

# Oracle Database

## Size

```

/* Get total size allocated */
SELECT SUM(bytes)/1024/1024 size_in_mb FROM dba_data_files;

/* Get total space used */
SELECT SUM(bytes)/1024/1024 size_in_mb FROM dba_segments;

/* Break down size usage by user */
SELECT owner, SUM(bytes)/1024/1024 Size_MB FROM dba_segments GROUP BY
owner;

```

## Monthly DB growth

```

SELECT to_char(CREATION_TIME, 'RRRR') YEAR, to_char(CREATION_TIME, 'MM')
MONTH, round(SUM(bytes)/1024/1024/1024) GB
FROM v$datafile
GROUP BY to_char(CREATION_TIME, 'RRRR'), to_char(CREATION_TIME, 'MM')
ORDER BY 1, 2;

```

## Get DB growth over time data lifetime

```

SET LINESIZE 200
SET PAGESIZE 200
COL "Database Size" FORMAT a13
COL "Used Space" FORMAT a11
COL "Used in %" FORMAT a11
COL "Free in %" FORMAT a11
COL "Database Name" FORMAT a13
COL "Free Space" FORMAT a12
COL "Growth DAY" FORMAT a11
COL "Growth WEEK" FORMAT a12
COL "Growth DAY in %" FORMAT a16
COL "Growth WEEK in %" FORMAT a16
SELECT
(SELECT MIN(creation_time) FROM v$datafile) "Create Time",
(SELECT name FROM v$database) "Database Name",
ROUND((SUM(USED.BYTES) / 1024 / 1024 ),2) || ' MB' "Database Size",
ROUND((SUM(USED.BYTES) / 1024 / 1024 ) - ROUND(FREE.P / 1024 / 1024 ),2) ||
' MB' "Used Space",
ROUND(((SUM(USED.BYTES) / 1024 / 1024 ) - (FREE.P / 1024 / 1024 )) /
ROUND(SUM(USED.BYTES) / 1024 / 1024 ,2)*100,2) || '% MB' "Used in %",
ROUND((FREE.P / 1024 / 1024 ),2) || ' MB' "Free Space",
ROUND(((SUM(USED.BYTES) / 1024 / 1024 ) - ((SUM(USED.BYTES) / 1024 / 1024 )

```

```

- ROUND(FREE.P / 1024 / 1024 ))/ROUND(SUM(USED.BYTES) / 1024 / 1024,2
)*100,2) || '% MB' "Free in %",
ROUND(((SUM(USED.BYTES) / 1024 / 1024 ) - (FREE.P / 1024 / 1024 ))/(SELECT
sysdate-MIN(creation_time) FROM v$datafile),2) || ' MB' "Growth DAY",
ROUND(((SUM(USED.BYTES) / 1024 / 1024 ) - (FREE.P / 1024 / 1024 ))/(SELECT
sysdate-MIN(creation_time) FROM v$datafile)/ROUND((SUM(USED.BYTES) / 1024 /
1024 ),2)*100,3) || '% MB' "Growth DAY in %",
ROUND(((SUM(USED.BYTES) / 1024 / 1024 ) - (FREE.P / 1024 / 1024 ))/(SELECT
sysdate-MIN(creation_time) FROM v$datafile)*7,2) || ' MB' "Growth WEEK",
ROUND((((SUM(USED.BYTES) / 1024 / 1024 ) - (FREE.P / 1024 / 1024 ))/(SELECT
sysdate-MIN(creation_time) FROM v$datafile)/ROUND((SUM(USED.BYTES) / 1024 /
1024 ),2)*100)*7,3) || '% MB' "Growth WEEK in %"
FROM (SELECT BYTES FROM V$DATAFILE
UNION ALL
SELECT BYTES FROM V$TEMPFILE
UNION ALL
SELECT BYTES FROM V$LOG) USED,
(SELECT SUM(BYTES) AS P FROM DBA_FREE_SPACE) FREE
GROUP BY FREE.P;

```

## Archived Redo Logs

### Identify tablespaces generating redo logs

```
SELECT tablespace_name, contents, logging FROM dba_tablespaces;
```

TABLESPACE_NAME	CONTENTS	LOGGING
SYSTEM	PERMANENT	LOGGING
SYSAUX	PERMANENT	LOGGING
UNDOTBS1	UNDO	LOGGING
TEMP	TEMPORARY	NOLOGGING
USERS	PERMANENT	LOGGING
UNDOTBS2	UNDO	LOGGING
UNDOTBS3	UNDO	LOGGING
APEX	PERMANENT	LOGGING
0_TS_DATA1	PERMANENT	NOLOGGING
0_TS_INDEX1	PERMANENT	NOLOGGING
0_TS_DATA2	PERMANENT	NOLOGGING
0_TS_INDEX2	PERMANENT	NOLOGGING
0_TS_DATA3	PERMANENT	NOLOGGING
0_TS_INDEX3	PERMANENT	NOLOGGING
TOOLS	PERMANENT	LOGGING
0_OSH_DATA1	PERMANENT	LOGGING
0_OSH_INDEX1	PERMANENT	LOGGING

```
SELECT tablespace_name, force_logging FROM dba_tablespaces;
```

```
ALTER tablespace 0_OSH_INDEX1 nologging;
```

```

SELECT * FROM USER_TABLES;
SELECT TABLE_NAME, tablespace_name, logging FROM USER_TABLES;
SELECT TABLE_NAME, tablespace_name, logging FROM USER_TABLES WHERE
logging='YES';

