PRACTICAS TEMA 4.
AJUSTE Y MONITORIZACIÓN DE LA INSTANCIA.

4.1. Revisar los eventos del sistema y comprobar los más significativos. Comprobar el evento “latch free” por sesiones. Comprobar sesiones esperando por el evento “db file sequential read”. Comprueba el parámetro TIMED_STATISTICS.

4.2. Revisa las estadísticas del sistema más significativas. Comprueba el tamaño medio de la PGA de cada sesión. Revisa las lecturas lógicas y físicas y calcula el ratio de E/S. Comparar el uso de CPU para “SQL del sistema” (acceso al DD) sobre el total.

4.3. Ver el estado de ocupación de las partes más significativas de la Shared Pool.

4.4. Comprobar la contención en latches de la Shared Pool y Library Cache.

4.5. Comprobar el pinhitratio de la Library Caché, así como los reloads. Verificar el espacio libre de la Shared Pool, y el valor de open_cursors.

4.6. Ver el ratio de la Row Cache.

4.7. Comprobar si el sistema recomienda ampliar la SharedPool.

4.8. Detectar sentencias similares que usan literales. Verificar los parámetros cursor_sharing y session_cached_cursors. Asignar cursor_sharing=similar.

4.9. Instalar el paquete DBMS_SHARED_POOL. Comprobar paquetes que se pueden “fijar” en la SharedPool y hacerlo.

4.10. Comprobar sentencias que ocupan mucha memoria (>=10% de SharedPool).

4.11. Calcular el ratio de eficiencia de la Caché de Datos. Comprobar el parámetro db_cache_advice. Consultar si Oracle recomienda incrementar la Caché de Datos. Comprobar si hay contención en el latch “cache buffers lru chain”. Ver si hay esperas del tipo “write complete waits” o “free buffer waits”.

4.12. Comprobar el tamaño de la Cache de Redo. Ver si hay contención en los latches de redo. Verificar la estadística “redo log space requests”.

4.13. Comprobar los segmentos de rollback ONLINE. Verificar las extensiones que tiene cada uno, así como el espacio total y libre del tablespace que los contiene. Ver si hay contención en segmentos de rollback.

4.14. Instalar la utilidad STATSPACK. Crear snapshots y generar informe. Analizar el informe.

Recursos.

- Máquina: cursos.atica.um.es
- Usuario unix: curso51 a curso70
- Conexión máquina unix: ssh con Secure Shell
- Bases de datos: CURSO51 a CURSO70
- Conexión a bd: sqlplus desde sesión unix, usuario “`/ as sysdba`”

4.1. Revisar los eventos del sistema y comprobar los más significativos. Comprobar el evento “latch free” por sesiones. Comprobar sesiones esperando por el evento “db file sequential read”. Comprueba el parámetro TIMED_STATISTICS.

Solución:

```
SQL> set linesize 132
SQL> select rpad(event,30),total_waits,total_timeouts,time_waited,average_wait
     from v$system_event;
RPAD(EVENT,30)                 TOTAL_WAITS TOTAL_TIMEOUTS TIME_WAITED AVERAGE_WAIT
------------------------------ ----------- -------------- ----------- ------------
latch free                               4              0           0            0
...
buffer busy waits                        1              0           4            4
...
db file sequential read              69998              0        3096            0
db file scattered read              234091              0        1496            0
...
db file parallel write               11484           5668           6            0
...

SQL> select sid,total_waits,total_timeouts,time_waited,average_wait
      from v$session_event
      where event='latch free';
       SID TOTAL_WAITS TOTAL_TIMEOUTS TIME_WAITED AVERAGE_WAIT
---------- ----------- -------------- ----------- ------------
         2           4              0           0            0

SQL> select event,sid,P1TEXT,P1,P2TEXT,P2,P3TEXT,P3,WAIT_TIME,SECONDS_IN_WAIT,STATE
      from v$session_wait
      where event like 'db file %';
no rows selected

SQL> show parameter timed_statistics
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
timed_statistics                     boolean     TRUE
```

Consulta la documentación para ver en qué consiste el evento “db file scattered read”. Lo puedes ver en “http://cursos.atica.um.es/oradoc92/server.920/a96536/apa5.htm#971547”.

4.2. Revisa las estadísticas del sistema más significativas. Comprueba el tamaño medio de la PGA de cada sesión. Revisa las lecturas lógicas y físicas y calcula el ratio de E/S. Comparar el uso de CPU para “SQL del sistema” (acceso al DD) sobre el total.

Solución:

Las estadísticas del sistema se pueden consultar en V$SYSSTAT.

```
SQL> select name,value from v$sysstat;
NAME                                                                  VALUE
---------------------------------------------------------------- ----------
logons cumulative                                                        84
logons current                                                            7
opened cursors cumulative                                             33071
opened cursors current                                                    1
user commits                                                            281
user rollbacks                                                             1
user calls                                                              5500
recursive calls                                                       459057
recursive cpu usage                                                     1155
...
CPU used by this session                                                3913
...
db block gets                                                          81265
consistent gets                                                      2670334
physical reads                                                       1913714
...
session pga memory                                                   6698384
session pga memory max                                               6903476
...
```

Las estadísticas sobre el uso de la PGA nos permiten hacer estimaciones sobre el consumo de memoria (RAM) de cada sesión:

```
SQL> select name,value from v$sysstat where name like 'session pga memory%';
NAME                                                                   VALUE
---------------------------------------------------------------- ----------
session pga memory                                                   6698384
session pga memory max                                               6903476
```

Las estadísticas de E/S nos permiten calcular el ratio de eficiencia de la caché de datos. Podemos ver el porcentaje de lecturas a disco sobre las lecturas lógicas:

```
SQL> r
  1  select 100*f1/(r1+r2)
  2  from (select value f1 from v$sysstat where name='physical reads'),
  3       (select value r1 from v$sysstat where name='consistent gets'),
  4*      (select value r2 from v$sysstat where name='db block gets')
100*F1/(R1+R2)
--------------
    69.4616461
```

También podemos calcular el tiempo de CPU que se lleva el “SQL del sistema” (acceso al DD) respecto al tiempo total de CPU:

```
SQL> select 100*cpu1/cpu2
  2  from (select value cpu1 from v$sysstat where name='recursive cpu usage'),
  3       (select value cpu2 from v$sysstat where name='CPU used by this session');
100*CPU1/CPU2
-------------
   29.5019157
```

4.3. Ver el estado de ocupación de las partes más significativas de la Shared Pool.

Solución:

La vista V$SGASTAT nos muestra el estado de las distintas partes de la Shared Pool.

```
SQL> r
  1  select * from v$sgastat
  2* where name in ('free memory','library cache','sql area','dictionary cache')
POOL        NAME                            BYTES
----------- -------------------------- ----------
shared pool sql area                      3755132
shared pool free memory                    680920
shared pool library cache                 2765552
shared pool dictionary cache              1610880
```

4.4. Comprobar la contención en latches de la Shared Pool y Library Cache.

Solución:

Los latches son microbloqueos necesarios para la gestión de los recursos compartidos del sistema. En la vista V$LATCH tenemos información sobre el uso de latches. La columna MISSES indica los fallos que hay al intentar coger un latch; y SLEEPS indica el nº de veces que el proceso que intenta coger el latch pasa al estado "dormido".

```
SQL> r
  1  select rpad(name,30),gets,misses,sleeps
  2  from v$latch
  3  where name like '%library cache%' or
  4*       name like '%shared pool%'
RPAD(NAME,30)                        GETS     MISSES     SLEEPS
------------------------------ ---------- ---------- ----------
shared pool                      41147855          0          0
library cache                      537666          0          0
library cache pin                  294556          0          0
library cache pin allocation       185739          0          0
library cache load lock              4850          0          0
```

4.5. Comprobar el pinhitratio de la Library Caché, así como los reloads. Verificar el espacio libre de la Shared Pool, y el valor de open_cursors.

Solución:

Podemos ver las estadísticas de la Library Caché en la vista V$LIBRARYCACHE. Los espacios correspondientes a SQL y PLSQL son:

```
SQL> r
  1  select namespace,pinhitratio,reloads from v$librarycache
  2* where namespace in ('SQL AREA','TABLE/PROCEDURE','BODY','TRIGGER')
NAMESPACE       PINHITRATIO    RELOADS
--------------- ----------- ----------
SQL AREA         .961152089        450
TABLE/PROCEDURE  .864664768        519
BODY             .420168067          3
TRIGGER          .922190202          1
```

Se recomienda un pinhitratio cercano a 1, y reloads casi cero. Llama la atención espacialmente el nº de reloads en "sql area" y "table/procedure"; así como el bajo ratio en "body". Podemos plantearnos subir algo el tamaño de la SharedPool. Primero habría que ver si hay espacio libre en la SharedPool, de forma continuada; y también revisar open_cursors.

```
SQL> r
1  select SharedPool, LibraryCache, Libre
2  from (select sum(bytes) SharedPool from v$sgastat where pool='shared pool'),
3   (select bytes LibraryCache from v$sgastat where name='library cache'),
4*  (select bytes Libre from v$sgastat where name='free memory' and pool='shared pool')
SHAREDPOOL LIBRARYCACHE      LIBRE
---------- ------------ ----------
  20971520      2793664     764464

SQL> show parameter open_cursors
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
open_cursors                         integer     50
```

4.6. Ver el ratio de la Row Cache.

Solución:

Podemos ver los ratios de la Row Caché en V$ROWCACHE, y también calcular un “ratio medio”..

```
SQL> r
  1  SELECT rpad(parameter,25), sum(gets), sum(getmisses)
  2       , 100*sum(gets - getmisses) / sum(gets) pct_succ_gets
  3       , sum(modifications) updates
  4*   FROM V$ROWCACHE WHERE gets > 0 GROUP BY parameter
RPAD(PARAMETER,25)         SUM(GETS) SUM(GETMISSES) PCT_SUCC_GETS    UPDATES
------------------------- ---------- -------------- ------------- ----------
dc_constraints                   272            103    62.1323529        272
dc_files                          41             11    73.1707317         15
dc_global_oids                    54             15    72.2222222          0
dc_histogram_defs               4283            466    89.1197759        139
dc_object_ids                  17309            549    96.8282396        177
dc_objects                      7787           1018    86.9269295        420
dc_profiles                       27              2    92.5925926          0
dc_rollback_segments           40097              5    99.9875302          9
dc_segments                     5925            471    92.0506329        407
dc_sequences                      20             14            30         20
dc_table_scns                     25             25             0          0
dc_tablespace_quotas             134              8    94.0298507        129
dc_tablespaces                 42956             26    99.9394729         17
dc_user_grants                   936             31    96.6880342          0
dc_usernames                    2891             26    99.1006572          3
dc_users                       38381             33      99.91402         10
16 rows selected.

SQL> SELECT SUM(GETS - GETMISSES - FIXED) * 100 / SUM(GETS) "Ratio Medio"
      FROM V$ROWCACHE;
Ratio Medio
-----------
 98.1917244
```

La recomendación es que dicho ratio ser >= 85%.

