
SET TERMOUT OFF FEEDBACK OFF
DROP TABLE undo_data;
```

<C>SQL Fragment to show deffered rollback usage:

```
SELECT segment_name, segment_type, tablespace_name
FROM sys.dba_segments
WHERE segment_type = 'DEFERRED ROLLBACK';
```

<C>SQL fragment to determine if a rollback segment under ORACLE7 has outstanding transactions.

```
      SELECT name, xacts 'ACTIVE TRANSACTIONS'
      FROM     v$rollname, v$rollstat
      WHERE status = 'PENDING OFFLINE'
            AND v$rollname.usn = v$rollstat.usn;
```

<C>SQL Script to monitor redo log activity

```
rem
rem Name:     log_stat.sql
rem
rem FUNCTION: Provide a current status for redo logs
rem
rem
COLUMN first_change# FORMAT 99999999  HEADING Change#
COLUMN group#        FORMAT 9,999     HEADING Grp#
COLUMN thread#       FORMAT 999       HEADING Th#
COLUMN sequence#     FORMAT 999,999   HEADING Seq#
COLUMN members       FORMAT 999       HEADING Mem
COLUMN archived      FORMAT a4        HEADING Arc?
COLUMN first_time    FORMAT a21       HEADING 'Switch Time'
BREAK ON thread#
SET PAGES 60 LINES 131 FEEDBACK OFF
START title132 'Current Redo Log Status'
SPOOL rep_out\&db\log_stat
SELECT thread#,
       group#,
       sequence#,
       bytes,
       members,
       archived,
       status,
       first_change#,
       TO_CHAR(first_time, 'DD-MM-YYYY HH24:MI:SS') first_time
  FROM
       sys.v_$log
  ORDER BY
       thread#,
       group#;
SPOOL OFF
PAUSE Press Enter to continue
SET PAGES 22 LINES 80 FEEDBACK ON
CLEAR BREAKS
CLEAR COLUMNS
TTILE OFF
```

<C> SQL Script to show redo log history:

```
REM
REM NAME   :log_hist.sql
REM PURPOSE:Provide info on logs for last 24 hour since last log switch
REM USE    : From SQLPLUS
REM Limitations   : None
```

```
REM
COLUMN thread#          FORMAT 999        HEADING 'Thrd#'
COLUMN sequence#        FORMAT 99999      HEADING 'Seq#'
COLUMN first_change#                      HEADING 'SCN Low#'
COLUMN next_change#                       HEADING 'SCN High#'
COLUMN archive_name     FORMAT a50        HEADING 'Log File'
COLUMN first_time       FORMAT a20        HEADING 'Switch Time'
COLUMN name             FORMAT a30        HEADING 'Archive Log'
SET LINES 132
@title132 "Log History Report"
SPOOL rep_out\&db\log_hist
REM
SELECT
      a.recid,
      a.thread#,
      a.sequence#,
      a.first_change#,
      a.next_change#,
      TO_CHAR(a.first_time,'DD-MON-YYYY HH24:MI:SS') first_time,
      x.name
FROM
 v$log_history a, v$archived_log x
WHERE
  a.first_time>
   (SELECT b.first_time-1
   FROM v$log_history b WHERE b.next_change# =
    (SELECT MAX(c.next_change#) FROM v$log_history c)) AND
      a.recid=x.sequence#(+);
SPOOL OFF
SET LINES 80
CLEAR COLUMNS
TTITLE OFF
PAUSE Press enter to continue
```

<C>SQL Script to monitor redo log statistics:

```
REM
REM NAME                : rdo_stat.sql
REM PURPOSE             : Show REDO latch statisitics
REM USE                 : from SQLPlus
REM Limitations         : Must have access to v$_ views
REM
SET PAGES 56 LINES 78 VERIFY OFF FEEDBACK OFF
START title80 "Redo Latch Statistics"
SPOOL rep_out/&&db/rdo_stat
rem
COLUMN name             FORMAT a30              HEADING Name
COLUMN percent          FORMAT 999.999          HEADING Percent
COLUMN total                                    HEADING Total
rem
SELECT
            l2.name,
            immediate_gets+gets Total,
            immediate_gets "Immediates",
            misses+immediate_misses "Total Misses",
            DECODE (100.*(GREATEST(misses+immediate_misses,1)/
```

```
            GREATEST(immediate_gets+gets,1)),100,0) Percent
FROM
            v$latch l1,
            v$latchname l2
WHERE
            l2.name like '%redo%'
            and l1.latch#=l2.latch# ;
rem
PAUSE Press ENTER to continue
rem
rem Name: Redo_stat.sql
rem
rem Function: Select redo statistics from v$sysstat
rem History:
rem Who           What              Date
rem ---------     ---------------   ----------------
rem Mike Ault     Revised from V6   1/04/97
rem Mike Ault     Verified Oracle8  6/15/97
rem
COLUMN name       FORMAT a30              HEADING 'Redo|Statistic|Name'
COLUMN value      FORMAT 999,999,999      HEADING 'Redo|Statistic|Value'
SET PAGES 80 LINES 60 FEEDBACK OFF VERIFY OFF
START title80 'Redo Log Statistics'
SPOOL rep_out/&&db/redo_stat
SELECT
      name,
      value
FROM
      v$sysstat
WHERE
      name LIKE '%redo%'
ORDER BY statistic#;
SPOOL OFF
SET LINES 24 FEEDBACK ON VERIFY ON
```

<C>SQL Script to document directories for database:

```
rem NAME: dir_rep.sql
rem FUNCTION: Report on Directories known by the database
rem HISTORY: MRA 6/16/97 Created
rem
COLUMN owner            FORMAT a10 HEADING 'Owner'
COLUMN directory_name   FORMAT a10 HEADING 'Directory'
COLUMN directory_path   FORMAT a40 HEADING 'Full Path'
SET VERIFY OFF PAGES 58 LINES 78 FEEDBACK OFF
START title80 'Database Directories Report'
SPOOL rep_out\&db\dir_rep.lis
SELECT
      owner,
      directory_name,
      directory_path
FROM
      dba_directories
ORDER BY
      owner;
```

```
SPOOL OFF
```

<C>SQL Script to document external libraries used by the database:

```
rem
rem NAME: lib_rep.sql
rem FUNCTION: Document External Library Entries in Database
rem HISTORY: MRA 6/16/97 Created
rem
COLUMN owner            FORMAT a8   HEADING 'Library|Owner'
COLUMN library_name     FORMAT a15  HEADING 'Library|Name'
COLUMN file_spec        FORMAT a30  HEADING 'File|Specification'
COLUMN dynamic          FORMAT a7   HEADING 'Dynamic'
COLUMN stauts           FORMAT a10  HEADING 'Status'
BREAK ON owner
SET FEEDBACK OFF VERIFY OFF LINES 78 PAGES 58
START title80 'Database External Libraries Report'
SPOOL rep_out\&db\lib_rep.lis
SELECT
      owner,
      library_name,
      file_spec,
      dynamic,
      status
FROM
      dba_libraries
ORDER BY
      owner;
SPOOL OFF
```

<C>SQL Script to document control file locations and status

```
rem
rem NAME : con_file.sql
rem FUNCTION: Document control file location and status
rem HISTORY: MRA 6/16/97 Creation
rem
COLUMN name   FORMAT a60 HEADING 'Control|File|Location' WORD_WRAPPED
COLUMN status FORMAT a7  HEADING 'Control|File|Status'
SET LINES 78 FEEDBACK OFF VERIFY OFF
START title80 'Control File Status'
SPOOL rep_out\&db\con_file.lis
SELECT
      name,
      status
FROM
      v$controlfile;
SPOOL OFF
```

<C>SQL Script to monitor control file records statistics

```
rem
rem NAME: con_rec.sql
rem FUNCTION: Provide documentation otf control file record stats
rem HISTORY: MRA 6/16/97 Creation
rem
COLUMN type             FORMAT a17        HEADING 'Record Type'
COLUMN record_size      FORMAT 999999     HEADING 'Record|Size'
COLUMN records_used     FORMAT 999999     HEADING 'Records|Used'
COLUMN first_index      FORMAT 9999999    HEADING 'First|Index'
COLUMN last_index       FORMAT 9999999    HEADING 'Last|Index'
COLUMN last_recid       FORMAT 999999     HEADING 'Last|Record|ID'
SET LINES 80 PAGES 58 FEEDBACK OFF VERIFY OFF
START title80 'Control File Records'
SPOOL rep_out\&db\con_rec.lis
SELECT
      type,
      record_size,
      records_total,
      records_used,
      first_index,
      last_index,
      last_recid
FROM
      v$controlfile_record_section;
SPOOL OFF
```

<C>SQL Script to document initialization parameters

```
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 a80 WORD_WRAPPED
COLUMN dbname NEW_VALUE db NOPRINT
SELECT name dbname FROM v$database;
DEFINE OUTPUT = 'rep_out\&db\init.ora'
SPOOL &OUTPUT
SELECT '# Init.ora file FROM v$parameter' name FROM dual
UNION
SELECT '# generated on:'||sysdate name FROM dual
UNION
SELECT '# script by MRA 11/7/95 REVEALNET' name FROM dual
UNION
SELECT '#' name FROM dual
UNION
SELECT name||' = '||value name  FROM v$parameter
WHERE value IS NOT NULL;
```

```
SPOOL OFF
CLEAR COLUMNS
SET NEWPAGE 0 VERIFY OFF
SET TERMOUT ON PAGES 22 LINES 80 HEADING ON
SET TERMOUT ON
UNDEF OUTPUT
PAUSE Press enter to continue
```

<C>SQL Script to monitor locks waiting on other locks:

```
rem NAME: waiters.sql
rem FUNCTION: Report on sessions waiting for locks
rem HISTORY: MRA 1/12/96 Creation
rem
COLUMN busername        FORMAT a10  HEADING 'Holding|User'
COLUMN wusername        FORMAT a10  HEADING 'Waiting|User'
COLUMN bsession_id                  HEADING 'Holding|SID'
COLUMN wsession_id                  HEADING 'Waiting|SID'
COLUMN mode_held        FORMAT a20  HEADING 'Mode|Held'
COLUMN mode_requested   FORMAT a20  HEADING 'Mode|Requested'
COLUMN lock_id1         FORMAT a20  HEADING 'Lock|ID1'
COLUMN lock_id2         FORMAT a20  HEADING 'Lock|ID2'
COLUMN type                                 HEADING 'Lock|Type'
SET LINES 132 PAGES 59 FEEDBACK OFF ECHO OFF
START title132 'Processes Waiting on Locks Report'
SPOOL rep_out/&db/waiters
SELECT
      holding_session bsession_id,
      waiting_session wsession_id,
      b.username busername,
      a.username wusername,
      c.lock_type type,
      mode_held, mode_requested,
      lock_id1, lock_id2
FROM
      sys.v_$session b,
      sys.dba_waiters c,
      sys.v_$session a
WHERE
      c.holding_session=b.sid and
      c.waiting_session=a.sid
/
SPOOL OFF
PAUSE press enter/return to continue
CLEAR COLUMNS
SET LINES 80 PAGES 22 FEEDBACK ON
TTITLE OFF
```

<C>SQL Script to show sessions causing blocks:

```
rem NAME: blockers.sql
```

```
rem FUNCTION: Show all processes causing a dead lock
rem HISTORY: MRA 1/15/96 Created
rem
COLUMN username         FORMAT a10  HEADING 'Holding|User'
COLUMN session_id                         HEADING 'SID'
COLUMN mode_held        FORMAT a20  HEADING 'Mode|Held'
COLUMN mode_requested   FORMAT a20  HEADING 'Mode|Requested'
COLUMN lock_id1         FORMAT a20  HEADING 'Lock|ID1'
COLUMN lock_id2         FORMAT a20  HEADING 'Lock|ID2'
COLUMN type                         HEADING 'Lock|Type'
SET LINES 132 PAGES 59 FEEDBACK OFF ECHO OFF
START title132 'Sessions Blocking Other Sessions Report'
SPOOL rep_out\&db\blockers
SELECT
      a.session_id,
      username,
      type,
      mode_held,
      mode_requested,
      lock_id1,
      lock_id2
FROM
      sys.v_$session b,
      sys.dba_blockers c,
      sys.dba_locks a
WHERE
      c.holding_session=a.session_id AND
      c.holding_session=b.sid
/
SPOOL OFF
PAUSE press enter/return to continue
CLEAR COLUMNS
SET LINES 80 PAGES 22 FEEDBACK ON
```

<C>SQL Script to generate DDL lock report