```

## Archive log sizes Per Day

```

SELECT SUM_ARCH.DAY,
       SUM_ARCH.GENERATED_MB,
       SUM_ARCH_DEL.DELETED_MB,
       SUM_ARCH.GENERATED_MB - SUM_ARCH_DEL.DELETED_MB "REMAINING_MB"
FROM ( SELECT TO_CHAR (COMPLETION_TIME, 'DD/MM/YYYY') DAY,
           SUM (ROUND ( (blocks * block_size) / (1024 * 1024), 2))
           GENERATED_MB
       FROM V$ARCHIVED_LOG
       WHERE ARCHIVED = 'YES'
       GROUP BY TO_CHAR (COMPLETION_TIME, 'DD/MM/YYYY')) SUM_ARCH,
      ( SELECT TO_CHAR (COMPLETION_TIME, 'DD/MM/YYYY') DAY,
           SUM (ROUND ( (blocks * block_size) / (1024 * 1024), 2))
           DELETED_MB
       FROM V$ARCHIVED_LOG
       WHERE ARCHIVED = 'YES' AND DELETED = 'YES'
       GROUP BY TO_CHAR (COMPLETION_TIME, 'DD/MM/YYYY')) SUM_ARCH_DEL
WHERE SUM_ARCH.DAY = SUM_ARCH_DEL.DAY(+)
ORDER BY TO_DATE (DAY, 'DD/MM/YYYY');

```

*/\* or \*/*

```

SELECT trunc(COMPLETION_TIME, 'DD') DAY,
       thread#,
       round(SUM(BLOCKS*BLOCK_SIZE)/1024/1024/1024) GB,
       COUNT(*) Archives_Generated
FROM v$archived_log
GROUP BY trunc(COMPLETION_TIME, 'DD'), thread#
ORDER BY 1;

```

To find sessions generating lots of redo, you can use either of the following methods. Both methods examine the amount of undo generated. When a transaction generates undo, it will automatically generate redo as well.

The methods are:

1) Query **V\$SESS\_IO**. This view contains the column **BLOCK\_CHANGES** which indicates how much blocks have been changed by the session. High values indicate a session generating lots of redo.

The query you can use is:

```

SELECT s.sid, s.serial#, s.username, s.program,
       i.block_changes

```

```
FROM v$session s, v$sess_io i
WHERE s.sid = i.sid
ORDER BY 5 DESC, 1, 2, 3, 4;
```

Run the query multiple times and examine the delta between each occurrence of **BLOCK\_CHANGES**. Large deltas indicate high redo generation by the session.

2) Query **V\$TRANSACTION**. This view contains information about the amount of undo blocks and undo records accessed by the transaction (as found in the **USED\_UBLK** and **USED\_UREC** columns).

The query you can use is:

```
SELECT s.sid, s.serial#, s.username, s.program,
t.used_ublk, t.used_urec
FROM v$session s, v$transaction t
WHERE s.taddr = t.addr
ORDER BY 5 DESC, 6 DESC, 1, 2, 3, 4;
```

Run the query multiple times and examine the delta between each occurrence of **USED\_UBLK** and **USED\_UREC**. Large deltas indicate high redo generation by the session.

## Performance Queries

### Get process and session count for instance

```
SELECT resource_name, current_utilization, max_utilization
FROM v$resource_limit WHERE resource_name IN ('processes', 'sessions');
```

### Query Average active sessions

```
SELECT round((COUNT(ash.sample_id) / ((CAST(end_time.sample_time AS DATE) -
CAST(start_time.sample_time AS DATE))*24*60*60)),2) AS AAS
FROM
  (SELECT MIN(sample_time) sample_time
   FROM v$active_session_history ash
  ) start_time,
  (SELECT MAX(sample_time) sample_time
   FROM v$active_session_history
  ) end_time,
  v$active_session_history ash
WHERE ash.sample_time BETWEEN start_time.sample_time AND
end_time.sample_time
GROUP BY end_time.sample_time, start_time.sample_time;
```

### As above but for the last hour

```

SELECT round((COUNT(ash.sample_id) / ((CAST(end_time.sample_time AS DATE) -
CAST(start_time.sample_time AS DATE))*24*60*60)),2) AS AAS
FROM
  (SELECT MIN(sample_time) sample_time
   FROM v$active_session_history ash
   WHERE sample_time BETWEEN sysdate-1/24 AND sysdate) start_time,
  (SELECT MAX(sample_time) sample_time
   FROM v$active_session_history
   WHERE sample_time BETWEEN sysdate-1/24 AND sysdate) end_time,
v$active_session_history ash
WHERE ash.sample_time BETWEEN start_time.sample_time AND
end_time.sample_time
GROUP BY end_time.sample_time,start_time.sample_time;