4.7. Comprobar si el sistema recomienda ampliar la SharedPool.

Solución:

Las estadísticas sobre posibles redimensionamientos de la SharedPool los podemos ver en V$SHARED_POOL_ADVICE.

```
SQL> r
  1  select SHARED_POOL_SIZE_FOR_ESTIMATE SIZE_ESTIMATE,
  2  SHARED_POOL_SIZE_FACTOR SIZE_FACTOR,
  3  ESTD_LC_TIME_SAVED_FACTOR PARSE_SAVED_FACTOR
  4*  from v$shared_pool_advice
SIZE_ESTIMATE SIZE_FACTOR PARSE_SAVED_FACTOR
------------- ----------- ------------------
            4          .5              .9817
            8           1                  1
           12         1.5             1.0183
           16           2             1.0183
```

Según los datos anteriores la ganancia por subir la SharedPool de 8 a 16M sería del 1.8% (de 1 a 1.0183).

4.8. Detectar sentencias similares que usan literales. Verificar el parámetros cursor_sharing y session_cached_cursors. Asignar cursor_sharing=similar.

Solución:

Para detectar sentencias similares miraremos en V$SQLAREA:

```
SQL> r
  1   SELECT substr(sql_text,1,40) "SQL", count(*), sum(executions) "TotExecs"
  2       FROM v$sqlarea
  3      WHERE executions < 5
  4*    GROUP BY substr(sql_text,1,40) HAVING count(*) > 20 ORDER BY 2
SQL                                        COUNT(*)   TotExecs
---------------------------------------- ---------- ----------
SELECT NAME name_col_plus_show_param,DEC         21         25

SQL> select sql_text from v$sqlarea where substr(sql_text,1,40)='SELECT NAME
name_col_plus_show_param,DEC';
SQL_TEXT
--------------------------------------------------------------------------------
SELECT NAME name_col_plus_show_param,DECODE(TYPE,1,'boolean',2,'string',3,'integ
er',4,'file',6,'big integer','unknown') TYPE,VALUE value_col_plus_show_param FRO
M V$PARAMETER WHERE UPPER(NAME) LIKE UPPER('%db_name%') ORDER BY name_col_plus_s
how_param,ROWNUM

SELECT NAME name_col_plus_show_param,DECODE(TYPE,1,'boolean',2,'string',3,'integ
er',4,'file',6,'big integer','unknown') TYPE,VALUE value_col_plus_show_param FRO
M V$PARAMETER WHERE UPPER(NAME) LIKE UPPER('%instance%') ORDER BY name_col_plus_
show_param,ROWNUM

SELECT NAME name_col_plus_show_param,DECODE(TYPE,1,'boolean',2,'string',3,'integ
er',4,'file',6,'big integer','unknown') TYPE,VALUE value_col_plus_show_param FRO
M V$PARAMETER WHERE UPPER(NAME) LIKE UPPER('%db_domain%') ORDER BY name_col_plus
_show_param,ROWNUM
...

SQL> show parameter cursor_sharing
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
cursor_sharing                       string      EXACT

SQL> show parameter session_cached_cursors
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
session_cached_cursors               integer     0
SQL> alter system set cursor_sharing=similar;
System altered.
```

4.9. Instalar el paquete DBMS_SHARED_POOL. Comprobar paquetes que se pueden “fijar” en la SharedPool y hacerlo.

El paquete dbms_shared_pool se instala con el script $ORACLE_HOME/rdbms/admin/dbmspool.sql, bajo el usuario SYS.

Solución:

Instalar el paquete.

```
SQL> connect / as sysdba
SQL> @$ORACLE_HOME/rdbms/admin/dbmspool.sql
```

Comprobar paquetes a fijar.

```
SQL> set serveroutput on size 2000
SQL> exec dbms_shared_pool.sizes(100)
SIZE(K) KEPT   NAME
------- ------ ---------------------------------------------------------------
...
377            SYS.STANDARD                  (PACKAGE)
...
```

Fijarlos.

```
SQL> exec dbms_shared_pool.keep('SYS.STANDARD');
PL/SQL procedure successfully completed.
```

Comprobarlo:

```
SQL> exec dbms_shared_pool.sizes(0)
SIZE(K) KEPT   NAME
------- ------ ---------------------------------------------------------------
...
377     YES    SYS.STANDARD                  (PACKAGE)
...
```

4.10. Comprobar sentencias que ocupan mucha memoria (>=10% de SharedPool).

Solución:

En V$SQL tenemos información sobre las sentencias SQL en ejecución, incluyendo el consumo de memoria. Nos interesa controlar aquellas que ocupan el 10% o más de la Shared Pool.

```
SQL> show parameter shared_pool_size
NAME                     TYPE                             VALUE
------------------------ -------------------------------- ------------------------------
shared_pool_size         big integer                      12582912

SQL> r
  1  selecT substr(sql_text,1,40) "Stmt", count(*),
  2                 sum(sharable_mem)     "Mem",
  3                 sum(users_opening)    "Open",
  4                 sum(executions)       "Exec"
  5           FROM v$sql
  6          GROUP BY substr(sql_text,1,40)
  7*        HAVING sum(sharable_mem) > 1200000
Stmt
--------------------------------------------------------------------------------
  COUNT(*)        Mem       Open       Exec
---------- ---------- ---------- ----------
select bytes from dba_EXTENTS where segm
         3    1488777          0         10
```

4.11. Calcular el ratio de eficiencia de la Caché de Datos. Comprobar el parámetro db_cache_advice. Consultar si Oracle recomienda incrementar la Caché de Datos. Comprobar si hay contención en el latch “cache buffers lru chain”. Ver si hay esperas del tipo “write complete waits” o “free buffer waits”.

Solución:

Lo calcularemos a partir de los datos de lecturas físicas y lógicas que podemos ver en V$SYSSTAT.

```
SQL> r
  1    select 100*(1 - (f1 - f2 - f3)/(r1 + r2 - f2 -f3)) Ratio
  2    from (select value f1 from v$sysstat where name='physical reads'),
  3         (select value f2 from v$sysstat where name='physical reads direct'),
  4         (select value f3 from v$sysstat where name='physical reads direct (lob)'),
  5         (select value r1 from v$sysstat where name='consistent gets'),
  6*        (select value r2 from v$sysstat where name='db block gets')
     RATIO
----------
58.9109049
```

Las recomendaciones sobre incrementos de tamaño de la Caché de Datos los podemos ver en V$DB_CACHE_ADVICE, siempre que el parámetro db_cache_advice=on.

```
SQL> show parameter db_cache_advice
NAME                 TYPE                              VALUE
-------------------- --------------------------------- ------------------------------
db_cache_advice      string                            ON

SQL> set linesize 100
SQL> r
  1  select id,name,size_for_estimate,size_factor,ESTD_PHYSICAL_READ_FACTOR
  2*   from v$db_cache_advice
        ID NAME                 SIZE_FOR_ESTIMATE SIZE_FACTOR ESTD_PHYSICAL_READ_FACTOR
---------- -------------------- ----------------- ----------- -------------------------
...
         1 KEEP                                80          20
         3 DEFAULT                              4           1                         1
         3 DEFAULT                              8           2                     .0211
         3 DEFAULT                             12           3                     .0211
...
```

Como se observa, merece la pena subir la caché de 4 a 8M, pero no más.

En la vista V$LATCH puedo ver si hay contención en “cache buffers lru chain”.

```
SQL> SELECT NAME,GETS,MISSES,SLEEPS FROM V$LATCH WHERE NAME='cache buffers lru chain';
NAME                                      GETS       MISSES     SLEEPS
----------------------------------------- ---------- ---------- ----------
cache buffers lru chain                   144069          3          3
```

En la vista V$BUFFER_POOL_STATISTICS puedo ver si hay esperas del tipo “free buffer wait” o “write complete wait”.

```
SQL> SELECT ID, NAME, FREE_BUFFER_WAIT, WRITE_COMPLETE_WAIT
       FROM V$BUFFER_POOL_STATISTICS;
        ID NAME                 FREE_BUFFER_WAIT WRITE_COMPLETE_WAIT
---------- -------------------- ---------------- -------------------
         1 KEEP                                0                   0
         3 DEFAULT                             0                   0
         5 DEFAULT                             0                   0
```

4.12. Comprobar el tamaño de la Cache de Redo. Ver si hay contención en los latches de redo. Verificar la estadística "redo log space requests".