```
rem Name: ddl_lock.sql
rem Function: Document DDL Locks currently in use
rem History: MRA 1/15/97 Creation
rem
COLUMN owner            FORMAT a15  HEADING 'User'
COLUMN session_id                   HEADING 'SID'
COLUMN mode_held        FORMAT a20  HEADING 'Lock Mode|Held'
COLUMN mode_requested   FORMAT a20  HEADING 'Lock Mode|Requested'
COLUMN type                         HEADING 'Type|Object'
COLUMN name                         HEADING 'Object|Name'
SET FEEDBACK OFF ECHO OFF PAGES 59 LINES 131
START title132 'Report on All DDL Locks Held'
SPOOL rep_out\&db\ddl_lock
SELECT
       NVL(owner,'SYS') owner,
      session_id,
      name,type,
      mode_held,
      mode_requested
```

```
FROM
      sys.dba_ddl_locks
ORDER BY 2
/
SPOOL OFF
PAUSE press enter/return to continue
CLEAR COLUMNS
SET FEEDBACK ON ECHO ON PAGES 22 LINES 80
TTITLE OFF
```

<C>SQL Script to document DML locks

```
rem NAME: dml_lock.sql
rem FUNCTION: Document DML locks currently in use
rem HISTORY: MRA 1/15/96 Creation
rem
COLUMN owner            FORMAT a15  HEADING 'User'
COLUMN session_id                   HEADING 'SID'
COLUMN mode_held        FORMAT a20  HEADING 'Mode|Held'
COLUMN mode_requested   FORMAT a20  HEADING 'Mode|Requested'
SET FEEDBACK OFF ECHO OFF PAGES 59 LINES 131
START title132 'Report on All DML Locks Held'
SPOOL rep_out\&db\dml_lock
SELECT
      NVL(owner,'SYS') owner,
      session_id,
      name,
      mode_held,
      mode_requested
FROM
      sys.dba_dml_locks
ORDER BY 2
/
SPOOL OFF
PAUSE press enter/return to continue
CLEAR COLUMNS
SET FEEDBACK ON ECHO ON PAGES 22 LINES 80
TTITLE OFF
```

<C>SQL Script to document internal locks:

```
rem NAME: int_lock.sql
rem FUNCTION: Document current internal locks
rem HISTORY: MRA 1/15/96 Creation
rem
COLUMN username         FORMAT a10  HEADING 'Lock|Holder'
COLUMN session_id                   HEADING 'User|SID'
COLUMN lock_type        FORMAT a27  HEADING 'Lock Type'
COLUMN mode_held        FORMAT a10  HEADING 'Mode|Held'
COLUMN mode_requested   FORMAT a10  HEADING 'Mode|Requested'
COLUMN lock_id1         FORMAT a30  HEADING 'Lock/Cursor|ID1'
```

```
COLUMN lock_id2         FORMAT a10  HEADING 'Lock|ID2'
PROMPT 'ALL is all types or modes'
ACCEPT lock PROMPT 'Enter Desired Lock Type: '
ACCEPT mode PROMPT 'Enter Lock Mode: '
SET LINES 132 PAGES 59 FEEDBACK OFF ECHO OFF VERIFY OFF
BREAK ON username
START title132 'Report on Internal Locks Mode: &mode Type: &lock'
SPOOL rep_out\&db\int_locks
SELECT
      NVL(b.username,'SYS') username,
      session_id,lock_type,mode_held,
      mode_requested,lock_id1,lock_id2
FROM
      sys.dba_lock_internal a, sys.v_$session b
WHERE
      UPPER(mode_held) like UPPER('%&mode%') OR
      UPPER('&mode')='ALL' AND
      UPPER(lock_type) like UPPER('%&lock%') OR
      UPPER(mode_held) like UPPER('%&mode%') OR
      UPPER('&mode')='ALL' AND
      UPPER('&lock')='ALL' AND
      a.session_id=b.sid
ORDER BY 1,2
/
SPOOL OFF
PAUSE press enter/return to continue
SET LINES 80 PAGES 22 FEEDBACK ON VERIFY ON
CLEAR COLUMNS
CLEAR BREAKS
UNDEF LOCK
UNDEF MODE
```

<C>SQL Script to report on Oracle events:

```
rem
rem FUNCTION: Generate a report on session events by user
rem
rem NAME:events.sql
rem HISTORY: MRA 6/15/97 Created
rem
COLUMN sid              HEADING Sid
COLUMN event            HEADING Event            FORMAT a40
COLUMN total_waits      HEADING Total|Waits
COLUMN total_timeouts   HEADING Total|Timeouts
COLUMN time_waited      HEADING Time|Waited
COLUMN average_wait     HEADING Average|Wait
COLUMN username         HEADING User
BREAK ON username
START title132 "Session Events By User"
SPOOL rep_out\&db\events
SET LINES 132 PAGES 59
SELECT
      username,
      event,
      total_waits,
```

```
      total_timeouts,
      time_waited,
      average_wait
FROM
      sys.v_$session_event a,
      sys.v_$session b
WHERE
      a.sid= b.sid
ORDER BY 1;
SPOOL OFF
PAUSE Press enter to continue
CLEAR COLUMNS
CLEAR BREAKS
SET LINES 80 PAGES 22
TTITLE OFF
```

<C>SQL Script to document invalid database objects:

```
rem Name: inv_obj.sql
rem Purpose: Show alll invalid objects in database
rem Mike Ault 7/2/96 TreCom/RevealNet
rem
COLUMN object_name      FORMAT A30  HEADING 'Object|Name'
COLUMN owner            FORMAT a10  HEADING 'Object|Owner'
COLUMN last_time        FORMAT a20  HEADING 'Last Change|Date'
SET LINES 80 FEEDBACK OFF PAGES 0 VERIFY OFF
START title80 'Invalid Database Objects'
SPOOL rep_out/&db/inv_obj
SELECT
      owner,
      object_name,
      object_type,
      TO_CHAR(last_ddl_time,'DD-MON-YY hh:mi:ss') Last_time
FROM
      dba_objects
WHERE
      status='INVALID'
/
PAUSE Press enter to continue
SET LINES 80 FEEDBACK ON PAGES 22 VERIFY ON
CLEAR COLUMNS
TTITLE OFF
```

<C>SQL Script to get report on all locks (bonus script, not in book)

```
rem
rem FUNCTION: Report all DB locks
rem
COLUMN osuser           FORMAT a15  HEADING 'User'
COLUMN session_id                   HEADING 'SID'
COLUMN mode_held        FORMAT a20  HEADING 'Mode|Held'
```

```
COLUMN mode_requested   FORMAT a20  HEADING 'Mode|Requested'
COLUMN lock_id1         FORMAT a10  HEADING 'Lock|ID1'
COLUMN lock_id2         FORMAT a10  HEADING 'Lock|ID2'
COLUMN type                         HEADING 'Type|Lock'
SET FEEDBACK OFF ECHO OFF PAGES 59 LINES 131
START title132 'Report on All Locks'
SPOOL rep_out\&db\locks
SELECT NVL(a.osuser,'SYS') osuser,b.session_id,type,
mode_held,mode_requested,
lock_id1,lock_id2
FROM sys.v_$session a, sys.dba_locks b
WHERE
a.sid=b.session_id
ORDER BY 2
/
SPOOL OFF
PAUSE press enter/return to continue
CLEAR COLUMNS
SET FEEDBACK ON PAGES 22 LINES 80
```

<C>SQL Script to show objects which can't get the next extent they need (bonus script):

```
rem
rem NAME: cant_ext.sql
rem FUNCTION Report on objects which cant get next extent
rem
rem MRA RevealNet
rem
SET LINES 132 PAGES 58 VERIFY OFF FEEDBACK OFF
@title132 'Objects Which Cannot Extend'
SPOOL rep_out/&db/cant_ext
SELECT seg.owner, seg.segment_name,
   seg.segment_type, seg.tablespace_name,
   DECODE(seg.segment_type,
      'TABLE', t.next_extent,
      'CLUSTER', c.next_extent,
      'INDEX', i.next_extent,
      'ROLLBACK', r.next_extent)
FROM sys.dba_segments seg,
   sys.dba_tables t,
   sys.dba_clusters c,
   sys.dba_indexes i,
   sys.dba_rollback_segs r
WHERE ((seg.segment_type = 'TABLE'
   AND seg.segment_name = t.table_name
   AND seg.owner = t.owner
   AND NOT EXISTS (SELECT tablespace_name
      FROM dba_free_space free
      WHERE free.tablespace_name = t.tablespace_name
      AND free.bytes >= t.next_extent))
OR (seg.segment_type = 'CLUSTER'
   AND seg.segment_name = c.cluster_name
   AND seg.owner = c.owner
   AND NOT EXISTS (SELECT tablespace_name
      FROM dba_free_space free
```

```
      WHERE free.tablespace_name = c.tablespace_name
      AND free.bytes >= c.next_extent))
OR (seg.segment_type = 'INDEX'
   AND seg.segment_name = i.index_name
   AND seg.owner = i.owner
   AND NOT EXISTS (SELECT tablespace_name
      FROM dba_free_space free
      WHERE free.tablespace_name = i.tablespace_name
      AND free.bytes >= i.next_extent))
OR (seg.segment_type = 'ROLLBACK'
   AND seg.segment_name = r.segment_name
   AND seg.owner = r.owner
   AND NOT EXISTS (SELECT tablespace_name
      FROM dba_free_space free
      WHERE free.tablespace_name = r.tablespace_name
     AND free.bytes >= r.next_extent)))
OR seg.extents = seg.max_extents OR seg.extents = (SELECT
DECODE(value,2048,121,4096,256,8192,502,16384, 1022) FROM v$parameter
WHERE
name='db_block_size')
/
spool off
```

<C>PL/SQL Procedure to check on objects you have entered into the kept_objects table to be pinned (This script included with dbms_revealnet.sql on CD):

```
CREATE OR REPLACE PROCEDURE chk_pin AS
      object_name kept_objects.object_name%TYPE;
      object_status dba_objects.status%TYPE;
      temp_name kept_objects.object_name%TYPE;
      CURSOR get_status(name VARCHAR2) IS
            SELECT status FROM dba_objects
            WHERE object_name=name
            AND object_type IN (
            'PACKAGE','PACKAGE BODY','PROCEDURE');
      CURSOR get_objects IS
            SELECT object_name FROM kept_objects;
BEGIN
  OPEN get_objects;
  FETCH get_objects INTO object_name;
  LOOP
      EXIT WHEN get_objects%NOTFOUND;
      temp_name:=substr(object_name,instr(object_name,'.')+1);
      OPEN get_status(temp_name);
      FETCH get_status INTO object_status;
      LOOP
            EXIT WHEN get_status%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE(rtrim(object_name)||'
'||object_status);
            IF object_status='INVALID' THEN
                  REDO_PIN;
                  EXIT;
            END IF;
            FETCH get_status INTO object_status;
```

```
      END LOOP;
      CLOSE get_status;
      IF object_status='INVALID' THEN
            EXIT;
      END IF;
      FETCH get_objects INTO object_name;
  END LOOP;
  CLOSE get_objects;
END;
```

<C>PL/SQL Procedure to redo object pins if objects found to be invalid or no longer pinned:

```
CREATE OR REPLACE PROCEDURE redo_pin AS
com VARCHAR2(100);
com_cur INTEGER;
processed INTEGER;
i BINARY_INTEGER:=1;
j BINARY_INTEGER:=0;
TYPE objects is
      TABLE OF
      kept_objects.object_name%TYPE
      INDEX BY BINARY_INTEGER;
object OBJECTS;
CURSOR get_objects IS
      SELECT
            object_name
      FROM
            kept_objects;
BEGIN
 OPEN get_objects;
 FETCH get_objects INTO object(i);
LOOP
      EXIT WHEN get_objects%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('UNKEEPING: '||object(i));
      DBMS_SHARED_POOL.UNKEEP(object(i));
      i:=i+1;
      FETCH get_objects INTO object(i);
 END LOOP;
 CLOSE get_objects;
 com_cur:=DBMS_SQL.OPEN_CURSOR;
 com:='ALTER SYSTEM FLUSH SHARED_POOL';
 DBMS_SQL.PARSE(com_cur,com,dbms_sql.v7);
 processed:=DBMS_SQL.EXECUTE(com_cur);
 COMMIT;
 processed:=DBMS_SQL.EXECUTE(com_cur);
 DBMS_SQL.CLOSE_CURSOR(com_cur);
 COMMIT;
 i:=i-1;
 FOR j IN 1..i LOOP
      IF object(j) IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('KEPT: '||object(j));
        DBMS_SHARED_POOL.KEEP(object(j));
      ELSE
        EXIT;
```

```
      END IF;
 END LOOP;
END;
/
```

<C>SQL Script to recompile invalid objects:

```
rem Name: com_proc.sql
rem Function: Create a compile list for invalid procedures
rem
rem MRA 5/1/96
rem
DEFINE cr='chr(10)'
SET HEADING OFF PAGES 0 ECHO OFF TERMOUT OFF FEEDBACK OFF VERIFY OFF
SPOOL recompile.sql
SELECT 'ALTER '||object_type||' '||object_name||' COMPILE;'||&&cr||
'SHOW ERROR'
FROM dba_objects WHERE status='INVALID'
/
SPOOL OFF
SET HEADING ON TERMOUT ON FEEDBACK ON VERIFY ON
UNDEF cr
```

<C>SQL Script to generate a report of objects that may be “hanging” (bonus script):

```
rem
rem FUNCTION: Report objects that need to be recompiled
rem FUNCTION: because they may be hanging
rem FUNCTION: run from SYS only
rem
@title80 "Objects that need Recompilation"
COLUMN obj# HEADING Object|Number
COLUMN name HEADING Object|Name
COLUMN owner# HEADING Owner|Number
SPOOL rep_out\&db\hanging
SELECT distinct o2.obj#,o2.name, o2.owner#
FROM sys.obj$ o,
     sys.dependency$ d,
     sys.obj$ o2
WHERE o.obj# = d.p_obj#
AND o.stime != d.p_timestamp
AND d.d_obj# = o2.obj#
AND o2.status != 5
ORDER BY o2.obj#
/
SPOOL OFF
CLEAR COLUMNS
TTITLE OFF
```

<C>SQL Script to execute explain plan against a provided statement:

```
rem NAME: explain.sql
rem FUNCTION:Q and D script to execute explaing plan on a statement
rem
rem MRA/Revealnet
rem
SET HEADING OFF VERIFY OFF FEEDBACK OFF
EXPLAIN PLAN SET STATEMENT_ID ='&statement_id'
FOR &sql_statement
/
@plan &&statement_id
```

<C>SQL Script to generate plan from plan table:

```
rem
rem NAME: plan.sql
rem FUNCTION: Exectue select on plan table
rem MRA/Revealnet
rem
define id = &1
COLUMN QUERY_PLAN FORMAT A60
SELECT statement_id,LPAD( '  ' , 2*LEVEL) || OPERATION || ' ' ||
OBJECT_NAME QUERY_PLAN
FROM &owner..PLAN_TABLE WHERE STATEMENT_ID = '&&id'
CONNECT BY PRIOR ID = PARENT_ID and
statement_id = '&&id'
START WITH ID=0;
```

<C>Example SQL session to get explain plan from a statement that has been explained:

```
1  SELECT statement_id,LPAD( '  ' , 2*LEVEL) || OPERATION || ' ' ||
OBJECT_NAME QUERY_PLAN
  2  FROM &owner..PLAN_TABLE WHERE STATEMENT_ID = '&&id'
  3  CONNECT BY PRIOR ID = PARENT_ID and
  4  statement_id = '&&id'
  5* START WITH ID=0
Enter value for owner: pic_accum_dbo
old   2: FROM &owner..PLAN_TABLE WHERE STATEMENT_ID = '&&id'
new   2: FROM pic_accum_dbo.PLAN_TABLE WHERE STATEMENT_ID = 'inner'
old   4: statement_id = '&&id'
new   4: statement_id = 'inner'
```

<C>SQL Script to run fprc.sql to rebuild a stored object or objects:

```
REM
REM NAME        : RUN_FPRC.SQL
REM FUNCTION    : Generate and execute the exe_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  'Enter object name:'
PROMPT Working....
SET ECHO OFF HEADING OFF VERIFY OFF FEEDBACK OFF
SELECT name dbname FROM v$database;
SPOOL rep_out\&db\do_fprc.sql
SELECT UNIQUE('START fprc_rct '||a.owner||' '||'"'||a.type||'"'||'
'||a.name)
FROM
      dba_source a, dba_objects b
WHERE
      a.owner LIKE UPPER('&owner') and
      a.type LIKE UPPER('&type') and
      a.name LIKE UPPER('&name')
      AND a.owner=b.owner AND
      a.type=b.object_type AND
      a.name=b.object_name;
SPOOL OFF
SET TERMOUT OFF
SPOOL rep_out\&db\exe_fprc.sql
START rep_out\&db\do_fprc.sql
SPOOL OFF
UNDEF OWNER
UNDEF TYPE
UNDEF NAME
CLEAR COLUMNS
SET HEADING ON VERIFY ON FEEDBACK ON
```

<C>SQL Script to rebuild packages, package bodies, procedures, functions from Data Dictionary

```
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 80 PAGES 0 HEADING OFF
SET RECSEP OFF SPACE 0
COLUMN text FORMAT a79 WORD_WRAP
```

```
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;
SELECT '/' FROM dual;
```

<C>Script to create a procedure for single table analysis:

```
CREATE OR REPLACE PROCEDURE analyze_table(table_name in VARCHAR2) AS
CURSOR get_index(tab_name VARCHAR2) is
   SELECT index_name FROM user_indexes
   WHERE table_name=tab_name;
i_name      index_stats.name%TYPE;
i_stats     index_stats%ROWTYPE;
cur1        INTEGER;
cur2        INTEGER;
processed   INTEGER;
com_strng VARCHAR2(90);
BEGIN
   com_strng:='ANALYZE TABLE '||table_name||' ESTIMATE STATISTICS SAMPLE
20 PERCENT';
   cur1:=dbms_sql.open_cursor;
   dbms_sql.parse(cur1,com_strng,dbms_sql.v7);
   processed:=dbms_sql.execute(cur1);
   dbms_sql.close_cursor(cur1);
   OPEN get_index(table_name);
   FETCH get_index INTO i_name;
   LOOP
           EXIT WHEN get_index%NOTFOUND;
           cur2:=dbms_sql.open_cursor;
           com_strng:='ANALYZE INDEX '||i_name||' VALIDATE STRUCTURE';
           dbms_sql.parse(cur2,com_strng,dbms_sql.v7);
           processed:=dbms_sql.execute(cur2);
           dbms_sql.close_cursor(cur2);
          INSERT INTO ind_stat_tab SELECT * FROM index_stats;
   END LOOP;
```

```
END;
/
```

<C>SQL Script to use DBMS_UTILITY package to analyze all schema in the database:

```
SET HEADING OFF VERIFY OFF PAGES 0 FEEDBACK OFF
TTITLE OFF
SPOOL analz_sch.sql
SELECT DISTINCT 'EXECUTE dbms_utility.analyze_schema('||chr(39)||
owner||chr(39)||','||chr(39)||'&METHOD'||chr(39)||','||&NUM_OF_ROWS||',&
PERCENT_TO_USE);'
FROM dba_tables WHERE owner NOT IN ('SYS','SYSTEM')
AND owner = UPPER(‘&owner_name’)
/
SPOOL OFF
SPOOL analz_sch.log
START analz_sch.sql
SPOOL OFF
```

<C>PL/SQL Procedure to analyze tables when contents increase/decrease >30%:

```
CREATE OR REPLACE PROCEDURE check_tables (owner_name in varchar2) AS

CURSOR get_tab_count (own varchar2) IS
        SELECT table_name, nvl(num_rows,1)
        FROM dba_tables
        WHERE owner = upper(own);

tab_name    VARCHAR2(32);
rows        INTEGER;
string      VARCHAR2(255);
cur         INTEGER;
ret         INTEGER;
row_count   INTEGER;
com_string  VARCHAR2(255);

BEGIN
OPEN get_tab_count (owner_name);
LOOP
        FETCH get_tab_count INTO tab_name, rows;
        IF rows=0 THEN
           rows:=1;
        END IF;
EXIT WHEN get_tab_count%NOTFOUND;
dbms_output.put_line('Table name: '||tab_name||' rows:
'||to_char(rows));
```

```
com_string :=
        'SELECT COUNT(*) FROM '||tab_name;
   cur := dbms_sql.open_cursor;
   dbms_sql.parse(cur,com_string,dbms_sql.v7);
   dbms_sql.define_column(cur,1,row_count);
   ret := dbms_sql.execute(cur);
   IF dbms_sql.fetch_rows(cur)>0 THEN
    dbms_sql.column_value(cur,1,row_count);
   END IF;
   dbms_sql.close_cursor(cur);
   IF row_count=0 THEN
        row_count:=1;
   END IF;
dbms_output.put_line('Row count for '||tab_name||':
'||to_char(row_count));
dbms_output.put_line('Ratio: '||to_char(row_count/rows));
        IF ABS((row_count/rows))>1.42 THEN
           string :=
   'ANALYZE TABLE '||tab_name||' ESTIMATE STATISTICS SAMPLE 20 PERCENT';
           cur := dbms_sql.open_cursor;
           dbms_sql.parse(cur,string,dbms_sql.v7);
           ret := dbms_sql.execute(cur)  ;
           dbms_sql.close_cursor(cur);
           dbms_output.put_line(' Table: '||tab_name||' had to be
analyzed.');
        END IF;
END LOOP;
CLOSE get_tab_count;

EXCEPTION
   WHEN OTHERS THEN
      raise_application_error(-20002,'Error in analyze:
'||to_char(sqlcode)||' on '||tab_name,TRUE);
      IF dbms_sql.is_open(cur) THEN
        dbms_sql.close_cursor(cur);
      END IF;
END;
/
```

<C>SQL Script to generate data dictionary cache report:

```
REM
REM NAME                    : DD_CACHE.SQL
REM FUNCTION                : GENERATE REPORT ON DATA DICTIONARY CACHE CONDITION
REM USE                     : FROM SQLPLUS
REM Limitations             : None
REM Revisions:
REM Date                    Modified By    Reason For change
REM 21-AUG-1991             MIKE AULT      INITIAL CREATE
REM 27-NOV-1991             MIKE AULT      ADD % CALCULATION TO REPORT
REM 28-OCT-1992             MIKE AULT      ADD CALL TO TITLE PROCEDURE
REM 21-Jun-1997             MIKE AULT      Updated to ORACLE8REM SET FLUSH OFF
REM SET TERM OFF
SET PAGESIZE 59
SET LINESIZE 79
```

```
COLUMN parameter FORMAT A20
COLUMN type FORMAT a10
COLUMN percent FORMAT 999.99 HEADING "%";
START title80 "DATA DICTIONARY CACHE STATISTICS"
SPOOL rep_out/&db/ddcache.lis
SELECT
              parameter,
              type,
              gets,
              getmisses,
              ( getmisses / gets * 100) percent,
              count,
              usage
FROM
              v$rowcache
WHERE
              gets > 100 AND
              getmisses > 0
ORDER BY parameter;
SPOOL OFF
```

Since we are actually only concerned with a aggregate look at the cache area performance the following query can be substituted into the report to give you an overall health indicator:

```
SELECT (SUM(getmisses) / SUM(gets)) 'DD CACHE MISS RATIO'
FROM V$ROWCACHE;
```

<C>SQL Script to generate library (intenal) cache report:

```
rem Title: libcache.sql
rem FUNCTION: Generate a library cache report
COLUMN namespace                        HEADING "Library Object"
COLUMN gets                             HEADING "Gets"
COLUMN gethitratio      FORMAT 999.99   HEADING "Get Hit%"
COLUMN pins                             HEADING "Pins"
COLUMN pinhitratio      FORMAT 999.99   HEADING "Pin Hit%"
COLUMN reloads                          HEADING "Reloads"
COLUMN invalidations                    HEADING  "Invalidations"
COLUMN db               FORMAT a10
SET PAGES 58 LINES 80
START title80 "Library Caches Report"
DEFINE output = rep_out\&db\lib_cache
SPOOL &output
SELECT
      namespace,gets,
      gethitratio*100 gethitratio,
      pins,
      pinhitratio*100 pinhitratio,
      reloads,invalidations
FROM
```

```
      v$librarycache
/
SPOOL OFF
PAUSE Press enter to continue
SET PAGES 22 LINES 80
TTITLE OFF
UNDEF output
```

<C>SQL Script to monitor shared pool for objects with high disk read counts:

```
rem Name: sqldrd.sql
rem Function: retrun the sql statements from the shared area with
rem Function: highest disk reads
rem History: Presented in paper 35 at IOUG-A 1997, converted for
rem use 6/24/97 MRA
rem
DEFINE access_level = 1000 (NUMBER)
COLUMN parsing_user_id  FORMAT 9999999     HEADING 'User Id'
COLUMN executions       FORMAT 9999        HEADING 'Exec'
COLUMN sorts            FORMAT 99999       HEADING 'Sorts'
COLUMN command_type     FORMAT 99999       HEADING 'CmdT'
COLUMN disk_reads       FORMAT 999,999,999 HEADING 'Block Reads'
COLUMN sql_text         FORMAT a40  HEADING 'Statement' WORD_WRAPPED
SET LINES 130 VERIFY OFF FEEDBACK OFF
START title132 'SQL Statements With High Reads'
SPOOL rep_out/&db/sqldrd.lis
SELECT
      parsing_user_id, executions,
      sorts,command_type,
      disk_reads,sql_text
FROM
      v$sqlarea
WHERE
      disk_reads > &&access_level
ORDER BY
      disk_reads;
SPOOL OFF
SET LINES 80 VERIFY ON FEEDBACK ON
```

<C>SQL Script to generate SQL Area memory summary report:

```
rem
rem FUNCTION: Generate a summary of SQL Area Memory Usage
rem FUNCTION: uses the sqlsummary view.
rem           showing user SQL memory usage
rem
rem sqlsum.sql
rem
COLUMN areas                                  HEADING Used|Areas
COLUMN sharable         FORMAT 999,999,999    HEADING Shared|Bytes
COLUMN persistent       FORMAT 999,999,999    HEADING Persistent|Bytes
```

```
COLUMN runtime          FORMAT 999,999,999      HEADING Runtime|Bytes
COLUMN username         FORMAT A15              HEADING "User"
START TITLE80 "USERS SQL AREA MEMORY USE"
SPOOL rep_out\&db\sqlsum
SET PAGES 59 LINES 80
BREAK ON REPORT
COMPUTE SUM OF sharable ON REPORT
COMPUTE SUM OF persistent ON REPORT
COMPUTE SUM OF runtime ON REPORT
SELECT
      username,
      SUM(sharable_mem) Sharable,
      SUM( persistent_mem) Persistent,
      SUM( runtime_mem) Runtime ,
      COUNT(*) Areas
FROM
      sql_summary
GROUP BY
      username
ORDER BY
      2;
SPOOL OFF
PAUSE Press enter to continue
CLEAR COLUMNS
CLEAR BREAKS
SET PAGES 22 LINES 80
TTITLE OFF

The report uses the following view:

CREATE OR REPLACE VIEW sql_summary AS
SELECT username, sharable_mem, persistent_mem, runtime_mem
FROM sys.v_$sqlarea a, dba_users b
WHERE a.parsing_user_id = b.user_id;
```

<C>SQL Script to show objects in SQL Area for a specific user:

```
rem
rem FUNCTION: Generate a report of SQL Area Memory Usage
rem           showing SQL Text and memory catagories
rem
rem sqlmem.sql
rem
COLUMN sql_text      FORMAT a40   HEADING Text word_wrapped
COLUMN sharable_mem               HEADING Shared|Bytes
COLUMN persistent_mem             HEADING Persistent|Bytes
COLUMN parse_calls                HEADING Parses
COLUMN users         FORMAT a15   HEADING "User"
COLUMN executions                 HEADING "Executions"
START title132 "Users SQL Area Memory Use"
SPOOL rep_out\&db\sqlmem
SET LONG 1000 PAGES 59 LINES 132
BREAK ON users
```

```
COMPUTE SUM OF sharable_mem ON users
COMPUTE SUM OF persistent_mem ON users
COMPUTE SUM OF runtime_mem ON users
SELECT username users, sql_text, Executions, parse_calls, sharable_mem,
persistent_mem
FROM sys.v_$sqlarea a, dba_users b
WHERE a.parsing_user_id = b.user_id
AND b.username LIKE UPPER('%&user_name%')
ORDER BY 1;
SPOOL OFF
PAUSE Press enter to continue
CLEAR COLUMNS
CLEAR COMPUTES
CLEAR BREAKS
SET PAGES 22 LINES 80
```

<C>Hit Ratio Table Creation Script:

```
      create table hit_ratios (
            CHECK_DATE        DATE,
            CHECK_HOUR        NUMBER,
            DB_BLOCK_GETS     NUMBER,
            CONSISTENT        NUMBER,
            PHY_READS         NUMBER,
            HITRATIO          NUMBER,
            PERIOD_HIT_RATIO  NUMBER,
            PERIOD_USAGE      NUMBER,
            USERS             NUMBER)
      storage (initial 10k next 10k pctincrease 0);
```

<C>SQL Script to be used to execute hitratio procedure from cron or queue:

```
REM
REM NAME    :RUN_B_HRATIO.SQL
REM PURPOSE :RUN PL/SQL PROCEDURE TO LOAD HIT RATIO AND USAGE DATA
REM USE     :FROM RUN_B_HRATIO.COM
REM Limitations         : None
REM Revisions:
REM         Date        Modified By Reason For change
REM         10-JUL-1992 M. AULT     INITIAL CREATE
REM         22-Jun-1997
execute hitratio;
exit
```

<C>PL/SQL Script to build hitratio procedure:

```
CREATE OR REPLACE PROCEDURE HITRATIO IS
    c_date DATE;
```

```
    c_hour NUMBER;
    h_ratio NUMBER;
    con_gets NUMBER;
    db_gets NUMBER;
    p_reads NUMBER;
    stat_name CHAR(64);
    temp_name CHAR(64);
    stat_val NUMBER;
    users  NUMBER;
BEGIN
  SELECT TO_CHAR(sysdate,'DD-MON-YY') INTO c_date FROM DUAL;
  SELECT TO_CHAR(sysdate,'HH24') INTO c_hour FROM DUAL;
  SELECT
        name, value
  INTO
        temp_name, stat_val
  FROM
        v$sysstat
  WHERE
        NAME = 'db block gets';
  db_gets:=stat_val;
  dbms_output.put_line(temp_name||'='||to_char(db_gets));
  SELECT
        name, value
  INTO
        temp_name, stat_val
  FROM
        v$sysstat
  WHERE
        name = 'consistent gets';
  con_gets:=stat_val;
  dbms_output.put_line(temp_name||'='||to_char(con_gets));
  SELECT
        name, value
  INTO
        temp_name, stat_val
  FROM
        v$sysstat
  WHERE
        name = 'physical reads';
  p_reads:=stat_val;
  dbms_output.put_line(temp_name||'='||to_char(p_reads));
  SELECT COUNT(*)
  INTO users
  FROM v$session
  WHERE username IS NOT NULL;
  dbms_output.put_line('Users='||to_char(users));
    H_RATIO := (((DB_GETS+CON_GETS-p_reads)/(DB_GETS+CON_GETS))*100);
  dbms_output.put_line('h_ratio='||to_char(h_ratio));
    INSERT INTO  hit_ratios
      VALUES (c_date,c_hour,db_gets,con_gets,p_reads,h_ratio,0,0,users);
  COMMIT;
  UPDATE hit_ratios SET period_hit_ratio =
    (SELECT ROUND((((h2.consistent-h1.consistent)+(h2.db_block_gets-h1.db_block_gets)-
        (h2.phy_reads-h1.phy_reads))/((h2.consistent-h1.consistent)+
        (h2.db_block_gets-h1.db_block_gets)))*100,2)
     FROM hit_ratios h1, hit_ratios h2
     WHERE h2.check_date = hit_ratios.check_date
      AND h2.check_hour = hit_ratios.check_hour
      AND ((h1.check_date = h2.check_date AND h1.check_hour+1 = h2.check_hour)
      OR(h1.check_date+1 = h2.check_date AND h1.check_hour = '23' AND h2.check_hour='0')))
  WHERE period_hit_ratio = 0;
  COMMIT;
  UPDATE hit_ratios SET period_usage =
    (SELECT ((h2.consistent-h1.consistent)+(h2.db_block_gets-h1.db_block_gets))
     FROM hit_ratios h1, hit_ratios h2 where h2.check_date = hit_ratios.check_date
       AND h2.check_hour = hit_ratios.check_hour
       AND ((h1.check_date = h2.check_date AND h1.check_hour+1 = h2.check_hour)
       OR (h1.check_date+1 = h2.check_date
       AND h1.check_hour = '23' and h2.check_hour='0')))
  WHERE period_USAGE = 0;
  COMMIT;
```

```
  EXCEPTION
    WHEN ZERO_DIVIDE THEN
    INSERT INTO  hit_ratios  VALUES (c_date,c_hour,db_gets,con_gets,p_reads,0,0,0,users);
    COMMIT;
END;
/
```

<C>PL/SQL to submit hitratio procedure for hourly processing by dbms_job instead of external cron or queue:

```
DECLARE
jobno NUMBER;
BEGIN
dbms_job.submit (jobno, 'HITRATIO;',sysdate,'sysdate+1');
dbms_output.put_line(TO_CHAR(jobno));
END;
```

Note: You must put a semi-colon at the end of the 'HITRATIO' statement.

<C>SQL Script to generate hit ratio report:

```
REM
REM NAME    :HRSUMM.SQL
REM FUNCTION:GENERATE SUMMARY REPORT OF PERIOD HIT RATIOS AND USAGE
REM FUNCTION:BETWEEN TWO DATES
REM USE     :FROM SQLPlus
REM Limitations         : None
REM Revisions:
REM         Date        Modified By Reason For change
REM         10-JUL-1992 M.AULT      INITIAL CREATE
REM         23-Jun-1997 M.AULT      Verify against 8
REM
SET VERIFY OFF PAGES 58 NEWPAGE 0
START title80 "HIT RATIO AND USAGE FOR &&CHECK_DATE1 TO &&CHECK_DATE2"
DEFINE output = rep_out/&db/hrsumm.lis
SPOOL &output
SELECT
            check_date,
            check_hour,
            period_hit_ratio,
            period_usage,
            users
FROM
            hit_ratios
WHERE
            check_date BETWEEN '&&check_date1' AND '&&check_date2'
ORDER BY
            check_date,check_hour;
SPOOL OFF
PAUSE Press return to continue
```

<C>SQL Script to generate ascii graph of hit ratio results:

```
REM
REM NAME    :HRATIO_REPORT.SQL
REM PURPOSE:CREATE PLOT OF PERIOD HIT RATIO FOR 1 DAY
REM USE       :FROM STATUS_REPORTS.COM
REM Limitations : None
REM Revisions:
REM         Date        Modified By Reason For change
REM         10-JUL-1992 M. AULT     INITIAL CREATE
REM         23-Jun-1997 M. Ault     Verify for 8
REM
rem host SET TERM/WID=132 REM: For VMS only, won't work under UNIX
SET LINES 131 NEWPAGE 0 VERIFY OFF PAGES 180 SPACE 0 FEEDBACK OFF
COLUMN hr FORMAT 99
START title132 "Period HR for &&check_date1 TO &&check_date2"
DEFINE OUTPUT = 'rep_out/&db/phrgrph.lis'
SPOOL &output
SELECT
    check_hour hr,
    DECODE(round(period_hit_ratio),0,'o',null) zchk0,
    DECODE(round(period_hit_ratio),1,'o',null) chk1,
    DECODE(round(period_hit_ratio),2,'o',null) chk2,
    DECODE(round(period_hit_ratio),3,'o',null) chk3,
    DECODE(round(period_hit_ratio),4,'o',null) chk4,
    DECODE(round(period_hit_ratio),5,'o',null) chk5,
            .
            .NOTE: fill in other 89 statments
            .Before you run this!
            .
    DECODE(round(period_hit_ratio),94,'o',null) chk94,
    DECODE(round(period_hit_ratio),95,'o',null) chk95,
    DECODE(round(period_hit_ratio),96,'o',null) chk96,
    DECODE(round(period_hit_ratio),97,'o',null) chk97,
    DECODE(round(period_hit_ratio),98,'o',null) chk98,
    DECODE(round(period_hit_ratio),99,'o',null) chk99,
    DECODE(round(period_hit_ratio),100,'o',null) chk100
FROM hit_ratios
WHERE check_date BETWEEN '&&check_date1' AND '&&check_date2'
ORDER BY check_date,check_hour;
SPOOL OFF
PAUSE press return to continue
rem host SET TERM/WID=80  rem: Only for VMS, will not work on UNIX
```

<C>SQL Selects to get data about incrementing db_block_buffers parameter:

First query gives basic data over the entire monitored interval:

```
SELECT SUM(count) "interval total"
      FROM v$kcbrbh
      WHERE indx BETWEEN ( interval start, interval end);
```

The second gives more detailed information and is the suggested method. It provides summation over several intervals and gives the DBA more detail upon which to base their choice of number of buffers to add:

```
SELECT
  50*TRUNC(indx/50)+1||' to '||50 * (TRUNC(indx/50)+1) "interval",
  SUM(count) "Buffer Cache Hits"
FROM  sys.x$kcbrbh
GROUP BY TRUNC(indx/50);
```

<C>SQL Script to report on incrementing SGA statistics:

```
rem  ****************************************************************
rem
rem  NAME: SGA_INC.sql
rem
rem  HISTORY:
rem  Date         Who                What
rem  --------     ----------------  ------------------------------------
rem  10/25/92  Cary Millsap         Creation
rem  01/07/93  Michael Brouillette  Switched to title80
rem  06/05/93  Mike Ault            Added capability to use interval
rem
rem  FUNCTION: Examine the statistice in the X$KCBRBH table with the
rem  intent to increase the size of the SGA.
rem
rem
****************************************************************
**
COLUMN bufval NEW_VALUE nbuf NOPRINT
COLUMN thits NEW_VALUE tot_hits NOPRINT
SELECT  value  bufval
FROM  v$parameter
WHERE
  LOWER(name) = 'db_block_lru_extended_statistics';
SELECT SUM(count)  thits FROM v$kcbrbh;
START title80 "Prospective Hits if &nbuf Cache Buffers were Added"
COLUMN interval   FORMAT         a20 JUSTIFY c HEADING 'Buffers'
COLUMN cache_hits FORMAT 999,999,990 JUSTIFY c HEADING -
  'Cache Hits that would have been|gained by adding Buffers'
COLUMN cum FORMAT 99.99 HEADING 'Percent of Gain'
SET TERMOUT OFF FEEDBACK OFF VERIFY OFF ECHO OFF
SPOOL rep_out/&db/sga_inc.lis
```

```
SELECT
lpad(TO_CHAR((&nbuf/&incr)*TRUNC(indx/(&nbuf/&&incr))+1,'999,990'),8)||'
to '||
LPAD(TO_CHAR((&nbuf/&&incr)*(TRUNC(indx/(&nbuf/&&incr))+1),'999,990'),8)
interval,
  SUM(count) cache_hits, SUM(count)/&tot_hits * 100 cum
FROM v$kcbrbh
GROUP BY
  TRUNC(indx/(&nbuf/&&incr));
SPOOL OFF
SET TERMOUT ON FEEDBACK 15 VERIFY ON
UNDEF NBUF
```

<C>SQL Selects to show results of decrementing SGA:

```
SELECT SUM(count) "Hit Misses"
      FROM x$kcbcbh
      WHERE indx >= 100;
```

To summarize data over intervals of buffers, a select similar to the following could be used:

```
SELECT 10*TRUNC(indx/10)+1||' to '||10*(TRUNC(indx/10)+1) "Interval",
      SUM(copunt) 'Buffer Hits'
FROM x$kcbcbh
WHERE indx > 0
GROUP BY TRUNC(indx/10);
```

<C>SQL Script to show results of decrementing SGA Buffers:

```
rem  ================================================================
rem  NAME: SGA_DEC.sql
rem
rem  HISTORY:
rem  Date              Who                    What
rem  --------          -------------------    ------------------------
rem  10/25/92          Cary Millsap           Creation
rem  01/07/93          Michael Brouillette    Switched to title80
rem  06/05/93          Mike Ault   Added selectable ranges
rem  FUNCTION: Examine statistics in the X$KCBCBH table with intent to
rem            shrink the SGA.
rem  ================================================================
```

```
COLUMN bufval NEW_VALUE nbuf NOPRINT
COLUMN thits NEW_VALUE tot_hits NOPRINT
SELECT value  bufval
FROM v$parameter
WHERE
  LOWER(name) = 'db_block_buffers';
SELECT SUM(count) thits
FROM x$kcbhcbh;
START title80 "Lost Hits if &nbuf Cache Buffers were Removed"
COLUMN interval   FORMAT         a20 JUSTIFY c HEADING 'Buffers'
COLUMN cache_hits FORMAT 999,999,990 JUSTIFY c HEADING -
  'Hits that would have been lost|had Cache Buffers been removed'
COLUMN cum FORMAT 99.99 'Percent of loss'
SET TERMOUT OFF FEEDBACK OFF VERIFY OFF ECHO OFF
SPOOL rep_out/&db/sga_dec.lis
SELECT
  LPAD(to_char(&&incr*trunc(indx/&&incr)+1,'999,990'),8)||' to '||
  LPAD(to_char(&&incr*(trunc(indx/&&incr)+1),'999,990'),8) interval,
  SUM(count)  cache_hits,
  SUM(count)/&tot_hits * 100 cum
FROM x$kcbcbh
WHERE indx > 0
GROUP BY
  TRUNC(indx/&&incr) ;
SPOOL OFF
SET TERMOUT ON FEEDBACK 15 VERIFY ON
```

<C>SQL Script to report on file IO efficiency:

```
REM
REM NAME      :FILE_EFF.SQL
REM PURPOSE :GENERATE FILE IO EFFICIENCIES REPORT
REM USE     :FROM STATUS_REPORTS.COM
REM Limitations :MUST BE RUN FROM ORACLE DBA ACCOUNT
REM Revisions:
REM Date                Modified By Reason For change
REM 10-JUL-1992         M. AULT     INITIAL CREATE
REM 07-JUN-1993         M.AULT      Added reads to writes, reformatted
REM 23-Jun-1997         M.Ault      kcffio went away, rewrote to use
REM                                 existing views/tables
SET PAGES 58 NEWPAGE 0
SET LINES 131
COLUMN eff  FORMAT A6               HEADING '% Eff'
COLUMN rw   FORMAT 9,999,999        HEADING 'Phys Block|read/writes'
COLUMN ts   FORMAT A22              HEADING 'Tablespace Name'
COLUMN name FORMAT A40              HEADING ‘File Name’
start title132 "FILE IO EFFICIENCY"
BREAK ON ts
DEFINE OUTPUT = 'rep_out/&db/file_io.lis'
spool &OUTPUT
SELECT
    f.tablespace_name ts,
     f.file_name name,
    v.phyreads+v.phywrts rw,
    TO_CHAR(DECODE(v.phyblkrd,0,null,
```

```
    ROUND(100*(v.phyrds+v.phywrts)/(v.phyblkrd+v.phyblkwrt),2))) eff
FROM dba_data_files f, v$filestat v
WHERE f.file_id=v.file#
ORDER BY 1,file#;
SPOOL OFF
PAUSE Press return to continue
```

<C>SQL Script and PL/SQL procedure code to generate a calculated statistics report, uses DBA_TEMP table (full script of cre_tab.sql at end of appendix).

```
REM
REM NAME    : DO_CALSTAT.SQL
REM FUNCTION :Generate calculated statisitics report using
REM FUNCTION :just_statistics procedure
REM USE     :FROM STATUS.SQL or SQLPLUS
REM Limitations         :
REM Revisions:
REM Date                Modified By Reason For change
REM 05-MAY-1992         Mike Ault   Initial Creation
REM 23-JUN-1997         Mike Ault   Updated to V8
REM
SET PAGES 58  NEWPAGE 0
EXECUTE just_statistics
START title80 "CALCULATED STATISTICS REPORT"
DEFINE output = rep_out\&db\cal_stat.lis
SPOOL &output
SELECT * FROM dba_temp;
SPOOL OFF
```

<C>Listing of just_statistics - The called PL/SQL procedure

```
CREATE OR REPLACE PROCEDURE just_statistics AS
    start_date          DATE;
    dd_ratio            NUMBER := 0;
    r_calls             NUMBER := 0;
    h_ratio             NUMBER := 0;
    suhw_cont           NUMBER := 0;
    subw_cont           NUMBER := 0;
    uhw_cont            NUMBER := 0;
    ubw_cont            NUMBER := 0;
    db_gets             NUMBER := 0;
    con_gets            NUMBER := 0;
    p_reads             NUMBER := 0;
    suh_waits           NUMBER := 0;
    sub_waits           NUMBER := 0;
    uh_waits            NUMBER := 0;
    ub_waits            NUMBER := 0;
    u_calls             NUMBER := 0;
    calls_u             NUMBER := 0;
    rlog_wait           NUMBER := 0;
    stat_name           VARCHAR2(64);
```

```
    temp_name           VARCHAR2(64);
    stat_val            NUMBER := 0;
    temp_value          NUMBER := 0;
    version             varchar2(9);
CURSOR get_latch IS
  SELECT a.name,100.*b.sleeps/b.gets
  FROM v$latchname a, v$latch b
  WHERE a.latch# = b.latch# and b.sleeps > 0;
CURSOR get_totals IS
  SELECT object_type,count(*)
  FROM dba_objects
  WHERE owner not IN ('SYS','SYSTEM')
  GROUP BY object_type
  ORDER BY object_type;
CURSOR get_stat(stat IN VARCHAR2) IS
  SELECT name,value
  FROM  v$sysstat
  WHERE name = stat;
CURSOR get_count(stat IN VARCHAR2) IS
  SELECT class,"COUNT"
  FROM v$waitstat
  WHERE class = stat_name;
BEGIN
  DELETE dba_temp;
BEGIN
DBMS_REVEALNET.STARTUP_DATE(start_date);
  IF start_date IS NOT NULL THEN
   INSERT INTO dba_temp VALUES
   'Startup Date:'||TO_CHAR(start_date,'dd-mon-yy hh24:mi:ss'),0,1);
  ELSE
   INSERT INTO dba_temp values ('Startup Date: unknown',0,1);
  END IF;
END;
BEGIN
  stat_name := 'recursive calls';
            OPEN get_stat(stat_name);
            FETCH get_stat INTO temp_name, r_calls;
            CLOSE get_stat;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
CLOSE get_stat;
END;
BEGIN
  stat_name := 'DATA DICTIONARY MISS %';
  SELECT
    stat_name,(SUM(getmisses)/SUM(gets))*100 INTO temp_name,dd_ratio
  FROM v$rowcache;
  INSERT INTO dba_temp VALUES (stat_name, dd_ratio,17);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
            INSERT INTO dba_temp VALUES (stat_name,0,17);
    COMMIT;
END;
BEGIN
  stat_name := 'user calls';
            OPEN  get_stat(stat_name);
            FETCH get_stat INTO temp_name, u_calls;
            CLOSE get_stat;
```