```

### Show DB block changes

```

SET pagesize 200 linesize 200
col owner format a10
col statistic_name format a10
col perc format 99.99
col object_type format a10
col VALUE format 999,999,999,999.99
col statistic_name format a30

SELECT a.inst_id,
       a.statistic_name,
       a.owner,
       a.object_name,
       a.object_type,
       a.value,
       (a.value / b.sum_value) * 100 perc
FROM (SELECT *
      FROM (SELECT inst_id,
                   owner,
                   object_name,
                   object_type,
                   VALUE,
                   rank() OVER(partition BY inst_id, statistic_name
ORDER BY VALUE DESC) rnk,
                   statistic_name
        FROM gv$segment_statistics
        WHERE VALUE > 0)
      WHERE rnk < 11) a,
      (SELECT inst_id, statistic_name, SUM(VALUE) sum_value
       FROM gv$segment_statistics
       GROUP BY statistic_name, inst_id) b
WHERE a.statistic_name = b.statistic_name
      AND a.inst_id = b.inst_id
      AND a.statistic_name = 'db block changes'

```

```
ORDER BY a.inst_id, a.statistic_name, a.value DESC;
```

### To find the execution plans of currently long-running queries

```
SET LINES 200 pages 200
col operation FOR a32
col plan_options FOR a20
col plan_object_name FOR a24
col id FOR 999
break ON sql_id ON plan_hash

SELECT sql_id, sql_plan_hash_value plan_hash, plan_line_id id, lpad (' ',
plan_depth) || plan_operation operation , plan_options , plan_object_name ,
plan_cardinality card, plan_cost
FROM v$sql_plan_monitor
WHERE STATUS = 'EXECUTING'
ORDER BY KEY, id;
```

### Gathering system statistics in Exadata mode (if needed)

To see if Exadata specific optimizer stats have been gathered, run the following query on a system with at least 11.2.0.2 BP18 or 11.2.0.3 BP8 Oracle software. If PVAL1 returns null or is not set, Exadata specific stats have not been gathered.

```
SELECT pname, PVAL1 FROM aux_stats$ WHERE pname='MBRC';

EXEC dbms_stats.gather_system_stats('EXADATA');

SELECT * FROM sys.aux_stats$;
```

### Show load balancing advisory events

```
SET PAGES 60 COLSEP '|' LINES 132 NUM 8 VERIFY OFF FEEDBACK OFF
COLUMN user_data HEADING "AQ Service Metrics" FORMAT A60 WRAP
BREAK ON service_name SKIP 1
SELECT
  TO_CHAR(enq_time, 'HH:MI:SS') Enq_time, user_data
FROM sys.sys$service_metrics_tab
ORDER BY 1 ;
```

## Memory

Memory allocation for an Oracle DB is tricky business, and requires some thought and examination.

The Maximum Availability Architecture document suggests:

OLTP applications:

**SUM of all databases' (SGA\_TARGET + PGA\_AGGREGATE\_TARGET) + 4MB \* (Maximum PROCESSES ) < Physical Memory per Database Node**

Some documents suggest up to 10MB for the per process memory allocation.

Data Warehouse applications:

**SUM of databases (SGA\_TARGET + 3\* PGA\_AGGREGATE\_TARGET) < Physical Memory per Database Node**

**get the number of processes**

```
SHOW parameter processes;
```

**consider the maximum and average memory used by a process when sizing this value**

```
SELECT AVG(ALLOCATED_AVG) FROM DBA_HIST_PROCESS_MEM_SUMMARY;
SELECT MAX(ALLOCATED_AVG) FROM DBA_HIST_PROCESS_MEM_SUMMARY;
```

**Show PGA size and stats**

```
COLUMN VALUE format 9999999999999999;
SELECT * FROM v$pgastat;
SHOW PARAMETER PGA_AGGREGATE_TARGET;

/* maximum historical PGA usage */
SELECT MAX(VALUE)/1024/1024 MaxMB FROM dba_hist_pgastat WHERE name='maximum
PGA allocated';
```

**Show SGA size, pool subareas and stats**

```
show parameter sga_target;
show parameter sga_max_size;
select pool, max(bytes)/1024/1024 as MaxMB from dba_hist_sgastat group by
pool order by max(bytes)/1024/1024 desc;
SELECT component, current_size/1024/1024 as size_mb, min_size/1024/1024 as
min_size_mb
FROM v$sga_dynamic_components
WHERE current_size > 0
ORDER BY component;
/* total current SGA */
SELECT SUM(current_size)/1024/1024 FROM v$sga_dynamic_components;
```

**show implication of possible changes to SGA size**

```
SELECT sga_size, sga_size_factor, estd_db_time_factor
FROM v$sga_target_advice
ORDER BY sga_size ASC;
```

Think of the ESTD\_DB\_TIME\_FACTOR as the amount of time required to process an operation which takes 1 second in the current configuration.