Solución:

El tamaño de la Caché de Redo lo determina el parámetro log_buffer.

```
SQL> show parameter log_buffer
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
log_buffer                           integer     1048576
```

Tendremos contención en los latches de redo si alguno de los siguientes ratios supera el 1%:

```
SQL> r
 1 SELECT substr(name,1,20) latch,
 2    100*decode(gets, 0, 0, misses/gets) mis_ratio,
 3    100*decode(immediate_gets + immediate_misses,0,0,
 4                    immediate_misses/(immediate_gets + immediate_misses)) im_mis_ratio
 5 FROM v$latch
 6 WHERE name in ('redo allocation', 'redo copy')
LATCH                 MIS_RATIO IM_MIS_RATIO
-------------------- ---------- ------------
redo copy                     0     .1179941
redo allocation               0            0
```

Finalmente, comprobemos la estadística "redo log space requests":

```
SQL> select  * from v$sysstat where name='redo log space requests';
STATISTIC# NAME                                              CLASS      VALUE
---------- ------------------------------------------------- ---------- ----------
       122 redo log space requests                           2          1
```

4.13. Comprobar los segmentos de rollback ONLINE. Verificar las extensiones que tiene cada uno, así como el espacio total y libre del tablespace que los contiene. Ver si hay contención en segmentos de rollback.

Solución:

En la vista DBA_ROLLBACK_SEGS podemos ver los segmentos de rollback, si están en línea, y a qué tablespace pertenecen. En V$ROLLSTAT (y V$ROLLNAME), podemos ver estadísticas de tamaño, peticiones, extensión dinámica, etc.

```
SQL> select segment_name, rpad(tablespace_name,10) tablespace, initial_extent,
            min_extents, max_extents,status
      from dba_rollback_segs;
SEGMENT_NAME                   TABLESPACE INITIAL_EXTENT MIN_EXTENTS MAX_EXTENTS STATUS
------------------------------ ---------- -------------- ----------- ----------- ------
SYSTEM                         SYSTEM             102400           1       32765 ONLINE
_SYSSMU1$                      UNDO_RBS           131072           2       32765 ONLINE
_SYSSMU2$                      UNDO_RBS           131072           2       32765 ONLINE
_SYSSMU3$                      UNDO_RBS           131072           2       32765 ONLINE
_SYSSMU4$                      UNDO_RBS           131072           2       32765 ONLINE

SQL> SELECT RPAD(NAME,10) NAME,EXTENTS,RSSIZE,GETS,WAITS,SHRINKS,EXTENDS,STATUS
       FROM V$ROLLNAME A, V$ROLLSTAT B
      WHERE A.USN=B.USN;
NAME          EXTENTS     RSSIZE       GETS      WAITS    SHRINKS    EXTENDS STATUS
------------- ---------- ---------- ---------- ---------- ---------- ------- --------
SYSTEM              7     456704         43          0          0          0 ONLINE
```

```
_SYSSMU1$            2     129024         33          0          0          0 ONLINE
_SYSSMU2$            2     129024         35          0          0          0 ONLINE
_SYSSMU3$            2     129024         39          0          0          0 ONLINE
_SYSSMU4$            2     129024         35          0          0          0 ONLINE
```

Podemos consultar las extensiones y espacio ocupado de cada RS, en la vista DBA_EXTENTS; y el espacio libre del tablespace que los contiene, en DBA_FREE_SPACE.

```
SQL> SELECT rpad(segment_name,10) Rsname,count(*),sum(bytes)
      FROM DBA_EXTENTS WHERE TABLESPACE_NAME='UNDO_RBS' group by segment_name;
RSNAME       COUNT(*) SUM(BYTES)
---------- ---------- ----------
_SYSSMU1$           2     129024
_SYSSMU2$           2     129024
_SYSSMU3$           2     129024
_SYSSMU4$           2     129024

SQL> select sum(bytes) from dba_free_space where tablespace_name='UNDO_RBS';
SUM(BYTES)
----------
  19333120
```

Finalmente, las estadísticas sobre contención en RS, las podemos ver en V$WAITSTAT. Si la hubiese en “undo header” tendríamos que crear más RS.

```
SQL> SELECT CLASS, COUNT FROM V$WAITSTAT
      WHERE CLASS like '%undo%';
CLASS                   COUNT
------------------ ----------
save undo block             0
save undo header            0
system undo header          0
system undo block           0
undo header                 0
undo block                  0
```

4.14. Instalar la utilidad STATSPACK. Crear snapshots y generar informe.

Solución:

Crear tablespace statspack_tsp con 100M en /u05/oradata/CURSOxy (crear dir. /u05/oradata/CURSOxy).
Crear tablespace statspack_temp con 2M en /u05/oradata/CURSOxy.
Instalar statspack con $ORACLE_HOME/rdbms/admin/spcreate.sql, indicando los datos que se piden: clave de usuario perfstat, tablespace por defecto, y tablespace temporal por defecto.

```
$ mkdir /u05/oradata/CURSOxy
$ chmod -R g+w /u05/oradata/CURSOxy

SQL> r
  1  create tablespace statpack_tsp
  2   datafile '/u05/oradata/CURSOxy/statpack_tsp01.dbf' size 100M
  3   autoextend on next 10M maxsize 200M
  4   extent management local autoallocate
  5*  segment space management auto
Tablespace created.

SQL> r
  1  create temporary tablespace statpack_temp
  2  tempfile '/u05/oradata/CURSOxy/statpack_temp01.dbf' size 2M
  3* autoextend on next 1M maxsize 10M
Tablespace created.
```

```
SQL> @$ORACLE_HOME/rdbms/admin/spcreate.sql
... Installing Required Packages
...
... Creating PERFSTAT user ...
Choose the PERFSTAT user's password.
Not specifying a password will result in the installation FAILING
Specify PERFSTAT password
Enter value for perfstat_password: perfcursoXY
...
Below are the list of online tablespaces in this database.
Decide which tablespace you wish to create the STATSPACK tables
and indexes.  This will also be the PERFSTAT user's default tablespace.

Specifying the SYSTEM tablespace will result in the installation
FAILING, as using SYSTEM for performance data is not supported.

TABLESPACE_NAME                CONTENTS
------------------------------ ---------
STATPACK_TEMP                  TEMPORARY
STATPACK_TSP                   PERMANENT
TEMP                           TEMPORARY
UNDO_RBS                       UNDO

Specify PERFSTAT user's default   tablespace
Enter value for default_tablespace: STATPACK_TSP
...
Choose the PERFSTAT user's temporary tablespace.

Specifying the SYSTEM tablespace will result in the installation
FAILING, as using SYSTEM for the temporary tablespace is not recommended.
Specify PERFSTAT user's temporary tablespace.
Enter value for temporary_tablespace: STATPACK_TEMP
...
No errors.
NOTE:
SPCPKG complete. Please check spcpkg.lis for any errors.
```

Una vez instalado STATSACK, comprobamos si ha habido errores (en los ficheros *.lis):

```
SQL> !grep -i err *.lis
spcpkg.lis:No errors.
spcpkg.lis:No errors.
spcpkg.lis:SPCPKG complete. Please check spcpkg.lis for any errors.
spctab.lis:SPCTAB complete. Please check spctab.lis for any errors.
spcusr.lis:SPCUSR complete. Please check spcusr.lis for any errors.
```

Ahora podemos utilizar el paquete STATSPACK.

Primero compruebo si está activada la recolección de estadísticas:

```
SQL> show parameter timed_statistics
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
timed_statistics                     boolean     TRUE
```

Ahora me conecto como PERFSTAT (o usuario con acceso al paquete STATSPACK) y genero el primer snapshot (baseline) con nivel 10 (i_snap_level). Después, espero al menos 5 minutos para generar otro snapshot, pues para sacar un informe necesito al menos dos:

```
SQL> connect perfstat
Enter password:
Connected.

SQL> exec STATSPACK.SNAP(i_snap_level=>10)
PL/SQL procedure successfully completed.
(... LANZAR ALGUNAS CONSULTAS, COMO LAS DOS SIGUIENTES ...)
```

```
SQL> SELECT COUNT(*) FROM DBA_OBJECTS;
  COUNT(*)
----------
      5924
SQL> SELECT COUNT(*) FROM DBA_EXTENTS;
  COUNT(*)
----------
      2647
(... ESPERAR 5 MINUTOS DESDE QUE HICIMOS EL SNAPSHOT...)
SQL> exec STATSPACK.SNAP
PL/SQL procedure successfully completed.
```

Ahora ya podemos generar el informe:

```
SQL> @$ORACLE_HOME/rdbms/admin/spreport.sql

Current Instance
~~~~~~~~~~~~~~~~

   DB Id    DB Name      Inst Num Instance
----------- ------------ -------- ------------
 1529149316 CURSOxy             1 CURSOxy



Instances in this Statspack schema
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   DB Id    Inst Num DB Name      Instance     Host
----------- -------- ------------ ------------ ------------
 1529149316        1 CURSOxy      CURSOxy      cursos.atica
                                               .um.es

Using 1529149316 for database Id
Using          1 for instance number

Completed Snapshots

                           Snap                     Snap
Instance     DB Name         Id   Snap Started    Level Comment
------------ ------------ ----- ----------------- ----- ---------------------
CURSOxy      CURSOxy          1 28 Apr 2005 11:08    10
                              2 28 Apr 2005 11:17     5



Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin snap: 1
Begin Snapshot Id specified: 1

Enter value for end_snap: 2
End   Snapshot Id specified: 2


Specify the Report Name
~~~~~~~~~~~~~~~~~~~~~~~
The default report file name is sp_1_2.  To use this name,
press <return> to continue, otherwise enter an alternative.
Enter value for report_name:

Using the refport name sp_1_2

STATSPACK report for

DB Name         DB Id    Instance     Inst Num Release     Cluster Host
```

```
------------ ----------- ------------ -------- ----------- ------- ------------
CURSOxy       1529149316 CURSOxy             1 9.2.0.1.0   NO      cursos.atica
                                                                   .um.es

            Snap Id     Snap Time      Sessions Curs/Sess Comment
            ------- ------------------ -------- --------- -------------------
Begin Snap:       1 28-Apr-05 11:08:31        7       5.3
  End Snap:       3 28-Apr-05 11:17:02        7       2.9
   Elapsed:                8.52 (mins)

Cache Sizes (end)
~~~~~~~~~~~~~~~~~
               Buffer Cache:         4M      Std Block Size:         2K
           Shared Pool Size:         8M          Log Buffer:     1,024K