```
EXCEPTION
    WHEN NO_DATA_FOUND THEN
CLOSE get_stat;
END;
BEGIN
  stat_name := 'db block gets';
            OPEN get_stat(stat_name);
            FETCH get_stat INTO temp_name, db_gets;
            CLOSE get_stat;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
CLOSE get_stat;
END;
BEGIN
  stat_name := 'consistent gets';
            OPEN get_stat(stat_name);
            FETCH get_stat INTO temp_name, con_gets;
            CLOSE get_stat;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
CLOSE get_stat;
END;
BEGIN
  stat_name := 'physical reads';
            OPEN get_stat(stat_name);
            FETCH get_stat INTO temp_name, p_reads;
            CLOSE get_stat;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
CLOSE get_stat;
END;
BEGIN
  stat_name := 'system undo header';
            OPEN get_count(stat_name);
            FETCH get_count INTO temp_name, suh_waits;
            CLOSE get_count;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
CLOSE get_count;
END;
BEGIN
  stat_name := 'system undo block';
            OPEN get_count(stat_name);
            FETCH get_count INTO temp_name, sub_waits;
            CLOSE get_count;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
CLOSE get_count;
END;
BEGIN
  stat_name := 'undo header';
            OPEN get_count(stat_name);
            FETCH get_count INTO temp_name, uh_waits;
            CLOSE get_count;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
CLOSE get_count;
END;
```

```
BEGIN
  stat_name := 'undo block';
            OPEN get_count(stat_name);
            FETCH get_count INTO temp_name, ub_waits;
            CLOSE get_count;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
CLOSE get_count;
END;
BEGIN
    calls_u := (r_calls/u_calls);
    h_ratio := ((db_gets+con_gets)/(db_gets+con_gets+p_reads));
    suhw_cont  := (suh_waits/(db_gets+con_gets)*100);
    subw_cont  := (sub_waits/(db_gets+con_gets)*100);
    uhw_cont  := (uh_waits/(db_gets+con_gets)*100);
    ubw_cont  := (ub_waits/(db_gets+con_gets)*100);
    stat_name := 'RECURSIVE CALLS PER USER';
  INSERT INTO dba_temp VALUES (stat_name, calls_u,18);
    stat_name := 'CUMMULATIVE HIT RATIO';
  INSERT INTO dba_temp VALUES (stat_name, H_RATIO,2);
    stat_name := 'SYS UNDO HDR WAIT CONTENTION %';
  INSERT INTO dba_temp VALUES (stat_name, suhw_cont,3);
    stat_name := 'SYS UNDO BLK WAIT CONTENTION %';
  INSERT INTO dba_temp VALUES (stat_name, subw_cont,3);
    stat_name := 'UNDO HDR WAIT CONTENTION %';
  INSERT INTO dba_temp VALUES (stat_name, uhw_cont,3);
    stat_name := 'UNDO BLK WAIT CONTENTION %';
  INSERT INTO dba_temp VALUES (stat_name, ubw_cont,3);
    stat_name := 'freelist';
            OPEN get_count(stat_name);
            FETCH get_count INTO temp_name, stat_val;
            CLOSE get_count;
  stat_name := 'FREE LIST CONTENTION RATIO';
    INSERT INTO dba_temp VALUES (stat_name,
stat_val/(db_gets+con_gets),18);
EXCEPTION
    WHEN ZERO_DIVIDE THEN
            INSERT INTO dba_temp VALUES (stat_name,0,32);
            CLOSE get_count;
    COMMIT;
    WHEN NO_DATA_FOUND THEN
            INSERT INTO dba_temp VALUES (stat_name,0,32);
            CLOSE get_count;
    COMMIT;
END;
BEGIN
version:=DBMS_REVEALNET.RETURN_VERSION;
IF substr(version,1,5) in
('7.2.3','7.3.0','7.3.1','7.3.2','7.3.3','8.0.0',
'8.0.0','8.0.1','8.0.2', '8.0.3') THEN
  stat_name := 'LATCH MISS %';
            SELECT (1-((SUM(sleeps)+SUM(immediate_misses))/(
            SUM(gets)+SUM(immediate_misses)+SUM(immediate_gets)))*100)
INTO stat_val
            FROM v$latch;
    INSERT INTO dba_temp VALUES (stat_name, stat_val,4);
END IF;
EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
            INSERT INTO dba_temp VALUES (stat_name,0,4);
    COMMIT;
END;
BEGIN
  stat_name := 'ROLLBACK WAIT %';
            SELECT (SUM(waits)/SUM(gets))*100 INTO stat_val
            FROM v$rollstat;
    INSERT INTO dba_temp VALUES (stat_name, stat_val,5);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
            INSERT INTO dba_temp VALUES (stat_name,0,5);
    COMMIT;
END;
BEGIN
  stat_name := 'LIBRARY RELOAD %';
            SELECT SUM(reloads)/SUM(pins)*100  INTO stat_val
            FROM v$librarycache;
    INSERT INTO dba_temp VALUES (stat_name, stat_val,5);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
            INSERT INTO dba_temp VALUES (stat_name,0,5);
    COMMIT;
END;
BEGIN
  stat_name := 'table fetch by rowid';
            OPEN get_stat(stat_name);
            FETCH get_stat INTO temp_name, stat_val;
            CLOSE get_stat;
    INSERT INTO dba_temp VALUES (stat_name, stat_val,9);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
            INSERT INTO dba_temp VALUES (stat_name,0,9);
CLOSE get_stat;
    COMMIT;
END;
BEGIN
  stat_name:='NON-INDEX LOOKUP RATIO';
  SELECT a.value/(a.value+b.value) INTO stat_val
  FROM v$sysstat a, v$sysstat b
  WHERE a.name='table scans (long tables)'
  AND b.name='table scans (short tables)';
  INSERT INTO dba_temp VALUES (stat_name, stat_val,8);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
            INSERT INTO dba_temp VALUES (stat_name,0,8);
            CLOSE get_stat;
    COMMIT;
END;
BEGIN
  stat_name := 'table fetch continued row';
            OPEN get_stat(stat_name);
            FETCH get_stat INTO temp_name, stat_val;
            CLOSE get_stat;
    INSERT INTO dba_temp VALUES (stat_name, stat_val,14);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
            INSERT INTO dba_temp VALUES (stat_name,0,14);
```

```
            CLOSE get_stat;
    COMMIT;
END;
BEGIN
  stat_name := 'sorts (memory)';
            OPEN get_stat(stat_name);
            FETCH get_stat INTO temp_name, stat_val;
            CLOSE get_stat;
    INSERT INTO dba_temp VALUES (stat_name, stat_val,15);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
            INSERT INTO dba_temp VALUES (stat_name,0,15);
            CLOSE get_stat;
    COMMIT;
END;
BEGIN
  stat_name := 'sorts (disk)';
            OPEN get_stat(stat_name);
            FETCH get_stat INTO temp_name, stat_val;
            CLOSE get_stat;
  INSERT INTO dba_temp VALUES (stat_name, stat_val,16);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
            INSERT INTO dba_temp VALUES (stat_name,0,16);
            CLOSE get_stat;
    COMMIT;
END;
BEGIN
  stat_name := 'redo log space requests';
            OPEN get_stat(stat_name);
            FETCH get_stat INTO temp_name, stat_val;
            CLOSE get_stat;
  INSERT INTO dba_temp VALUES (stat_name, stat_val,6);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
            INSERT INTO dba_temp VALUES (stat_name,0,6);
            CLOSE get_stat;
    COMMIT;
END;
BEGIN
  stat_name := 'redo log space wait time';
            OPEN get_stat(stat_name);
            FETCH get_stat INTO temp_name, stat_val;
            CLOSE get_stat;
  INSERT INTO dba_temp VALUES (stat_name, stat_val, 6);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
            INSERT INTO dba_temp VALUES (stat_name,0,6);
            CLOSE get_stat;
    COMMIT;
END;
BEGIN
  stat_name := 'TOTAL ALLOCATED MEG';
            SELECT SUM(BYTES)/1048576 INTO stat_val
            FROM dba_data_files WHERE
            STATUS = 'AVAILABLE';
  INSERT INTO dba_temp VALUES (stat_name, stat_val,25);
EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
            INSERT INTO dba_temp VALUES (stat_name,0,25);
    COMMIT;
END;
BEGIN
  stat_name := 'TOTAL USED MEG';
            SELECT SUM(BYTES)/1048576 INTO stat_val
            FROM dba_extents;
  INSERT INTO dba_temp VALUES (stat_name, stat_val,26);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
            INSERT INTO dba_temp VALUES (stat_name,0,26);
    COMMIT;
END;
BEGIN
  stat_name := 'TOTAL SGA SIZE';
            SELECT stat_name, SUM(b.value) INTO temp_name, stat_val
            FROM v$sga b;
  INSERT INTO dba_temp VALUES (stat_name, stat_val,31);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
            INSERT INTO dba_temp VALUES (stat_name,0,31);
    COMMIT;
END;
BEGIN
OPEN get_latch;
LOOP
            FETCH get_latch INTO stat_name,stat_val;
            EXIT WHEN get_latch%NOTFOUND;
            INSERT INTO dba_temp VALUES (stat_name, stat_val,33);
END LOOP;
            CLOSE get_latch;
            COMMIT;
END;
BEGIN
OPEN get_totals;
LOOP
            FETCH get_totals INTO stat_name,stat_val;
            EXIT WHEN get_totals%NOTFOUND;
            INSERT INTO dba_temp VALUES (stat_name, stat_val,34);
END LOOP;
            CLOSE get_totals;
            COMMIT;
END;
  COMMIT;
END;
```

<C>SQL Script to report on buffer busy wait contentions:

```
REM
REM NAME: CONTEND.SQL
REM FUNCTION: Shows where possible contention for resources
REM                      in buffer busy waits use to pinpoint additional
REM                      tuning areas.
REM
```

```
REM USE: Called from status
REM
SET VERIFY OFF FEEDBACK OFF
SET PAGES 58
SET LINES 79
START title80 "AREA OF CONTENTION REPORT"
DEFINE output = 'rep_out\&db\contend'
SPOOL &output
SELECT
            class,
            SUM(count) total_waits,
            SUM(time) total_time
FROM
            v$waitstat
GROUP BY
            class;
SPOOL OFF
PAUSE Press return to continue
SET VERIFY ON FEEDBACK ON PAGES 22 LINES 80
TTITLE OFF
```

<C>SQL Script to monitor latch contention:

```
REM
REM NAME        : LTCH7_CO.SQL
REM FUNCTION    : Genereate latch contention report
REM USE         : From SQLPlus or other front end
REM Limitations : None
REM
COLUMN name    FORMAT A30
COLUMN ratio1  FORMAT 999.999
COLUMN ratio2  FORMAT 999.999
SET PAGES 58 NEWPAGE 0
START title80 "LATCH CONTENTION REPORT"
SPOOL rep_out\&db\latchs
SELECT
  a.name,
  100.*b.misses/b.gets ratio1
  100.*b.immediate_misses/(b.immediate_gets+b.immediate_misses) ratio2
FROM
  v$latchname a, v$latch b
WHERE
      a.latch# = b.latch# AND b.misses > 0;
SPOOL OFF
PAUSE PRESS RETURN TO CONTINUE
CLEAR COLUMNS
TTITLE OFF
SET PAGES 22
```

<C>SQL Script to monitor dispatcher processes in MTS:

```
rem Name: mts_disp.sql
rem Function: Generate percent busy report for dispatchers
rem History: MRA Revealnet script
COLUMN protocol FORMAT A9 HEADING ‘Dispatcher|Protocol’
COLUMN busy FORMAT 999.99 HEADING ‘Percent|Busy’
SET FEEDBACK OFF VERIFY OFF LINES 78 PAGES 58
START title80 ‘Dispatcher Status’
SPOOL rep_out\&&db\mts_disp.lis
SELECT network protocol,
      ((SUM(busy]/(SUM(busy)+SUM(idle))*100) busy
FROM v$dispatcher
GROUP BY network;
SPOOL OFF
SET FEEDBACK ON VERIFY ON
TTITLE OFF
```

<C>SQL Script to monitor MTS Dispatcher Wait times:

```
rem Name: mts_wait.sql
rem Function: Generate wait time report for dispatchers
rem History: MRA Revealnet script
COLUMN network FORMAT A9 HEADING ‘Protocol’
COLUMN aw FORMAT A30 HEADING ‘Average Wait Time %’
SET FEEDBACK OFF VERIFY OFF LINES 78 PAGES 58
START title80 ‘Dispatcher Wait Times’
SPOOL rep_out\&&db\mts_wait.lis
SELECT
      NETWORK,
      DECODE (SUM(totalq),0,'No responses',
      SUM(wait)/SUM(totalq)*100||'Seconds Wait Per response') aw
FROM v$queue q, v$dispatcher d
WHERE q.type = 'DISPATCHER' AND
      q.paddr = d.paddr
GROUP BY network;
SPOOL OFF
SET FEEDBACK ON VERIFY ON
TTITLE OFF
```

<C>SQL Script to monitor average dispatcher wait time in MTS:

```
rem Name: mts_awt.sql
rem Function: Generate Average wait time report for dispatchers
rem History: MRA Revealnet script
COLUMN awt FORMAT A30 HEADING ‘Average Wait Time per Request’
SET FEEDBACK OFF VERIFY OFF LINES 78 PAGES 58
START title80 ‘Dispatcher Average Wait Time’
SPOOL rep_out\&&db\mts_awt.lis
SELECT
      DECODE (TOTALQ,0, 'No Requests',
```

```
      (wait/totalq)*100||'Seconds Request Wait') awt
FROM
      v$queue
WHERE
      type = 'COMMON';
SPOOL OFF
SET FEEDBACK ON VERIFY ON LINES 80 PAGES 22
PAUSE Press enter to continue
```

<C>SQL Script to generate hot backup UNIX shell script:

```
rem******* RevealNet Oracle Administration*********************
rem
rem  File: oline_bu.sql
rem
rem  This is a part of the RevealNet Oracle Administration library.
rem  Copyright (C) 1996-97 RevealNet, Inc.
rem  All rights reserved.
rem
rem  For more information, call RevealNet at 1-800-REVEAL4
rem  or check out our Web page: www.revealnet.com
rem
rem  Modifications (Date, Who, Description)
rem
rem FUNCTION: Perform Hot Unix backup
rem
rem ********************************************************
rem
CREATE TABLE bu_temp (line_no NUMBER,line_txt VARCHAR2(2000));
SET VERIFY OFF
DEFINE dest_dir=&1;
DECLARE
CURSOR get_tbsp IS
      SELECT
            tablespace_name
      FROM
            dba_tablespaces;
CURSOR bbu_com (tbsp VARCHAR2) IS
      SELECT
            'ALTER TABLESPACE '||tablespace_name||' BEGIN BACKUP;'
      from
            dba_tablespaces
      WHERE
            tablespace_name=tbsp;
cursor tar_com (tbsp varchar2) is
      SELECT
            '!/bin/tar cvf - '||file_name||'|compress>&&dest_dir/'||
            SUBSTR(file_name,INSTR(file_name,'/',-1,1)+1,
            LENGTH(file_name))||'.Z'
      FROM
            dba_data_files
      WHERE
            tablespace_name=tbsp;
CURSOR ebu_com (tbsp varchar2) IS
      SELECT
```

```
            'ALTER TABLESPACE '||tablespace_name||' END BACKUP;'
      FROM
            dba_tablespaces
      WHERE
            tablespace_name=tbsp;
tbsp_name VARCHAR2(64);
line_num NUMBER:=0;
line_text VARCHAR2(2000);
BEGIN
line_num := line_num+1;
OPEN get_tbsp;
LOOP
      FETCH get_tbsp INTO tbsp_name;
      EXIT WHEN get_tbsp%NOTFOUND;
      OPEN bbu_com (tbsp_name);
      FETCH bbu_com INTO line_text;
      INSERT INTO bu_temp VALUES (line_num,line_text);
      CLOSE bbu_com;
      OPEN tar_com (tbsp_name);
      LOOP
            FETCH tar_com INTO line_text;
            EXIT WHEN tar_com%NOTFOUND;
            line_num:=line_num+1;
            INSERT INTO bu_temp VALUES (line_num,line_text);
      END LOOP;
  CLOSE tar_com;
  OPEN ebu_com(tbsp_name);
  FETCH ebu_com INTO line_text;
  line_num:=line_num+1;
  INSERT INTO bu_temp VALUES (line_num,line_text);
  CLOSE ebu_com;
END LOOP;
  CLOSE get_tbsp;
  SELECT
      'alter system switch logfile;'
  INTO
      line_text
  FROM
      dual;
  line_num:=line_num+1;
  INSERT INTO bu_temp VALUES (line_num,line_text);
  SELECT '!compress '||SUBSTR (value,1,INSTR(value,'/',-1,1))||'*'
  INTO line_text FROM v$parameter WHERE name='log_archive_dest';
  line_num:=line_num+1;
  INSERT INTO bu_temp VALUES (line_num,line_text);
  SELECT '!tar cvf - '||SUBSTR (value,1,INSTR(value,'/',-1,1))||'*.Z'||
  '|compress>&&dest_dir/'||
  SUBSTR(value,instr(value,'/',-1,1)+1,LENGTH(value))||'.Z'
  INTO line_text FROM v$parameter WHERE name='log_archive_dest';
  line_num:=line_num+1;
  INSERT INTO bu_temp VALUES (line_num,line_text);
END;
/
SET VERIFY OFF FEEDBACK OFF HEADING OFF TERMOUT OFF PAGES 0
SET EMBEDDED ON LINES 132
COLUMN line_no NOPRINT
COLUMN dbname NEW_VALUE db NOPRINT
SELECT name dbname FROM v$database;
```

```
SPOOL rep_out\&db\thot_bu.sql
SELECT * FROM bu_temp ORDER BY line_no;
SPOOL OFF
rem uncomment this next line and change
rem directory syntax for UNIX
rem
! sed '1,$ s/ *$//g' rep_out\&db\thot_bu.sql>rep_out\&db\hot_bu.sql
rem
DROP TABLE bu_temp;
SET VERIFY ON FEEDBACK ON HEADING ON TERMOUT ON PAGES 22
SET EMBEDDED OFF LINES 80
CLEAR COLUMNS
undef dest_dir
```

<C>SQL Fragment to extract sessions memory usage (bonus script):

```
SELECT username, value || 'bytes' "Current session memory"
   FROM v$session sess, v$sesstat stat, v$statname name
WHERE sess.sid = stat.sid
   AND stat.statistic# = name.statistic#
   AND name.name = 'session memory'
/
```

<C>SQL Script to monitor stored object statistics(bonus script):

```
rem
rem FUNCTION: Report Stored Object Statistics
rem
COLUMN owner            FORMAT a11        HEADING Schema
COLUMN name             FORMAT a30        HEADING Object|Name
COLUMN namespace                          HEADING Name|Space
COLUMN type                               HEADING Object|Type
COLUMN kept             FORMAT a4         HEADING Kept
COLUMN sharable_mem     FORMAT 999,999    HEADING Shared|Memory
COLUMN executions       FORMAT 999,999    HEADING Executes
SET LINES 132 PAGES 47 FEEDBACK OFF
@title132 'Oracle Objects Report'
BREAK ON owner ON namespace ON type
SPOOL rep_out/&db/o_stat
SELECT
      owner,
      namespace,
      type,
      name,
      sharable_mem,
      loads,
      executions,
      locks,
      pins,
      kept
FROM
```

```
      v$db_object_cache
WHERE
      type NOT IN ('NOT LOADED','NON-EXISTENT')
      AND executions>0
ORDER BY owner,namespace,type,executions desc;
SPOOL OFF
SET LINES 80 PAGES 22 FEEDBACK ON
CLEAR CLOUMNS
CLEAR BREAKS
TTITLE OFF
```

<C>SQL Script to report on “bad” objects (Bonus Script):

```
rem
rem FUNCTION: Report on "bad" objects
rem
rem
COLUMN owner            FORMAT a10        HEADING Schema
COLUMN name             FORMAT a30        HEADING Object|Name
COLUMN namespace                          HEADING Name|Space
COLUMN type                               HEADING Object|Type
COLUMN kept             FORMAT a4         HEADING Kept
COLUMN sharable_mem     FORMAT 999,999    HEADING Shared|Memory
COLUMN executions       FORMAT 9,999      HEADING Executes
SET LINES 132 PAGES 47 FEEDBACK OFF
@title132 'Oracle Objects Report'
BREAK ON owner ON namespace ON type
SPOOL rep_out/&db/o_stat2
SELECT
      owner,
      namespace,
      type,
      name,
      sharable_mem,
      loads,
      executions,
      locks,
      pins,
      kept
FROM
      v$db_object_cache
WHERE
      type IN ('NOT LOADED','NON-EXISTENT')
ORDER BY owner,namespace,type,executions desc;
SPOOL OFF
CLEAR COLUMNS
CLEAR BREAKS
SET LINES 80 PAGES 22 FEEDBACK ON
```

<C>SQL Script to recreate users (bonus script):

```
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 HEADING OFF
SET TERMOUT ON
PROMPT Creating user create script...
SET TERMOUT OFF
rem
rem FUNCTION: Create a use recreate script
rem
COLUMN dbname NEW_VALUE db NOPRINT
SELECT name dbname FROM v$database;
DEFINE cr='chr(10)'
SPOOL rep_out\&db\crt_usrs.sql

SELECT 'CREATE USER '||username||' identified by values
'||password||&&cr||
' DEFAULT TABLESPACE '||default_tablespace||&&cr||
' TEMPORARY TABLESPACE '||temporary_tablespace||&&cr||
' PROFILE '||profile||&&cr||
' QUOTA UNLIMITED ON '||default_tablespace||';'||&&cr x
FROM dba_users
WHERE username not in ('SYS','SYSTEM')
UNION
SELECT 'ALTER USER '||username||&&cr||
'QUOTA '||bytes||' ON '||tablespace_name||';'||&&cr x
FROM dba_ts_quotas
WHERE username NOT IN ('SYS','SYSTEM')
ORDER BY x desc
/
SPOOL OFF
SET VERIFY ON FEEDBACK ON TERMOUT ON PAGESIZE 22 EMBEDDED OFF
SET HEADING ON
CLEAR COLUMNS
UNDEF CR
```

<C>SQL Script to generate user quota report:

```
rem ************************************************************
rem  NAME: quota.sql
```

```
rem  FUNCTION: Print the tablespace quotas of users.
rem ************************************************************
PROMPT Percent signs are wild cards
ACCEPT username PROMPT 'Enter user name or wild card '
PROMPT Print the details of the Users Tablespace Quotas
START title80 "Database Users Space Quotas by Tablespace"
COLUMN un         FORMAT a25        HEADING 'User Name'
COLUMN ta         FORMAT a25        HEADING 'Tablespace'
COLUMN usd        FORMAT 9,999,999  HEADING 'K Used'
COLUMN maxb       FORMAT 9,999,999  HEADING 'Max K '
SET VERIFY OFF FEEDBACK OFF NEWPAGE 0 HEADING ON
SPOOL rep_out\&db\tsquotas
BREAK ON ta skip 2
SELECT
 tablespace_name  ta,
 username         un,
 bytes/1024       usd,
 max_bytes/1024   maxb
FROM dba_ts_quotas
   WHERE username = UPPER('&&username')
ORDER BY tablespace_name,username;
PROMPT End of Report
SPOOL OFF
SET VERIFY ON
UNDEF username
CLEAR BREAKS
CLEAR COLUMNS
CLEAR COMPUTES
SET VERIFY ON FEEDBACK ON HEADING ON
TTITLE OFF
```

<C>PL/SQL--SQL--SQLPLUS Script to rebuild rollback segments:

```
REM rbk_rct.sql
REM
REM FUNCTION: SCRIPT FOR CREATING ROLLBACK SEGMENTS
REM
REM This script must be run by a user with select on the DBA views.
REM
REM This script is intended to run with Oracle7 or Oracle8.
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 rbk_rct.sql
REM
REM FUNCTION: SCRIPT FOR CREATING ROLLBACK SEGMENTS
REM
REM This script must be run by a user with select on the DBA views.
REM
REM This script is intended to run with Oracle7 or Oracle8.
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 back 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 FEEDBACK OFF TERMOUT OFF ECHO OFF PAGES 0
SET TERMOUT ON
SELECT 'Creating rollback segment build script...' from dual;
SET TERMOUT OFF
DEFINE cr='CHR(10)'
CREATE table rb_temp (lineno NUMBER, rb_name VARCHAR2(30),
                text VARCHAR2(800))
/