### Session current and max memory usage per session

```
SELECT to_char(ssn.sid, '9999') || ' - ' || nvl(ssn.username, nvl(bgp.name,
'background')) ||
nvl(LOWER(ssn.machine), ins.host_name) "SESSION",
to_char(prc.spid, '999999999') "PID/THREAD",
to_char((se1.value/1024)/1024, '999G999G990D00') || ' MB' " CURRENT SIZE",
to_char((se2.value/1024)/1024, '999G999G990D00') || ' MB' " MAXIMUM SIZE"
FROM v$sesstat se1, v$sesstat se2, v$session ssn, v$bgprocess bgp, v$process
prc,
v$instance ins, v$statname stat1, v$statname stat2
WHERE se1.statistic# = stat1.statistic# AND stat1.name = 'session pga
memory'
AND se2.statistic# = stat2.statistic# AND stat2.name = 'session pga memory
max'
AND se1.sid = ssn.sid
AND se2.sid = ssn.sid
AND ssn.paddr = bgp.paddr (+)
AND ssn.paddr = prc.addr (+);
```

### show pga usage per process

```
SELECT
s.sid,
p.spid,
DECODE(s.program, NULL, p.program, s.program) AS "Program",
pga_used_mem,
pga_alloc_mem,
pga_max_mem
FROM v$process p, v$session s
WHERE s.paddr = p.addr
ORDER BY s.sid;
```

### Tuning advice for MEMORY\_TARGET parameter

```
SELECT * FROM v$memory_target_advice ORDER BY memory_size;
```

**List total Oracle DB memory (SGA+PGA)**

```

export ORAENV_ASK=NO
for ORACLE_SID in `ps -ef|grep pmon|grep -v grep|awk -F_ '{print $3}'`
do
. oraenv
sqlplus -s "/ as sysdba" <<e1
set pages 0 head off feed off
select '$ORACLE_SID sga', sum(value)/1024/1024 mb from v\$$sga
/
select '$ORACLE_SID pga', sum(value)/1024/1024 mb from v\$$sesstat s,
v\$$statname n
where s.statistic# = n.statistic#
and n.name = 'session pga memory'
group by '$ORACLE_SID pga'
/
select '$ORACLE_SID sga_max_size', value/1024/1024 mb from v\$$parameter
where name = 'sga_max_size';
select '$ORACLE_SID pga_target', value/1024/1024 mb from v\$$parameter where
name = 'pga_aggregate_target';
e1
done

```

**Parameter Queries****Get SGA size in MB**

```
SELECT SUM(VALUE)/1024/1024 FROM v$$sga;
```

**Get PGA size in MB**

```
SELECT name, VALUE/1024/1024 FROM v$$pgastat WHERE name LIKE 'total PGA a%';
```

**Get memory parameter values**

```

SELECT sid, name, VALUE FROM v$$spparameter
WHERE name IN
('memory_target', 'sga_target', 'sga_max_size',
'pga_aggregate_target', 'memory_max_target',
'use_large_pages');

```

**Buffer Cache Tuning**

```
COLUMN size_for_estimate          FORMAT 999,999,999,999 heading 'Cache Size
```

```
(MB) '
COLUMN buffers_for_estimate      FORMAT 999,999,999 heading 'Buffers'
COLUMN estd_physical_read_factor  FORMAT 999.90 heading 'Estd Phys|Read
Factor'
COLUMN estd_physical_reads       FORMAT 999,999,999 heading 'Estd Phys|
Reads'

SELECT size_for_estimate, buffers_for_estimate, estd_physical_read_factor,
estd_physical_reads
  FROM V$DB_CACHE_ADVICE
 WHERE name           = 'DEFAULT'
    AND block_size    = (SELECT VALUE FROM V$PARAMETER WHERE name =
'db_block_size')
    AND advice_status = 'ON';
```

Cache Size (MB)	Buffers	Estd Phys Read Factor	Estd Phys Reads	
2,432	284,924	6.39	#####	<-- 10% of cache
4,864	569,848	2.10	#####	
7,296	854,772	1.49	#####	
9,728	1,139,696	1.31	#####	
12,160	1,424,620	1.21	#####	
14,592	1,709,544	1.14	#####	
17,024	1,994,468	1.10	#####	
19,456	2,279,392	1.06	#####	
21,888	2,564,316	1.03	#####	
24,320	2,849,240	1.01	#####	
25,216	2,954,212	1.00	#####	<-- Current size of cache
26,752	3,134,164	.99	#####	
29,184	3,419,088	.97	#####	
31,616	3,704,012	.96	#####	
34,048	3,988,936	.94	#####	
36,480	4,273,860	.93	#####	
38,912	4,558,784	.92	#####	
41,344	4,843,708	.92	#####	
43,776	5,128,632	.91	#####	
46,208	5,413,556	.90	#####	
48,640	5,698,480	.85	#####	<-- 200% of cache

**Get allocated cpu**

```
SHOW parameter cpu_count;
```

**Logging**

## Disable Client / Server logging

To disable Oracle Net logging to the sqlnet.log file, add the following parameters and values to the *SQLNET.ORA* file:

```
LOG_DIRECTORY_CLIENT = /dev/null OR LOG_FILE_CLIENT = /dev/null
```

Default location of the net admin files is *ORACLE\_HOME/network/admin*. Any process already running will continue to log to the sqlnet.log file, until it is restarted.