Load Profile
~~~~~~~~~~~~                            Per Second       Per Transaction
                                   ---------------       ---------------
                  Redo size:              2,690.58            458,296.00
              Logical reads:                123.43             21,024.67
              Block changes:                 12.80              2,180.33
             Physical reads:                 31.89              5,432.67
            Physical writes:                 18.02              3,069.00
                 User calls:                  0.49                 82.67
                     Parses:                  2.86                488.00
                Hard parses:                  0.23                 39.67
                      Sorts:                  0.68                116.00
                     Logons:                  0.00                  0.67
                   Executes:                  3.68                626.67
               Transactions:                  0.01

  % Blocks changed per Read:   10.37    Recursive Call %:    99.26
 Rollback per transaction %:    0.00       Rows per Sort:    37.36

Instance Efficiency Percentages (Target 100%)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            Buffer Nowait %:  100.00       Redo NoWait %:   99.95
            Buffer  Hit   %:   75.14    In-memory Sort %:   98.85
            Library Hit   %:   93.34        Soft Parse %:   91.87
         Execute to Parse %:   22.13         Latch Hit %:  100.00
Parse CPU to Parse Elapsd %:   64.10     % Non-Parse CPU:   81.48

 Shared Pool Statistics        Begin   End
                               ------  ------
             Memory Usage %:   96.78   95.04
    % SQL with executions>1:   78.38   47.25
  % Memory for SQL w/exec>1:   56.49   40.74

Top 5 Timed Events
~~~~~~~~~~~~~~~~~~                                                     % Total
Event                                               Waits    Time (s) Ela Time
-------------------------------------------- ------------ ----------- --------
db file sequential read                             3,764           7    61.42
CPU time                                                            3    22.80
db file parallel read                                 106           1     7.19
db file scattered read                              1,087           1     6.25
local write wait                                      206           0      .99
          -------------------------------------------------------------

Wait Events for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> s  - second
-> cs - centisecond -     100th of a second
-> ms - millisecond -    1000th of a second
-> us - microsecond - 1000000th of a second
-> ordered by wait time desc, waits desc (idle events last)
```

```
                                                                   Avg
                                                     Total Wait   wait    Waits
Event                               Waits   Timeouts   Time (s)   (ms)     /txn
---------------------------- ------------ ---------- ---------- ------ --------
db file sequential read             3,764          0          7      2  1,254.7
db file parallel read                 106          0          1      8     35.3
db file scattered read              1,087          0          1      1    362.3
local write wait                      206          0          0      1     68.7
control file parallel write           194          0          0      1     64.7
log file sync                           4          0          0      5      1.3
db file single write                    8          0          0      1      2.7
undo segment extension                  1          1          0     10      0.3
log file switch completion              2          0          0      4      0.7
control file sequential read          301          0          0      0    100.3
db file parallel write                482        235          0      0    160.7
log file parallel write                26         25          0      0      8.7
SQL*Net break/reset to clien            8          0          0      0      2.7
SQL*Net more data to client             3          0          0      0      1.0
direct path read                       75          0          0      0     25.0
log file single write                   2          0          0      0      0.7
direct path write                      15          0          0      0      5.0
buffer busy waits                       1          0          0      0      0.3
log file sequential read                2          0          0      0      0.7
latch free                              1          0          0      0      0.3
SQL*Net message from client           167          0        481   2878     55.7
SQL*Net message to client             169          0          0      0     56.3
SQL*Net more data from clien           26          0          0      0      8.7
          -------------------------------------------------------------

Background Wait Events for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> ordered by wait time desc, waits desc (idle events last)

                                                                   Avg
                                                     Total Wait   wait    Waits
Event                               Waits   Timeouts   Time (s)   (ms)     /txn
---------------------------- ------------ ---------- ---------- ------ --------
db file sequential read             1,843          0          5      3    614.3
control file parallel write           178          0          0      1     59.3
db file scattered read                896          0          0      0    298.7
log file sync                           1          0          0     15      0.3
db file parallel write                482        235          0      0    160.7
log file parallel write                26         25          0      0      8.7
control file sequential read          101          0          0      0     33.7
rdbms ipc reply                         1          0          0      1      0.3
log file single write                   2          0          0      0      0.7
buffer busy waits                       1          0          0      0      0.3
log file sequential read                2          0          0      0      0.7
latch free                              1          0          0      0      0.3
rdbms ipc message                     802        523      1,492   1860    267.3
smon timer                              2          1        346 ######      0.7
SQL*Net message to client               7          0          0      0      2.3
SQL*Net message from client             7          0        -66  -9365      2.3
          -------------------------------------------------------------

SQL ordered by Gets for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> End Buffer Gets Threshold:   10000
-> Note that resources reported for PL/SQL includes the resources used by
   all SQL statements called within the PL/SQL code.  As individual SQL
   statements are also reported, it is possible and valid for the summed
   total % to exceed 100

                                                     CPU      Elapsd
  Buffer Gets    Executions  Gets per Exec  %Total Time (s)  Time (s) Hash Value
--------------- ------------ -------------- ------ -------- --------- ----------
```

```
         21,244            1       21,244.0   33.7     0.90      6.49 2582333063
Module: SQL*Plus
SELECT COUNT(*) FROM DBA_EXTENTS

         15,040            3        5,013.3   23.8     0.12      0.22  274405826
Module: SQL*Plus
SELECT COUNT(*) FROM DBA_OBJECTS

         10,350            2        5,175.0   16.4     0.68     10.49 2554250747
Module: SQL*Plus
SELECT COUNT(*) FROM DBA_SOURCE

          3,174        1,058            3.0    5.0     0.10      0.11 4059714361
select type#,blocks,extents,minexts,maxexts,extsize,extpct,user#
,iniexts,NVL(lists,65535),NVL(groups,65535),cachehint,hwmincr, N
VL(spare1,0) from seg$ where ts#=:1 and file#=:2 and block#=:3

            278           67            4.1    0.4     0.01      0.00 2085632044
select intcol#,nvl(pos#,0),col# from ccol$ where con#=:1

            215          215            1.0    0.3     0.01      0.01  787810128
select /*+ rule */ bucket_cnt, row_cnt, cache_cnt, null_cnt, tim
estamp#, sample_size, minimum, maximum, distcnt, lowval, hival,
density, col#, spare1, spare2, avgcln from hist_head$ where obj#
=:1 and intcol#=:2

            213          105            2.0    0.3     0.01      0.01 1966425544
select text from view$ where rowid=:1

            108            7           15.4    0.2     0.00      0.01 1819073277
select owner#,name,namespace,remoteowner,linkname,p_timestamp,p_
obj#, d_owner#, nvl(property,0),subname from dependency$,obj$ wh
ere d_obj#=:1 and p_obj#=obj#(+) order by order#

             81            7           11.6    0.1     0.00      0.03 4049165760
select order#,columns,types from access$ where d_obj#=:1

             78           26            3.0    0.1     0.00      0.00  189272129
select o.owner#,o.name,o.namespace,o.remoteowner,o.linkname,o.su
bname,o.dataobj#,o.flags from obj$ o where o.obj#=:1

             72           20            3.6    0.1     0.01      0.04 2591785020
select obj#,type#,ctime,mtime,stime,status,dataobj#,flags,oid$,
spare1, spare2 from obj$ where owner#=:1 and name=:2 and namespa
ce=:3 and(remoteowner=:4 or remoteowner is null and :4 is null)a
nd(linkname=:5 or linkname is null and :5 is null)and(subname=:6
 or subname is null and :6 is null)

             48           14            3.4    0.1     0.01      0.01 2385919346
select name,intcol#,segcol#,type#,length,nvl(precision#,0),decod

SQL ordered by Gets for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> End Buffer Gets Threshold:   10000
-> Note that resources reported for PL/SQL includes the resources used by
   all SQL statements called within the PL/SQL code.  As individual SQL
   statements are also reported, it is possible and valid for the summed
   total % to exceed 100

                                                     CPU      Elapsd
  Buffer Gets    Executions  Gets per Exec  %Total Time (s)  Time (s) Hash Value
--------------- ------------ -------------- ------ -------- --------- ----------
e(type#,2,nvl(scale,-127/*MAXSB1MINAL*/),178,scale,179,scale,180
,scale,181,scale,182,scale,183,scale,231,scale,0),null$,fixedsto
rage,nvl(deflength,0),default$,rowid,col#,property, nvl(charseti
d,0),nvl(charsetform,0),spare1,spare2,nvl(spare3,0) from col$ wh
```

```
             29           10            2.9    0.0     0.00      0.01  931956286
select grantee#,privilege#,nvl(col#,0),max(mod(nvl(option$,0),2)
)from objauth$ where obj#=:1 group by grantee#,privilege#,nvl(co
l#,0) order by grantee#

             27            9            3.0    0.0     0.00      0.00 1705880752
select file# from file$ where ts#=:1

             20           10            2.0    0.0     0.00      0.01 1453445442
select col#, grantee#, privilege#,max(mod(nvl(option$,0),2)) fro
m objauth$ where obj#=:1 and col# is not null group by privilege
#, col#, grantee# order by col#, grantee#

          -------------------------------------------------------------

SQL ordered by Reads for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> End Disk Reads Threshold:    1000

                                                     CPU      Elapsd
 Physical Reads  Executions  Reads per Exec %Total Time (s)  Time (s) Hash Value
--------------- ------------ -------------- ------ -------- --------- ----------
          7,698            1        7,698.0   47.2     0.90      6.49 2582333063
Module: SQL*Plus
SELECT COUNT(*) FROM DBA_EXTENTS