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_$rollstat
                   WHERE usn=r_no;
   lv_seg_name            sys.dba_rollback_segs.segment_name%TYPE;
   lv_tablespace_name     sys.dba_rollback_segs.tablespace_name%TYPE;
   lv_owner               VARCHAR2(10);
   lv_segment_id          sys.dba_rollback_segs.segment_id%TYPE;
   lv_initial_extent      sys.dba_rollback_segs.initial_extent%TYPE;
   lv_next_extent         sys.dba_rollback_segs.next_extent%TYPE;
   lv_min_extents         sys.dba_rollback_segs.min_extents%TYPE;
   lv_max_extents         sys.dba_rollback_segs.max_extents%TYPE;
   lv_status              sys.dba_rollback_segs.status%TYPE;
   lv_usn                 sys.v_$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_seg_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_seg_name);
write_out(lv_lineno, lv_seg_name, lv_string);
lv_lineno := lv_lineno + 1;
lv_string:='TABLESPACE ' || LOWER(lv_tablespace_name);
write_out(lv_lineno, lv_seg_name, lv_string);
lv_lineno := lv_lineno + 1;
lv_string:='STORAGE ' || '(INITIAL '||lv_initial_extent||' NEXT '||
           lv_next_extent||&&cr||' MINEXTENTS '||lv_min_extents||
           ' MAXEXTENTS '|| lv_max_extents||&&cr||
           ' OPTIMAL ' || lv_optsize || ')' ;
write_out(lv_lineno, lv_seg_name, lv_string);
lv_lineno := lv_lineno + 1;
lv_string:=
'/'||&&cr||'ALTER ROLLBACK SEGMENT '||lv_seg_name||' ONLINE;'||&&cr;
write_out(lv_lineno, lv_seg_name, lv_string);
ELSE
lv_string:='CREATE ' || lv_owner || 'ROLLBACK SEGMENT ' ||
           LOWER(lv_seg_name);
write_out(lv_lineno, lv_seg_name, lv_string);
lv_lineno := lv_lineno + 1;
lv_string:='TABLESPACE ' || LOWER(lv_tablespace_name);
write_out(lv_lineno, lv_seg_name, lv_string);
lv_lineno := lv_lineno + 1;
lv_string:='STORAGE ' || '(INITIAL '||lv_initial_extent||' NEXT '||
           lv_next_extent||&&cr||' MINEXTENTS '||lv_min_extents||
           ' MAXEXTENTS '|| lv_max_extents||')';
write_out(lv_lineno, lv_seg_name, lv_string);
lv_lineno := lv_lineno + 1;
lv_string:=
'/'||&&cr||'ALTER ROLLBACK SEGMENT '||lv_seg_name||' ONLINE;'||&&cr;
write_out(lv_lineno, lv_seg_name, lv_string);
END IF;
END LOOP;
  CLOSE rb_cursor;
```

```
END;
/
COLUMN dbname NEW_VALUE db NOPRINT
SELECT name dbname FROM v$database;
SPOOL rep_out\&db\crt_rbks.sql
SET HEADING OFF
COLUMN text FORMAT a80 WORD_WRAP
SELECT text FROM  rb_temp ORDER BY rb_name, lineno;
SPOOL OFF;
DROP TABLE rb_temp;
SET VERIFY ON FEEDBACK ON TERMOUT ON PAGESIZE 22 LINES 80 HEADING ON
CLEAR COLUMNS
```

<C>SQL Script to report on thread status (Bonus Script):

```
rem
rem FUNCTION: Provide data on Redo Log Threads
rem
rem name:     thread.sql
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\theads
SELECT * FROM sys.v_$thread
ORDER BY thread#;
SPOOL OFF
PAUSE Press enter to continue
SET LINES 80 PAGES 22
TTITLE OFF
CLEAR COLUMNS
```

<C>SQL Script to generate 132 column headers:

```
rem
rem TITLE132.SQL
rem
rem FUNCTION:    This SQL*Plus script builds a standard report heading
rem              heading for database reports that are 132 columns
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
```

```
rem
DEFINE company = " " /* put  your company name here */
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
rem
rem
SET HEADING OFF TERMOUT OFF
rem
SELECT TO_CHAR(SYSDATE,'MM/DD/YY') TODAY,
      TO_CHAR(SYSDATE,'HH:MI AM') TIME,
      name||' database' DATABASE,
      RTRIM(name) passout
FROM   sys.v_$database;
rem
SET HEADING ON TERMOUT ON
SET NEWPAGE 0
DEFINE db = '&dbname'
```

<C>SQL Script to generate 80 column headers:

```
rem
rem TITLE80.SQL
rem
rem FUNCTION:    This SQL*Plus script builds a standard report heading
rem              heading for database reports that are 80 columns
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
rem
DEFINE company = " " /* Put yor company name here */
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
rem
SET HEADING OFF
SET PAGESIZE 0
rem
SET TERMOUT OFF
SELECT TO_CHAR(SYSDATE,'MM/DD/YY') TODAY,
      TO_CHAR(SYSDATE,'HH:MI AM') TIME,
      name||' database' DATABASE,
      rtrim(name) passout
FROM   v$database;
rem
SET TERMOUT ON
```

```
SET HEADING ON
SET PAGESIZE 58
SET NEWPAGE 0
DEFINE db = '&dbname'
```

<C>The following script creates all of the tables used by the scripts in the appendix.

```
DROP TABLE dba_temp;
CREATE TABLE dba_temp (
name        VARCHAR2(64),
value       NUMBER,
rep_order   NUMBER);

DROP TABLE temp_size_table;
CREATE TABLE temp_size_table (
table_name  VARCHAR2(64),
blocks      NUMBER);

DROP TABLE hit_ratios;
CREATE TABLE hit_ratios (
check_date        DATE NOT NULL,
check_hour        NUMBER NOT NULL,
db_block_gets     NUMBER,
consistent        NUMBER,
phy_reads         NUMBER,
hitratio          NUMBER,
period_hit_ratio  NUMBER,
period_usage      NUMBER,
users             NUMBER);

CREATE UNIQUE INDEX hr_index ON hit_ratios (
check_date,
check_hour);

REM  You must have direct select grants on the undelying tables
REM  for these views to be generated.

CREATE OR REPLACE VIEW free_space (
      tablespace,
      file_id,
      pieces,
      free_bytes,
      free_blocks,
      largest_bytes,
      largest_blks,
      fsfi)
AS
SELECT
      tablespace_name,
      file_id,
      count(*),
      sum(bytes),
      sum(blocks),
      max(bytes),
```

```
      max(blocks),
      sqrt(max(blocks)/sum(blocks))*(100/sqrt(sqrt(count(blocks))))
FROM
      sys.dba_free_space
GROUP BY
      tablespace_name,
      file_id;

REM
REM FUNCTION: create views required for rbk1 and rbk2 reports.
REM
rem exit
CREATE OR REPLACE VIEW rollback1 AS
SELECT
      d.segment_name,extents,
      optsize,shrinks,
      aveshrink,aveactive,
      d.status
FROM
      v$rollname n,
      v$rollstat s,
      dba_rollback_segs d
WHERE
      d.segment_id=n.usn(+)
      AND d.segment_id=s.usn(+)
;

CREATE OR REPLACE VIEW rollback2 AS
SELECT
      d.segment_name,
      extents,
      xacts,
      hwmsize,
      rssize,
      waits,
      wraps,
      extends,
      d.status
FROM
      v$rollname n,
      v$rollstat s,
      dba_rollback_segs d
WHERE
      d.segment_id=n.usn(+)
      and d.segment_id=s.usn(+);

rem FUNCTION: Creates summary of v_$sqlarea and dba_users for use in
rem           sqlmem.sql and sqlsummary.sql reports
rem
rem
CREATE OR REPLACE VIEW sql_summary AS
SELECT
      username,
      sharable_mem,
      persistent_mem,
      runtime_mem
FROM
      sys.v_$sqlarea a,
```

```
      dba_users b
WHERE
      a.parsing_user_id = b.user_id;

rem
rem trans_per_rollback view gives a quick look at who is doing what to
rollbacks
rem
CREATE OR REPLACE VIEW trans_per_rollback
(name,sid,pid,transaction,terminal) AS
SELECT
      r.name,
      l.Sid,
      p.spid,
      NVL(p.username, 'no transaction'),
      p.terminal
FROM
      v$lock l,
      v$process p,
      v$rollname r
WHERE
      l.Sid = p.pid (+)
AND
      TRUNC(l.id1(+) / 65536) = r.usn and l.type(+) = 'TX'
      and l.lmode(+) = 6;

rem
rem proc_count view
rem provides line count data for procedures
rem
CREATE OR REPLACE VIEW proc_count AS
SELECT
      owner,
      name,
      type,
      count(*) lines
FROM
      dba_source
GROUP BY
      owner,name,type
/
PAUSE Finished with table and view creates
```

<C>Grant Script to give direct grants (even to SYSTEM) to build views, functions and procedures in this appendix:

```
Rem In order for the views used for monitoring to be created
Rem these direct grants have to be made to the user who will be
Rem doing the monitoring. It is suggested that the user
Rem also be granted the DBA role or the MONITORER role
Rem These grants must be made from the SYS user.
Rem
GRANT SELECT ON DBA_FREE_SPACE TO &&MONITORING_USER;
GRANT SELECT ON V_$ROLLSTAT TO &&MONITORING_USER;
GRANT SELECT ON V_$ROLLNAME TO &&MONITORING_USER;
```

```
GRANT SELECT ON V_$SGASTAT TO &&MONITORING_USER;
GRANT SELECT ON V_$SQLAREA TO &&MONITORING_USER;
GRANT SELECT ON V_$LOCK TO &&MONITORING_USER;
GRANT SELECT ON DBA_USERS TO &&MONITORING_USER;
GRANT SELECT ON V_$PROCESS TO &&MONITORING_USER;
GRANT SELECT ON DBA_SOURCE TO &&MONITORING_USER;
GRANT SELECT ON DBA_ROLLBACK_SEGS TO &&MONITORING_USER;
GRANT SELECT ON V_$ROWCACHE TO &&MONITORING_USER;
GRANT SELECT ON V_$SYSSTAT TO &&MONITORING_USER;
GRANT SELECT ON V_$WAITSTAT TO &&MONITORING_USER;
GRANT SELECT ON V_$INSTANCE TO &&MONITORING_USER;
GRANT SELECT ON V_$LIBRARYCACHE TO &&MONITORING_USER;
GRANT SELECT ON V_$SGA TO &&MONITORING_USER;
GRANT SELECT ON V_$LATCHNAME TO &&MONITORING_USER;
GRANT SELECT ON V_$LATCH TO &&MONITORING_USER;
GRANT SELECT ON dba_tablespaces TO &&MONITORING_USER;
GRANT SELECT ON dba_indexes TO &&MONITORING_USER;
GRANT SELECT ON dba_extents TO &&MONITORING_USER;
GRANT SELECT ON dba_objects TO &&MONITORING_USER;
GRANT SELECT On DBA_DATA_FILES TO &&MONITORING_USER;
GRANT SELECT ON DBA_TABLES TO &&MONITORING_USER;
GRANT SELECT ON DBA_TAB_COLUMNS TO &&MONITORING_USER;
GRANT SELECT ON V_$FIXED_VIEW_DEFINITION TO &&MONITORING_USER;
GRANT SELECT ON V_$SESSION TO &&MONITORING_USER;
GRANT SELECT ON V_$SESSTAT TO &&MONITORING_USER;
GRANT SELECT ON V_$DATABASE TO &&MONITORING_USER;
rem on some releases the following may be required
CREATE PUBLIC SYNONYM V$DATABASE FOR SYS.V_$DATABASE;
PAUSE Finished with grants - press enter to continue
```