## Disable Listener Logging

In order to disable logging without stopping the listener, using the LSNRCTL command as follows:

```
LSNRCTL>SET current_listener <listener_name> (IF NOT USING DEFAULT LISTENER)
LSNRCTL>SET LOG_STATUS off
LSNRCTL>save_config
```

If you're trying to disable listener logging temporarily, do not issue *SAVE\_CONFIG*. YMMV for scan listeners.

Once listener logging is disabled, you can now safely delete or archive the existing listener log.

To restore logging again and/or to create a new listener log, simply enable logging as follows:

```
LSNRCTL>SET current_listener <listener_name> (IF NOT USING DEFAULT LISTENER)
LSNRCTL>SET LOG_STATUS ON
```

For a more permanent solution, disable logging in the listener.ora file  
*\$ORACLE\_HOME/network/admin/listener.ora*

```
LOGGING_LISTENER = OFF
LOGGING_LISTENER_SCAN1 = OFF
LOGGING_LISTENER_SCAN2 = OFF
LOGGING_LISTENER_SCAN3 = OFF
LOGGING_LISTENER_IB = OFF
TRACE_LEVEL_LISTENER = OFF
```

Once complete, run

```
$ lsnrctl
LSNRCTL> set current_listener <LISTENER_NAME>
LSNRCTL> reload
```

## Clear all listener logs

```
for i in `ls /u01/app/grid/diag/tnslsnr/*/*/alert/log.xml`; do > $i; done
```

## ADRCI diag log management

```
adrci>
SHOW homes
SET home <home>
SHOW control <home>
SET control (SHORTP_POLICY=360) /* hours */
SET control (LONGP_POLICY=4380)
SHOW control
purge
```

## Problems with ADRCI

```
ADR base = "/misc/oracle"
adrci> purge
DIA-48322: Relation [INCIDENT] OF ADR V[2] incompatible WITH V[2] tool
DIA-48210: Relation NOT Found
DIA-48166: error WITH opening ADR block file because file does NOT exist
[/misc/oracle/diag/tnslsnr/yyy/listener/metadata/INCIDENT.ams] [0]

adrci> SHOW home
ADR Homes:
diag/tnslsnr/yyy/listener
adrci> migrate schema
Schema migrated.
adrci> purge
adrci> quit
```

The error is due to the mismatch in the metadata and one can use “*migrate schema*” command(as shown above) to upgrade the metadata of the corresponding diagnostics destination to the level mandated by the invoked adrci utility.

Similarly for errors like “DIA-48318: ADR Relation [INCIDENT] of version=4 cannot be supported”, one has to downgrade the schema using the higher level adrci(adrci>*migrate schema -downgrade*) and then use low level adrci to get away with the error.

## setAllADRCIHomePolicies

```
for f in $( adrci exec="show homes" | grep -v "ADR Homes:" );
do
    echo "set control ${f}:";
    adrci exec="set home $f; set control \ (SHORTP_POLICY=360,
LONGP_POLICY=4380\); show control;" ;
done
```

## purgeAllADRCIHomes

```
for f in $( adrci exec="show homes" | grep -v "ADR Homes:" );
```

```
do
adrci exec="set home $f; purge;" ;
done
```

Change diagnostic files destination:

```
ALTER system SET diagnostic_dest='/u01/app/oracle/admin';
ALTER system SET diagnostic_dest='/u01/app/oracle/admin' scope=BOTH;
```

## Auditing Control

### Show audited options

```
SELECT * FROM DBA_STMT_AUDIT_OPTS ORDER BY user_name,audit_option;
SELECT * FROM DBA_PRIV_AUDIT_OPTS ORDER BY user_name,privilege;
SELECT * FROM DBA_OBJ_AUDIT_OPTS ORDER BY owner,object_name,object_type;
SELECT * FROM ALL_DEF_AUDIT_OPTS;
```

### Empty AUD\$ table

```
SELECT * FROM AUD$;
TRUNCATE TABLE AUD$;
```

### Examples

```
noaudit <table>; /* Don't audit this table */
noaudit ALTER <table>; /* Don't audit alter commands on this table; */
noaudit SELECT <table> BY <username>; /* Don't audit selects on this table
by username */
noaudit INSERT <table> BY <username>; /* Don't audit inserts on this table
by username */
noaudit INSERT TABLE; /* Don't audit inserts on this table */
```

```
SELECT username, action_name FROM dba_audit_trail;
```

```
# DBMS_AUDIT_MGMT IS a package FOR managing audit trail records
BEGIN
  DBMS_AUDIT_MGMT.init_cleanup(
    audit_trail_type          => DBMS_AUDIT_MGMT.AUDIT_TRAIL_ALL,
    default_cleanup_interval => 12 /* hours */);
END;
```

### Showing/Setting Audit Log Parameters

```
SHOW parameter AUDIT_SYS_OPERATIONS;
SHOW parameter AUDIT_SYSLOG_LEVEL;

ALTER system SET AUDIT_SYSLOG_LEVEL='local0.info' scope=spfile sid='*';
ALTER system SET AUDIT_SYS_OPERATIONS=FALSE scope=spfile sid='*';
```