          7,274            2        3,637.0   44.6     0.68     10.49 2554250747
Module: SQL*Plus
SELECT COUNT(*) FROM DBA_SOURCE

            626            3          208.7    3.8     0.12      0.22  274405826
Module: SQL*Plus
SELECT COUNT(*) FROM DBA_OBJECTS

            146        1,058            0.1    0.9     0.10      0.11 4059714361
select type#,blocks,extents,minexts,maxexts,extsize,extpct,user#
,iniexts,NVL(lists,65535),NVL(groups,65535),cachehint,hwmincr, N
VL(spare1,0) from seg$ where ts#=:1 and file#=:2 and block#=:3

             10            7            1.4    0.1     0.00      0.03 4049165760
select order#,columns,types from access$ where d_obj#=:1

              8            7            1.1    0.0     0.00      0.01 1819073277
select owner#,name,namespace,remoteowner,linkname,p_timestamp,p_
obj#, d_owner#, nvl(property,0),subname from dependency$,obj$ wh
ere d_obj#=:1 and p_obj#=obj#(+) order by order#

              7           14            0.5    0.0     0.01      0.01 2385919346
select name,intcol#,segcol#,type#,length,nvl(precision#,0),decod
e(type#,2,nvl(scale,-127/*MAXSB1MINAL*/),178,scale,179,scale,180
,scale,181,scale,182,scale,183,scale,231,scale,0),null$,fixedsto
rage,nvl(deflength,0),default$,rowid,col#,property, nvl(charseti
d,0),nvl(charsetform,0),spare1,spare2,nvl(spare3,0) from col$ wh

              7           20            0.4    0.0     0.01      0.04 2591785020
select obj#,type#,ctime,mtime,stime,status,dataobj#,flags,oid$,
spare1, spare2 from obj$ where owner#=:1 and name=:2 and namespa
ce=:3 and(remoteowner=:4 or remoteowner is null and :4 is null)a
nd(linkname=:5 or linkname is null and :5 is null)and(subname=:6
 or subname is null and :6 is null)

              3           10            0.3    0.0     0.00      0.01  931956286
select grantee#,privilege#,nvl(col#,0),max(mod(nvl(option$,0),2)
)from objauth$ where obj#=:1 group by grantee#,privilege#,nvl(co
l#,0) order by grantee#
```

```
              3           10            0.3    0.0     0.00      0.01 1453445442
select col#, grantee#, privilege#,max(mod(nvl(option$,0),2)) fro
m objauth$ where obj#=:1 and col# is not null group by privilege
#, col#, grantee# order by col#, grantee#

              2          105            0.0    0.0     0.01      0.01 1966425544
select text from view$ where rowid=:1

              1           67            0.0    0.0     0.01      0.00 2085632044
select intcol#,nvl(pos#,0),col# from ccol$ where con#=:1

SQL ordered by Reads for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> End Disk Reads Threshold:    1000

                                                     CPU      Elapsd
 Physical Reads  Executions  Reads per Exec %Total Time (s)  Time (s) Hash Value
--------------- ------------ -------------- ------ -------- --------- ----------

              0           26            0.0    0.0     0.00      0.00  189272129
select o.owner#,o.name,o.namespace,o.remoteowner,o.linkname,o.su
bname,o.dataobj#,o.flags from obj$ o where o.obj#=:1

              0          215            0.0    0.0     0.01      0.01  787810128
select /*+ rule */ bucket_cnt, row_cnt, cache_cnt, null_cnt, tim
estamp#, sample_size, minimum, maximum, distcnt, lowval, hival,
density, col#, spare1, spare2, avgcln from hist_head$ where obj#
=:1 and intcol#=:2

              0            9            0.0    0.0     0.00      0.00 1705880752
select file# from file$ where ts#=:1

          -------------------------------------------------------------

SQL ordered by Executions for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> End Executions Threshold:     100

                                                CPU per    Elap per
 Executions   Rows Processed   Rows per Exec    Exec (s)   Exec (s)  Hash Value
------------ --------------- ---------------- ----------- ---------- ----------
       1,058           1,058              1.0       0.00        0.00 4059714361
select type#,blocks,extents,minexts,maxexts,extsize,extpct,user#
,iniexts,NVL(lists,65535),NVL(groups,65535),cachehint,hwmincr, N
VL(spare1,0) from seg$ where ts#=:1 and file#=:2 and block#=:3

         215               0              0.0       0.00        0.00  787810128
select /*+ rule */ bucket_cnt, row_cnt, cache_cnt, null_cnt, tim
estamp#, sample_size, minimum, maximum, distcnt, lowval, hival,
density, col#, spare1, spare2, avgcln from hist_head$ where obj#
=:1 and intcol#=:2

         105             105              1.0       0.00        0.00 1966425544
select text from view$ where rowid=:1

          67              72              1.1       0.00        0.00 2085632044
select intcol#,nvl(pos#,0),col# from ccol$ where con#=:1

          26              26              1.0       0.00        0.00  189272129
select o.owner#,o.name,o.namespace,o.remoteowner,o.linkname,o.su
bname,o.dataobj#,o.flags from obj$ o where o.obj#=:1

          20              12              0.6       0.00        0.00 2591785020
select obj#,type#,ctime,mtime,stime,status,dataobj#,flags,oid$,
spare1, spare2 from obj$ where owner#=:1 and name=:2 and namespa
ce=:3 and(remoteowner=:4 or remoteowner is null and :4 is null)a
```

```
nd(linkname=:5 or linkname is null and :5 is null)and(subname=:6
 or subname is null and :6 is null)

          14             258             18.4       0.00        0.00 2385919346
select name,intcol#,segcol#,type#,length,nvl(precision#,0),decod
e(type#,2,nvl(scale,-127/*MAXSB1MINAL*/),178,scale,179,scale,180
,scale,181,scale,182,scale,183,scale,231,scale,0),null$,fixedsto
rage,nvl(deflength,0),default$,rowid,col#,property, nvl(charseti
d,0),nvl(charsetform,0),spare1,spare2,nvl(spare3,0) from col$ wh

          10               9              0.9       0.00        0.00  931956286
select grantee#,privilege#,nvl(col#,0),max(mod(nvl(option$,0),2)
)from objauth$ where obj#=:1 group by grantee#,privilege#,nvl(co
l#,0) order by grantee#

          10               0              0.0       0.00        0.00 1453445442
select col#, grantee#, privilege#,max(mod(nvl(option$,0),2)) fro
m objauth$ where obj#=:1 and col# is not null group by privilege
#, col#, grantee# order by col#, grantee#

           9               9              1.0       0.00        0.00 1705880752
select file# from file$ where ts#=:1

           7              34              4.9       0.00        0.00 1819073277
select owner#,name,namespace,remoteowner,linkname,p_timestamp,p_
obj#, d_owner#, nvl(property,0),subname from dependency$,obj$ wh
ere d_obj#=:1 and p_obj#=obj#(+) order by order#

           7              30              4.3       0.00        0.00 4049165760

SQL ordered by Executions for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> End Executions Threshold:     100

                                                CPU per    Elap per
 Executions   Rows Processed   Rows per Exec    Exec (s)   Exec (s)  Hash Value
------------ --------------- ---------------- ----------- ---------- ----------
select order#,columns,types from access$ where d_obj#=:1

           3               3              1.0       0.04        0.07  274405826
Module: SQL*Plus
SELECT COUNT(*) FROM DBA_OBJECTS

           2               0              0.0       0.34        5.24 2554250747
Module: SQL*Plus
SELECT COUNT(*) FROM DBA_SOURCE

           1               1              1.0       0.90        6.49 2582333063
Module: SQL*Plus
SELECT COUNT(*) FROM DBA_EXTENTS

          -------------------------------------------------------------

SQL ordered by Parse Calls for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> End Parse Calls Threshold:      1000

                           % Total
 Parse Calls  Executions   Parses  Hash Value
------------ ------------ -------- ----------
       1,058        1,058    72.27 4059714361
select type#,blocks,extents,minexts,maxexts,extsize,extpct,user#
,iniexts,NVL(lists,65535),NVL(groups,65535),cachehint,hwmincr, N
VL(spare1,0) from seg$ where ts#=:1 and file#=:2 and block#=:3

         105          105     7.17 1966425544
select text from view$ where rowid=:1
```

```
          10          26     0.68  189272129
select o.owner#,o.name,o.namespace,o.remoteowner,o.linkname,o.su
bname,o.dataobj#,o.flags from obj$ o where o.obj#=:1

          10         215     0.68  787810128
select /*+ rule */ bucket_cnt, row_cnt, cache_cnt, null_cnt, tim
estamp#, sample_size, minimum, maximum, distcnt, lowval, hival,
density, col#, spare1, spare2, avgcln from hist_head$ where obj#
=:1 and intcol#=:2

          10          10     0.68  931956286
select grantee#,privilege#,nvl(col#,0),max(mod(nvl(option$,0),2)
)from objauth$ where obj#=:1 group by grantee#,privilege#,nvl(co
l#,0) order by grantee#

          10          10     0.68 1453445442
select col#, grantee#, privilege#,max(mod(nvl(option$,0),2)) fro
m objauth$ where obj#=:1 and col# is not null group by privilege
#, col#, grantee# order by col#, grantee#

           9           9     0.61 1705880752
select file# from file$ where ts#=:1

           9          14     0.61 2385919346
select name,intcol#,segcol#,type#,length,nvl(precision#,0),decod
e(type#,2,nvl(scale,-127/*MAXSB1MINAL*/),178,scale,179,scale,180
,scale,181,scale,182,scale,183,scale,231,scale,0),null$,fixedsto
rage,nvl(deflength,0),default$,rowid,col#,property, nvl(charseti
d,0),nvl(charsetform,0),spare1,spare2,nvl(spare3,0) from col$ wh

           7           7     0.48 1819073277
select owner#,name,namespace,remoteowner,linkname,p_timestamp,p_
obj#, d_owner#, nvl(property,0),subname from dependency$,obj$ wh
ere d_obj#=:1 and p_obj#=obj#(+) order by order#

           7           7     0.48 4049165760
select order#,columns,types from access$ where d_obj#=:1

           6          20     0.41 2591785020
select obj#,type#,ctime,mtime,stime,status,dataobj#,flags,oid$,
spare1, spare2 from obj$ where owner#=:1 and name=:2 and namespa
ce=:3 and(remoteowner=:4 or remoteowner is null and :4 is null)a
nd(linkname=:5 or linkname is null and :5 is null)and(subname=:6
 or subname is null and :6 is null)

           3           3     0.20  274405826

SQL ordered by Parse Calls for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> End Parse Calls Threshold:      1000

                           % Total
 Parse Calls  Executions   Parses  Hash Value
------------ ----------- -------- ----------
Module: SQL*Plus
SELECT COUNT(*) FROM DBA_OBJECTS

           2           2     0.14 2554250747
Module: SQL*Plus
SELECT COUNT(*) FROM DBA_SOURCE

           1           1     0.07 2582333063
Module: SQL*Plus
SELECT COUNT(*) FROM DBA_EXTENTS
```

```
           0           67    0.00 2085632044
select intcol#,nvl(pos#,0),col# from ccol$ where con#=:1

          -------------------------------------------------------------

Instance Activity Stats for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2

Statistic                                      Total     per Second    per Trans
--------------------------------- ------------------ -------------- ------------
CPU used by this session                         270            0.5         90.0
CPU used when call started                       270            0.5         90.0
CR blocks created                                  0            0.0          0.0
DBWR buffers scanned                           4,367            8.6      1,455.7
DBWR checkpoint buffers written                  468            0.9        156.0
DBWR checkpoints                                   1            0.0          0.3
DBWR free buffers found                        3,749            7.3      1,249.7
DBWR lru scans                                    18            0.0          6.0
DBWR make free requests                           18            0.0          6.0
DBWR revisited being-written buff                  0            0.0          0.0
DBWR summed scan depth                         4,367            8.6      1,455.7
DBWR transaction table writes                      4            0.0          1.3
DBWR undo block writes                           337            0.7        112.3
SQL*Net roundtrips to/from client                160            0.3         53.3
active txn count during cleanout                 244            0.5         81.3
background checkpoints completed                   1            0.0          0.3
background checkpoints started                     1            0.0          0.3
background timeouts                              498            1.0        166.0
buffer is not pinned count                    44,208           86.5     14,736.0
buffer is pinned count                        85,640          167.6     28,546.7
bytes received via SQL*Net from c            138,328          270.7     46,109.3
bytes sent via SQL*Net to client              69,844          136.7     23,281.3
calls to get snapshot scn: kcmgss              3,438            6.7      1,146.0
calls to kcmgas                                1,108            2.2        369.3
calls to kcmgcs                                  110            0.2         36.7
change write time                                  9            0.0          3.0
cleanout - number of ktugct calls              1,298            2.5        432.7
cleanouts and rollbacks - consist                  0            0.0          0.0
cleanouts only - consistent read               1,050            2.1        350.0
cluster key scan block gets                    7,673           15.0      2,557.7
cluster key scans                              5,441           10.7      1,813.7
commit cleanout failures: buffer                   0            0.0          0.0
commit cleanout failures: callbac                 11            0.0          3.7
commit cleanouts                                 443            0.9        147.7
commit cleanouts successfully com                432            0.9        144.0
commit txn count during cleanout               1,159            2.3        386.3
consistent changes                               206            0.4         68.7
consistent gets                               58,289          114.1     19,429.7
consistent gets - examination                 12,770           25.0      4,256.7
current blocks converted for CR                    0            0.0          0.0
cursor authentications                            26            0.1          8.7
data blocks consistent reads - un                  0            0.0          0.0
db block changes                               6,541           12.8      2,180.3
db block gets                                  4,785            9.4      1,595.0
deferred (CURRENT) block cleanout                 56            0.1         18.7
dirty buffers inspected                          684            1.3        228.0
enqueue conversions                               58            0.1         19.3
enqueue releases                               1,922            3.8        640.7
enqueue requests                               1,922            3.8        640.7
execute count                                  1,880            3.7        626.7
free buffer inspected                            685            1.3        228.3
free buffer requested                         16,231           31.8      5,410.3
hot buffers moved to head of LRU                 393            0.8        131.0
immediate (CR) block cleanout app              1,050            2.1        350.0
immediate (CURRENT) block cleanou                257            0.5         85.7
index fast full scans (full)                       4            0.0          1.3
```

```
Instance Activity Stats for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2

Statistic                                      Total     per Second    per Trans
--------------------------------- ------------------ -------------- ------------
index fetch by key                            11,672           22.8      3,890.7
index scans kdiixs1                            3,967            7.8      1,322.3
leaf node 90-10 splits                             7            0.0          2.3
leaf node splits                                  94            0.2         31.3
logons cumulative                                  2            0.0          0.7
messages received                                519            1.0        173.0
messages sent                                    519            1.0        173.0
no buffer to keep pinned count                     0            0.0          0.0
no work - consistent read gets                35,956           70.4     11,985.3
opened cursors cumulative                      1,458            2.9        486.0
parse count (failures)                             1            0.0          0.3
parse count (hard)                               119            0.2         39.7
parse count (total)                            1,464            2.9        488.0
parse time cpu                                    50            0.1         16.7
parse time elapsed                                78            0.2         26.0
physical reads                                16,298           31.9      5,432.7
physical reads direct                            616            1.2        205.3
physical writes                                9,207           18.0      3,069.0
physical writes direct                         7,236           14.2      2,412.0
physical writes non checkpoint                 9,148           17.9      3,049.3
pinned buffers inspected                           1            0.0          0.3
prefetched blocks                             11,254           22.0      3,751.3
prefetched blocks aged out before                 63            0.1         21.0
process last non-idle time             1,114,679,751    2,181,369.4 ############
recursive calls                               33,289           65.1     11,096.3
recursive cpu usage                               84            0.2         28.0
redo blocks written                            3,011            5.9      1,003.7
redo buffer allocation retries                     2            0.0          0.7
redo entries                                   4,193            8.2      1,397.7
redo log space requests                            2            0.0          0.7
redo log space wait time                           0            0.0          0.0
redo size                                  1,374,888        2,690.6    458,296.0
redo synch time                                    3            0.0          1.0
redo synch writes                                  5            0.0          1.7
redo wastage                                   7,160           14.0      2,386.7
redo write time                                    1            0.0          0.3
redo writer latching time                          0            0.0          0.0
redo writes                                       27            0.1          9.0
rollback changes - undo records a                  4            0.0          1.3
rollbacks only - consistent read                   0            0.0          0.0
rows fetched via callback                      4,243            8.3      1,414.3
session connect time                   1,114,679,751    2,181,369.4 ############
session logical reads                         63,074          123.4     21,024.7
session pga memory max                       235,268          460.4     78,422.7
session uga memory max                       607,908        1,189.6    202,636.0
shared hash latch upgrades - no w              4,220            8.3      1,406.7
sorts (disk)                                       4            0.0          1.3
sorts (memory)                                   344            0.7        114.7
sorts (rows)                                  13,001           25.4      4,333.7
summed dirty queue length                      2,356            4.6        785.3
switch current to new buffer                       1            0.0          0.3
table fetch by rowid                          50,104           98.1     16,701.3
table fetch continued row                          3            0.0          1.0
table scan blocks gotten                       8,019           15.7      2,673.0
table scan rows gotten                         8,440           16.5      2,813.3
table scans (long tables)                         13            0.0          4.3

Instance Activity Stats for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2

Statistic                                      Total     per Second    per Trans
```

```
--------------------------------- ------------------ -------------- ------------
table scans (short tables)                        87            0.2         29.0
transaction rollbacks                              2            0.0          0.7
user calls                                       248            0.5         82.7
user commits                                       3            0.0          1.0
write clones created in backgroun                  1            0.0          0.3
write clones created in foregroun                 16            0.0          5.3
          -------------------------------------------------------------

Tablespace IO Stats for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
->ordered by IOs (Reads + Writes) desc

Tablespace
------------------------------
                 Av      Av     Av                    Av        Buffer Av Buf
         Reads Reads/s Rd(ms) Blks/Rd       Writes Writes/s      Waits Wt(ms)
-------------- ------- ------ ------- ------------ -------- ---------- ------
SYSTEM
         7,683      15    1.2     2.0        1,077        2          0    0.0
STATPACK_TEMP
           158       0    0.0     3.9        5,251       10          0    0.0
TEMP
             4       0    0.0     1.0        2,188        4          0    0.0
STATPACK_TSP
           148       0    0.0     1.0          347        1          0    0.0
UNDO_RBS
             4       0    0.0     1.0          344        1          2    0.0
          -------------------------------------------------------------

File IO Stats for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
->ordered by Tablespace, File

Tablespace               Filename
------------------------ ----------------------------------------------------
                 Av      Av     Av                    Av        Buffer Av Buf
         Reads Reads/s Rd(ms) Blks/Rd       Writes Writes/s      Waits Wt(ms)
-------------- ------- ------ ------- ------------ -------- ---------- ------
STATPACK_TEMP            /u03/oradata/CURSOxy/statpack_temp01.dbf
           158       0    0.0     3.9        5,251       10          0

STATPACK_TSP             /u02/oradata/CURSOxy/statpack_tsp01.dbf
           148       0    0.0     1.0          347        1          0

SYSTEM                   /u02/oradata/CURSOxy/system01.dbf
         7,683      15    1.2     2.0        1,077        2          0

TEMP                     /u03/oradata/CURSOxy/temp01.dbf
             4       0    0.0     1.0        2,188        4          0