## Session Tracing

### Disable Session Tracing by session\_id

```
SELECT inst_id,sid, serial#, program FROM gv$session
       WHERE program LIKE '%LMS%' AND INST_ID=3;

EXEC DBMS_SYSTEM.set_ev(si=>687, se=>1, ev=>10046, le=>0, nm=> '');
EXEC DBMS_SYSTEM.set_sql_trace_in_session(687,1,FALSE);
/* or */
EXEC DBMS_SUPPORT.stop_trace;
/* or single session */
EXEC DBMS_SYSTEM.set_sql_trace_in_session(785,1,FALSE);
```

### Loop to disable all tracing on all processes

```
for i in `cat procs.txt`; do sqlplus '/as sysdba' <<EOF
EXEC DBMS_SYSTEM.set_ev(si=>$i, se=>1, ev=>10046, le=>0, nm=> '');
EXEC DBMS_SYSTEM.set_sql_trace_in_session($i,1,false);
EOF
done
```

### Disable tracing with DBMS\_MONITOR package

```
EXEC DBMS_MONITOR.session_trace_disable;

FOR i IN `cat procs.txt`; do sqlplus '/as sysdba' <<EOF
EXEC DBMS_MONITOR.session_trace_disable(session_id=>$i, serial_num=>1);
EOF
done
```

## Resource Manager

### Activate Resource Manager Plan

```
ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = 'FORCE:MIXED_WORKLOAD_PLAN';
```

**Show top(active) plan**

```
SELECT name, is_top_plan FROM v$rsrc_plan;
```

NAME	IS_TO
-----	-----
DAY_PLAN	TRUE

**Show plans**

```
SELECT plan,STATUS,comments FROM dba_rsrc_plans;
```

**List consumer groups per session**

```
SELECT sid,serial#,username,resource_consumer_group FROM v$session;
```

**Show sessions and waits**

```
SELECT name, active_sessions, queue_length,
       consumed_cpu_time, cpu_waits, cpu_wait_time FROM v$rsrc_consumer_group;
```

```
SELECT se.sid sess_id, co.name consumer_group,
       se.state, se.consumed_cpu_time cpu_time, se.cpu_wait_time, se.queued_time
FROM v$rsrc_session_info se, v$rsrc_consumer_group co
WHERE se.current_consumer_group_id = co.id;
```

**Show historical CPU waits**

```
SELECT SEQUENCE# seq, name, cpu_wait_time, cpu_waits,
       consumed_cpu_time FROM v$rsrc_cons_group_history;
```

**Show resource consumer groups privileges**

```
SELECT * FROM dba_rsrc_consumer_group_privs;
```

**Show resource plan directives**

```
SELECT plan, group_or_subplan, cpu_p1, cpu_p2, cpu_p3
       FROM DBA_RSRC_PLAN_DIRECTIVES ORDER BY plan, cpu_p1 DESC, cpu_p2 DESC,
       cpu_p3 DESC;
/* with utilization and parallelism limits */
SELECT plan, group_or_subplan, mgmt_p1, mgmt_p2, mgmt_p3, mgmt_p4,
```

```
parallel_degree_limit_p1, max_utilization_limit, parallel_queue_timeout FROM
DBA_RSRC_PLAN_DIRECTIVES ORDER BY plan, mgmt_p1 DESC, mgmt_p2 DESC, mgmt_p3
DESC;
```

## Parallelism

### Show/Alter parallelism

```
SHOW parameter PARALLEL_DEGREE_POLICY;
ALTER system SET parallel_degree_policy=auto;

SHOW parameter PARALLEL_THREADS_PER_CPU;
```

### Show parallel query statistics

```
SELECT * FROM v$pdq_sesstat;
SELECT * FROM v$pdq_sysstat;
SELECT * FROM v$px_process_sysstat;
```

### Show current parallel query session detail

```
SELECT slave_name, sessions, idle_time_cur, cpu_secs_total
FROM v$pdq_slave ORDER BY slave_name;
```

```
<code>
== SHOW global parallelism details ==
<code sql>
SELECT name, VALUE
FROM gv$sysstat WHERE
UPPER(NAME) LIKE '%PARALLEL OPERATIONS%' OR
UPPER(NAME) LIKE '%PARALLELIZED%' OR
UPPER(NAME) LIKE '%PX%';
```

### SQL statement to display the most recent parallel query execution details

```
SELECT
  tq_id,
  server_type,
  process,
  num_rows
FROM
  v$pdq_tqstat
WHERE
  dfo_number =
  (SELECT MAX(dfo_number)
```

```
FROM
    v$pg_tqstat)
ORDER BY
    tq_id,
    decode (substr(server_type,1,4),
        'Prod', 0, 'Cons', 1, 3)
;
```

### Show statistics on parallel execution operations

```
SELECT dfo_number, tq_id, server_type, process, num_rows
FROM V$PQ_TQSTAT ORDER BY dfo_number DESC, tq_id, server_type, process;
```