UNDO_RBS                 /u03/oradata/CURSOxy/undo_rbs01.dbf
             4       0    0.0     1.0          344        1          2    0.0

          -------------------------------------------------------------

Buffer Pool Statistics for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> Standard block size Pools  D: default,  K: keep,  R: recycle
-> Default Pools for other block sizes: 2k, 4k, 8k, 16k, 32k

                                                           Free    Write  Buffer
     Number of Cache      Buffer    Physical   Physical  Buffer Complete    Busy
P      Buffers Hit %        Gets       Reads     Writes   Waits    Waits   Waits
--- ---------- ----- ----------- ----------- ---------- ------- --------  ------
D        1,875  82.4      88,956      15,678      1,971       0        0       2
          -------------------------------------------------------------
```

```
Instance Recovery Stats for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> B: Begin snapshot,  E: End snapshot

  Targt Estd                                    Log File   Log Ckpt   Log Ckpt
  MTTR  MTTR   Recovery    Actual     Target      Size     Timeout    Interval
   (s)   (s)   Estd IOs  Redo Blks  Redo Blks  Redo Blks  Redo Blks  Redo Blks
- ----- ----- ---------- ---------- ---------- ---------- ---------- ----------
B     0     5        360        354        326       1836        326
E     0     5        277       1677       1836       1836       3029
          -------------------------------------------------------------
Buffer Pool Advisory for DB: CURSOxy  Instance: CURSOxy  End Snap: 3
-> Only rows with estimated physical reads >0 are displayed
-> ordered by Block Size, Buffers For Estimate

        Size for  Size      Buffers for  Est Physical          Estimated
P   Estimate (M) Factr         Estimate   Read Factor     Physical Reads
--- ------------ ----- ---------------- ------------- ------------------
D              4   1.0            1,875          1.00             24,958
D              8   2.0            3,750          0.80             19,843
D             12   3.0            5,625          0.60             15,036
D             16   4.0            7,500          0.59             14,844
D             20   5.0            9,375          0.51             12,729
D             24   6.0           11,250          0.50             12,421
D             28   7.0           13,125          0.50             12,421
D             32   8.0           15,000          0.50             12,421
D             36   9.0           16,875          0.50             12,421
D             40  10.0           18,750          0.50             12,421
D             44  11.0           20,625          0.50             12,421
D             48  12.0           22,500          0.50             12,421
D             52  13.0           24,375          0.50             12,421
D             56  14.0           26,250          0.50             12,421
D             60  15.0           28,125          0.50             12,421
D             64  16.0           30,000          0.50             12,421
D             68  17.0           31,875          0.50             12,421
D             72  18.0           33,750          0.50             12,421
D             76  19.0           35,625          0.50             12,421
D             80  20.0           37,500          0.50             12,421
          -------------------------------------------------------------


Buffer wait Statistics for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> ordered by wait time desc, waits desc

                                 Tot Wait    Avg
Class                    Waits   Time (s) Time (ms)
------------------ ----------- ---------- ---------
undo header                  1          0         0
          -------------------------------------------------------------

Rollback Segment Stats for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
->A high value for "Pct Waits" suggests more rollback segments may be required
->RBS stats may not be accurate between begin and end snaps when using Auto Undo
  managment, as RBS may be dynamically created and dropped as needed

        Trans Table       Pct   Undo Bytes
RBS No      Gets        Waits     Written        Wraps  Shrinks  Extends
------ -------------- ------- --------------- -------- -------- --------
     0            5.0    0.00               0        0        0        0
     1            7.0    0.00             110        0        0        0
     2          374.0    0.27         258,440        6        3        4
     3            7.0    0.00             110        0        0        0
     4          245.0    0.41         150,970        3        4        2
          -------------------------------------------------------------

Rollback Segment Storage for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
```

```
->Optimal Size should be larger than Avg Active

RBS No    Segment Size      Avg Active    Optimal Size    Maximum Size
------ --------------- --------------- --------------- ---------------
     0         456,704               0                         456,704
     1      18,872,320          90,131                      18,872,320
     2         522,240         232,831                      19,920,896
     3      18,872,320               0                      18,872,320
     4         325,632         110,190                      20,969,472
          -------------------------------------------------------------

Undo Segment Summary for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> Undo segment block stats:
-> uS - unexpired Stolen,   uR - unexpired Released,   uU - unexpired reUsed
-> eS - expired   Stolen,   eR - expired   Released,   eU - expired   reUsed

Undo           Undo        Num  Max Qry     Max Tx Snapshot Out of uS/uR/uU/
 TS#         Blocks      Trans  Len (s)   Concurcy  Too Old  Space eS/eR/eU
---- -------------- ---------- -------- ---------- -------- ------ -------------
   1            227      2,283        0          1        0      0 0/0/0/0/0/0
          -------------------------------------------------------------


Undo Segment Stats for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> ordered by Time desc

                     Undo      Num Max Qry   Max Tx  Snap   Out of uS/uR/uU/
End Time           Blocks    Trans Len (s)    Concy Too Old  Space eS/eR/eU
------------ ------------ -------- ------- -------- ------- ------ -------------
28-Apr 11:09          227    2,283       0        1       0      0 0/0/0/0/0/0
          -------------------------------------------------------------

Latch Activity for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
->"Get Requests", "Pct Get Miss" and "Avg Slps/Miss" are statistics for
  willing-to-wait latch get requests
->"NoWait Requests", "Pct NoWait Miss" are for no-wait latch get requests
->"Pct Misses" for both should be very close to 0.0

                                           Pct    Avg   Wait                 Pct
                              Get          Get   Slps   Time       NoWait NoWait
Latch                       Requests      Miss  /Miss    (s)     Requests   Miss
------------------------ -------------- ------ ------ ------ ------------ ------
Consistent RBA                       29    0.0             0            0
FIB s.o chain latch                  20    0.0             0            0
FOB s.o list latch                   35    0.0             0            0
SQL memory manager worka            134    0.0             0            0
active checkpoint queue             624    0.0             0            0
archive control                       2    0.0             0            0
cache buffer handles                 22    0.0             0            0
cache buffers chains            172,312    0.0             0       28,476    0.0
cache buffers lru chain          26,789    0.0             0           18    0.0
channel handle pool latc              4    0.0             0            0
channel operations paren            373    0.0             0            0
checkpoint queue latch           31,027    0.0             0        1,668    0.0
child cursor hash table           2,122    0.0             0            0
dml lock allocation                 164    0.0             0            0
dummy allocation                      4    0.0             0            0
enqueue hash chains               3,900    0.0             0            0
enqueues                          3,495    0.0    1.0      0            0
event group latch                     2    0.0             0            0
file number translation           7,810    0.0             0            0
hash table column usage               0                    0        7,742    0.0
hash table modification               1    0.0             0            0
ktm global data                       4    0.0             0            0
lgwr LWN SCN                        179    0.0             0            0
```

```
library cache                    24,345    0.0             0          405    0.0
library cache load lock             262    0.0             0            0
library cache pin                11,372    0.0             0            0
library cache pin alloca          7,072    0.0             0            0
list of block allocation             22    0.0             0            0
loader state object free             40    0.0             0            0
messages                          2,457    0.0             0            0
mostly latch-free SCN               179    0.0             0            0
multiblock read objects           9,254    0.0             0            0
ncodef allocation latch               8    0.0             0            0
object stats modificatio          1,007    0.0             0            0
post/wait queue                      59    0.0             0            8    0.0
process allocation                    2    0.0             0            2    0.0
process group creation                4    0.0             0            0
redo allocation                   4,436    0.0             0            0
redo copy                             0                    0        4,221    0.0
redo writing                      1,062    0.0             0            0
row cache enqueue latch          17,148    0.0             0            0
row cache objects                17,439    0.0             0          596    0.0
sequence cache                        9    0.0             0            0
session allocation                   87    0.0             0            0
session idle bit                    528    0.0             0            0
session switching                     8    0.0             0            0
session timer                       178    0.0             0            0
shared pool                      13,268    0.0             0            0
simulator hash latch              3,720    0.0             0            0
simulator lru latch               1,156    0.0             0            3    0.0

Latch Activity for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
->"Get Requests", "Pct Get Miss" and "Avg Slps/Miss" are statistics for
  willing-to-wait latch get requests
->"NoWait Requests", "Pct NoWait Miss" are for no-wait latch get requests
->"Pct Misses" for both should be very close to 0.0

                                           Pct    Avg   Wait                 Pct
                              Get          Get   Slps   Time       NoWait NoWait
Latch                       Requests      Miss  /Miss    (s)     Requests   Miss
------------------------ -------------- ------ ------ ------ ------------ ------
sort extent pool                    392    0.0             0            0
transaction allocation                4    0.0             0            0
transaction branch alloc              8    0.0             0            0
undo global data                  1,800    0.0             0            0
user lock                             8    0.0             0            0
          -------------------------------------------------------------

Latch Sleep breakdown for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> ordered by misses desc

                                      Get                            Spin &
Latch Name                       Requests      Misses      Sleeps Sleeps 1->4
-------------------------- -------------- ----------- ----------- ------------
enqueues                            3,495           1           1 0/1/0/0/0
          -------------------------------------------------------------

Latch Miss Sources for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
-> only latches with sleeps are shown
-> ordered by name, sleeps desc

                                                     NoWait              Waiter
Latch Name               Where                       Misses     Sleeps   Sleeps
------------------------ -------------------------- ------- ---------- --------
enqueues                 ksqgel: create enqueue           0          1        1
          -------------------------------------------------------------

Dictionary Cache Stats for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
```

```
->"Pct Misses"  should be very low (< 2% in most cases)
->"Cache Usage" is the number of cache entries being used
->"Pct SGA"     is the ratio of usage to allocated size for that cache

                                   Get    Pct    Scan   Pct      Mod      Final
Cache                         Requests   Miss    Reqs  Miss     Reqs      Usage
------------------------- ------------ ------ ------- ----- -------- ----------
dc_histogram_defs                  513   41.9       0              0        215
dc_object_ids                      749    3.6       0              0        205
dc_objects                         448    7.8       0              0        463
dc_profiles                          1    0.0       0              0          1
dc_rollback_segments                37    0.0       0              0          6
dc_segments                        399    1.3       0              0        105
dc_tablespaces                   3,724    0.0       0              0          5
dc_user_grants                       7    0.0       0              0          5
dc_usernames                       134    0.0       0              0          4
dc_users                         2,679    0.0       0              0          5
          -------------------------------------------------------------


Library Cache Activity for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2
->"Pct Misses"  should be very low

                         Get  Pct        Pin        Pct               Invali-
Namespace           Requests  Miss     Requests     Miss     Reloads  dations
--------------- ------------ ------ -------------- ------ ---------- --------
BODY                       6    0.0              6    0.0          0        0
CLUSTER                   66    0.0             43    0.0          0        0
SQL AREA               1,438    6.6          5,128    5.0          2        0
TABLE/PROCEDURE          792    8.2          1,115   14.6         20        0
          -------------------------------------------------------------

Shared Pool Advisory for DB: CURSOxy  Instance: CURSOxy  End Snap: 3
-> Note there is often a 1:Many correlation between a single logical object
   in the Library Cache, and the physical number of memory objects associated
   with it.  Therefore comparing the number of Lib Cache objects (e.g. in
   v$librarycache), with the number of Lib Cache Memory Objects is invalid

                                                          Estd
Shared Pool    SP       Estd         Estd     Estd Lib LC Time
   Size for  Size  Lib Cache    Lib Cache   Cache Time   Saved  Estd Lib Cache
  Estim (M) Factr   Size (M)      Mem Obj    Saved (s)   Factr    Mem Obj Hits
----------- ----- ---------- ------------ ------------ ------- ---------------
          4    .5          4        1,048           29     1.0          11,431
          8   1.0          7        2,668           29     1.0          11,499
         12   1.5          7        2,736           29     1.0          11,499
         16   2.0          7        2,736           29     1.0          11,499
          -------------------------------------------------------------

SGA Memory Summary for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2

SGA regions                       Size in Bytes
------------------------------ ----------------
Database Buffers                      4,194,304
Fixed Size                              450,560
Redo Buffers                          1,191,936
Variable Size                        29,360,128
                               ----------------
sum                                  35,196,928
          -------------------------------------------------------------


SGA breakdown difference for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2

Pool   Name                                Begin value        End value  % Diff
```

```
------ ------------------------------ ---------------- ---------------- -------
shared 1M buffer                             2,098,176        2,098,176    0.00
shared Checkpoint queue                        282,304          282,304    0.00
shared FileIdentificatonBlock                  319,452          319,452    0.00
shared FileOpenBlock                           114,908          114,908    0.00
shared KGK heap                                  3,756            3,756    0.00
shared KGLS heap                             1,123,860        1,027,440   -8.58
shared KQR M PO                                532,008          628,776   18.19
shared KQR S PO                                110,084          110,084    0.00
shared KQR S SO                                  1,280            1,280    0.00
shared KSXR large reply queue                  166,024          166,024    0.00
shared KSXR pending messages que               840,636          840,636    0.00
shared KSXR pending reply queue                 90,292           90,292    0.00
shared KSXR receive buffers                  1,032,500        1,032,500    0.00
shared PL/SQL DIANA                          2,977,928          414,408  -86.08
shared PL/SQL MPCODE                           161,288          178,316   10.56
shared PLS non-lib hp                            2,068            2,068    0.00
shared SYSTEM PARAMETERS                        96,836           96,836    0.00
shared character set object                    279,700          279,700    0.00
shared dictionary cache                      1,610,880        1,610,880    0.00
shared event statistics per sess               223,608          223,608    0.00
shared fixed allocation callback                   188              188    0.00
shared free memory                             675,940        1,039,300   53.76
shared joxs heap init                              784              784    0.00
shared kgl simulator                           597,052          626,704    4.97
shared kglsim sga                              134,144          134,144    0.00
shared krvxrr                                  126,528          126,528    0.00
shared ksm_file2sga region                     148,652          148,652    0.00
shared library cache                         2,758,012        2,877,900    4.35
shared message pool freequeue                  833,032          833,032    0.00
shared miscellaneous                         1,789,964        1,974,136   10.29
shared parameters                                4,252            5,504   29.44
shared sim memory hea                           71,996           71,996    0.00
shared sim trace entries                        98,304           98,304    0.00
shared sql area                              1,591,876        3,438,184  115.98
shared table definiti                            1,144            2,656  132.17
shared trace events array                       72,000           72,000    0.00
shared trigger inform                               64               64    0.00
       buffer_cache                          4,194,304        4,194,304    0.00
       fixed_sga                               450,560          450,560    0.00
       log_buffer                            1,180,672        1,180,672    0.00
          -------------------------------------------------------------

init.ora Parameters for DB: CURSOxy  Instance: CURSOxy  Snaps: 1 -2

                                                                  End value
Parameter Name                Begin value                       (if different)
----------------------------- --------------------------------- --------------
background_dump_dest          /u01/app/oracle/admin/CURSOxy/bdu
compatible                    9.2.0
control_files                 /u02/oradata/CURSOxy/control1.ctl
core_dump_dest                /u01/app/oracle/admin/CURSOxy/cdu
db_cache_size                 4194304
db_name                       CURSOxy
java_pool_size                0
large_pool_size               0
log_buffer                    1048576
log_checkpoint_interval       0
log_checkpoint_timeout        1800
max_dump_file_size            10240
processes                     16
remote_login_passwordfile     EXCLUSIVE
sga_max_size                  35196928
shared_pool_size              8388608
undo_management               AUTO
```

```
undo_tablespace               undo_rbs
user_dump_dest                /u01/app/oracle/admin/CURSOxy/udu
          -------------------------------------------------------------

End of Report
```

4.15. Probar autotrace de sqlplus.

Solución:

Antes de usar AUTOTRACE tenemos que tener una tabla PLAN_TABLE accesible. Vamos a instalar dicha tabla bajo el usuario SYSTEM. Luego vamos a ejecutar una consulta sobre el DD indicando claúsula WHERE q puede usar índices. Una vez lanzada la sentencia, y visto el plan de ejecución, cambiaremos el modo del optimizador para ver un plan “diferente”:

```
SQL> connect system/systcursoXY
Enter password:
Connected.
SQL> @/u01/app/oracle/product/9.2.0.1.0/rdbms/admin/utlxplan.sql
Table created.

SQL> set autotrace traceonly explain
SQL> select count(*) from dba_objects where status='VALID';
Execution Plan
----------------------------------------------------------
 0 SELECT STATEMENT Optimizer=CHOOSE
 1                  0   SORT (AGGREGATE)
 2                  1     VIEW OF 'DBA_OBJECTS'
 3                  2       UNION-ALL
 4                  3         FILTER
 5                  4           TABLE ACCESS (BY INDEX ROWID) OF 'OBJ$'
 6                  5             NESTED LOOPS
 7                  6               TABLE ACCESS (FULL) OF 'USER$'
 8                  6               INDEX (RANGE SCAN) OF 'I_OBJ2' (UNIQUE)
 9                  4           TABLE ACCESS (BY INDEX ROWID) OF 'IND$'
10                  9             INDEX (UNIQUE SCAN) OF 'I_IND1' (UNIQUE)
11                  3         NESTED LOOPS
12                 11           TABLE ACCESS (FULL) OF 'USER$'
13                 11           INDEX (RANGE SCAN) OF 'I_LINK1' (NON-UNIQUE)

SQL> alter session set optimizer_mode=first_rows;
Session altered.

SQL> select count(*) from dba_objects where status='VALID';
Execution Plan
----------------------------------------------------------
 0 SELECT STATEMENT Optimizer=FIRST_ROWS (Cost=65 Card=1 Bytes=5)
 1  0   SORT (AGGREGATE)
 2  1     VIEW OF 'DBA_OBJECTS' (Cost=65 Card=2025 Bytes=10125)
 3  2       UNION-ALL
 4  3         FILTER
 5  4           NESTED LOOPS (Cost=18 Card=1 Bytes=148)
 6  5              TABLE ACCESS (FULL) OF 'OBJ$' (Cost=17 Card=1 Bytes=135)
 7  5             TABLE ACCESS (CLUSTER) OF 'USER$' (Cost=1 Card=1 Bytes=13)
 8  7              INDEX (UNIQUE SCAN) OF 'I_USER#' (NON-UNIQUE)
 9  4           TABLE ACCESS (BY INDEX ROWID) OF 'IND$' (Cost=2 Card=1 bytes=26)
10  9             INDEX (UNIQUE SCAN) OF 'I_IND1' (UNIQUE) (Cost=1 Card=2024)
11  3         NESTED LOOPS (Cost=47 Card=2024 Bytes=52624)
12  11           INDEX (FULL SCAN) OF 'I_LINK1' (NON-UNIQUE) (Cost=26 Card=21 Bytes=273)
13  11           TABLE ACCESS (CLUSTER) OF 'USER$' (Cost=1 Card=96 Bytes=1248)
```

```
14  13             INDEX (UNIQUE SCAN) OF 'I_USER#' (NON-UNIQUE)

SQL> set autotrace traceonly statistics
SQL> select count(*) from dba_objects where status='VALID';

Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
      13435  consistent gets
        349  physical reads
          0  redo size
        380  bytes sent via SQL*Net to client
        503  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

Ya hemos visto como ver de forma automática el plan de ejecución de una sentencia, así como las estadísticas de E/S. También podemos calcular el tiempo de repuesta al usuario con SET TIMING:

```
SQL> set autotrace off
SQL> set timing on
SQL> select count(*) from dba_objects where status='VALID'

  COUNT(*)
----------
      5918
Elapsed: 00:00:00.06
```