## ASM

[How to Replace a Hard Drive in an Exadata Storage Server \(Hard Failure\) \(Doc ID 1386147.1\)](#)  
Identify and add failed Exadata disks back into ASM

### Check rebalancing operations

```
SQL>
SELECT * FROM gv$asm_operation;
```

### Check the number of disks per group

For Exadata there should be 12 DATA, 10 RECO, 12 DBFS (MODE\_STATUS=ONLINE or MOUNT\_STATUS=CACHED).

```
SQL> SELECT group_number, failgroup, mode_status, COUNT(*) FROM v$asm_disk
GROUP BY group_number, failgroup, mode_status;
```

### Group 0 are problem disks

```
SQL> SELECT path, header_status FROM v$asm_disk WHERE group_number=0;
```

### Get more detail

```
SQL> SELECT group_number, path, header_status, mount_status, name FROM
v$asm_disk WHERE path LIKE '%cell16man%';
```

## Add disk to diskgroups

```
SQL> ALTER diskgroup dbfs_dg ADD disk
'o/192.168.3.143/DBFS_DG_CD_04_m1cel03man' name DBFS_DG_CD_04_m1CEL03MAN
force rebalance nowait
SQL> ALTER diskgroup reco_m1 ADD disk
'o/192.168.3.143/RECO_M1_CD_04_m1cel03man' name RECO_HEM1_CD_04_M1CEL03MAN
```

## Show disks by group

```
SELECT d.name, r.group_number, r.inst_id, r.operation, r.state, r.power FROM
gv$asm_operation r, v$asm_diskgroup d WHERE r.group_number = d.group_number;
```

## Find offline ASM disks

```
SELECT group_number, name, path, mode_status, mount_status, failgroup FROM
v$asm_disk WHERE mode_status='OFFLINE';
```

## Show disks per diskgroup

```
SELECT NVL(a.name, '[CANDIDATE]') disk_group_name, b.path disk_file_path,
b.name disk_file_name, b.failgroup disk_file_fail_group FROM v$asm_diskgroup
a RIGHT OUTER JOIN v$asm_disk b USING (group_number) ORDER BY a.name;
```

## Drop old disk

```
ALTER diskgroup DATA_HEM1 DROP disk '_DROPPED_0074_DATA_HEM1';
```

## Change ASM rebalance power (priority)

```
ALTER diskgroup <disk GROUP name> rebalance POWER 1 nowait
```

## ASMCMD

### Moving multiple files inside ASM

```
for i in $(asmcmd ls +RECO1/EIWODSD/ARCHIVELOG/RESTORE2); do
asmcmd cp +RECO1/EIWODSD/ARCHIVELOG/RESTORE2/$i
+DATA1/TMPARCH/RESTORE2;
asmcmd rm +RECO1/EIWODSD/ARCHIVELOG/RESTORE2/$i;
```

done

## Licensing

<https://www.oracle.com/assets/processor-core-factor-table-070634.pdf>

[https://docs.oracle.com/cd/E80920\\_01/DBMLI/exadata-capacity-on-demand.htm#DBMLI147](https://docs.oracle.com/cd/E80920_01/DBMLI/exadata-capacity-on-demand.htm#DBMLI147)

## Oracle Enterprise Manager

Have found that OEM 12cv5 Agent misbehaves - high CPU, memory leaks.

Here's how to resolve:

1. As Oracle user, gracefully shutdown by killing any leftover processes

```
$agent_inst/bin/emctl stop agent  
$ps -ef | grep java | grep '<agent based dir>'  
$ps -ef | grep perl
```

Kill if any JAVA /PERL process id is active from AGENT HOME directory.

```
$kill -9 <Process id>
```

2. Move/Delete old files from /agent\_inst/sysman/emd/state/\* to a new directory

Example:

```
$mv /u02/agent_inst/sysman/emd/state/* /u02/tmp/
```

3. Execute agent clearstate agent

```
$agent_inst/bin/emctl clearstate agent
```

4. Startup agent

```
$agent_inst/bin/emctl status agent  
$agent_inst/bin/emctl start agent
```

## RMAN

### Register database in rman catalog

```
CONNECT target /  
CONNECT catalog rmanuser/rmanpass@catalogserver.co.za ;  
REGISTER DATABASE;
```

## REPORT SCHEMA;

### Show progress of running backup

[rman\\_backup\\_progress.sql](#)

```

col dbsize_mbytes      FOR 99,999,990.00 justify RIGHT head "DBSIZE_MB"
col input_mbytes       FOR 99,999,990.00 justify RIGHT head "READ_MB"
col output_mbytes      FOR 99,999,990.00 justify RIGHT head
"WRITTEN_MB"
col output_device_type FOR a10          justify LEFT  head "DEVICE"
col complete           FOR 990.00       justify RIGHT head "COMPLETE
%"
col compression        FOR 990.00       justify RIGHT head "COMPRESS|%
ORIG"
col est_complete       FOR a20          head "ESTIMATED COMPLETION"
col recid              FOR 9999999     head "ID"

SELECT recid
       , output_device_type
       , dbsize_mbytes
       , input_bytes/1024/1024 input_mbytes
       , output_bytes/1024/1024 output_mbytes
       , (output_bytes/input_bytes*100) compression
       , (mbytes_processed/dbsize_mbytes*100) complete
       , to_char(start_time + (sysdate-
start_time)/(mbytes_processed/dbsize_mbytes), 'DD-MON-YYYY HH24:MI:SS')
est_complete
  FROM v$rman_status rs
       , (SELECT SUM(bytes)/1024/1024 dbsize_mbytes FROM v$datafile)
 WHERE STATUS='RUNNING'
       AND output_device_type IS NOT NULL
/

```

### Block Change Tracking

Can be seen by existence of Oracle ctwr (change tracker writer) process

```

[root@server ~]# ps -ef | grep ctwr
oracle      72084      1  0 15:43 ?          00:00:04 ora_ctwr_dbnamed1

```

Used by RMAN for a change tracking index for incremental backups.

For large 11.2.0.4 databases this can cause backups to hang.

The following queries can be used to tune this feature.

```

/* Status of feature */

```

```

SELECT STATUS FROM V$BLOCK_CHANGE_TRACKING;
/* List the BCT filename */
SELECT filename,SIZE FROM V$BLOCK_CHANGE_TRACKING;
/* Disable change tracking altogether */
ALTER DATABASE DISABLE BLOCK CHANGE TRACKING;

/* Show memory allocation for BCT (this could be too small)*/
SELECT * FROM v$sgastat WHERE name LIKE 'CTWR%';
/* Recommended size for change tracking memory buffer */
SELECT dba_buffer_count_public*dba_entry_count_public*dba_entry_size FROM
x$krstat;
/* Set the undocumented change tracking buffer size (part of the SGA) */
ALTER SYSTEM SET "_bct_public_dba_buffer_size"= 33562624;

```

## ACFS

Cluster mountable Filesystem within ASM

### Create ACFS Command

Either created manually or via the `*asmca*` tool.

```
/sbin/mkfs -t acfs /dev/asm/<acfs_volume>
```

Following commands should be run as privileged user

```

/u01/app/12.1.0.2/grid/bin/srvctl add filesystem -d /dev/asm/<acfs_volume> -
m /u01/mount/point -u oracle -fstype ACFS -autostart ALWAYS

/u01/mount/point start filesystem -d /dev/asm/<acfs_volume>

chown oracle:oinstall /u01/mount/point

chmod 775 /u01/mount/point

```

## DBMS SCHEDULER

### List scheduled jobs and if active

```

SELECT owner, job_name, enabled FROM DBA_SCHEDULER_JOBS WHERE owner =
'OWNER_USER';

```

### Identify job action

```

SELECT owner, job_name, job_action, enabled FROM DBA_SCHEDULER_JOBS WHERE

```

```
job_name = 'JOB_NAME';
```

### Stop a running job

```
EXEC DBMS_SCHEDULER.STOP_JOB(job_name => 'OWNER.JOB_NAME', force => TRUE);
```

### Disable a job (once stopped)

```
EXEC DBMS_SCHEDULER.disable('OWNER.JOB_NAME');
```

## Automatic Database Diagnostic Monitor (ADDM)

The CONTROL\_MANAGEMENT\_PACK\_ACCESS has three settings:

NONE - Oracle Diagnostics Pack and Oracle Tuning Pack is disabled on the database server, is strongly discouraged by Oracle, but you must if you haven't purchased a license.

DIAGNOSTIC - Oracle Diagnostics Pack is enabled on the database server

DIAGNOSTIC+TUNING - Both Oracle Diagnostics Pack and Oracle Tuning Pack are enabled on the database server

```
/* Display current value */
```

```
SQL> SHOW parameter control_management_pack_access
```

NAME	TYPE	VALUE
control_management_pack_access	string	DIAGNOSTIC+TUNING

```
SQL> SELECT COUNT(*) FROM V$ACTIVE_SESSION_HISTORY;
```

```

COUNT(*)
-----
        64 <-----

```

```
/* Disable feature */
```

```
ALTER system SET control_management_pack_access='NONE';
```

```
SELECT COUNT(*) FROM V$ACTIVE_SESSION_HISTORY;
```

```

COUNT(*)
-----
         0 <-----

```

The `STATISTICS_LEVEL` initialization parameter has been around for quite some time and determines the level of database and operating system statistics that will be collected. Setting this parameter to `TYPICAL` or `ALL` will enable automatic database diagnostic monitoring, which is what we want. The three valid values are as follows:

`TYPICAL` - the default setting will collect all the major statistics that Oracle deems necessary for database self-management while providing the best overall performance. Typically, no pun intended, `TYPICAL` is usually adequate for most environments.

`ALL` - in addition to the `TYPICAL` collections, `ALL` will collect additional statistics such as timed operating system (OS) statistics and plan execution statistics.

`BASIC` - disables many of the important statistics and is highly discouraged.

```
SHOW parameters STATISTICS_LEVEL;
```

From:

<https://wiki.dewberry.co.za/> - **Shaun's Wiki**

Permanent link:

<https://wiki.dewberry.co.za/doku.php?id=oracle&rev=1582279302>

Last update: **2020/02/21 10:01**

