Oracle DBA Checklist

Version 1.0

Revised: 26-Jan-1999

Authors: Thomas B. Cox, with Christine Choi

Purpose: This document gives details for performing daily, weekly, and monthly checks of the status of one or more Oracle databases. All SQL and PL/SQL code for the listed checks can be found in the appendix.

Change Note: This version is generic and must be customized for the local site

Support Information:

Help Desk:		<phone number>
Physical DBA:	<name>	<phone number>
Application DBA:	<name>	<phone number>
Oracle Support:	CSI: <number>	<phone number>

Acknowledgements:

This paper was inspired by the work of David Cook (see References), and has been largely fleshed out by Christine Choi of Hewlett-Packard (Components Group), San Jose, California. I am grateful to both for their contributions to this document.

Please send your corrections, suggestions, and feedback to me at the address below, with your return address so I may credit your contribution. Thank you.

-Thomas B. Cox
tbcox@worldnet.att.net

Index

I. Daily Procedures

A. Verify all instances are up

Make sure the database is available. Log into each instance and run daily reports or test scripts. Some sites may wish to automate this.
Optional implementation: use Oracle Enterprise Manager's 'probe' event.

B. Look for any new alert log entries

- Connect to each managed systems.
- Use 'telnet' or comparable program.
- For each managed instance, go to the background dump destination, usually $ORACLE_BASE/<SID>/bdump. Make sure to look under each managed database's SID.
- At the prompt, use the Unix 'tail' command to see the alert_<SID>.log, or otherwise examine the most recent entries in the file.
- If any ORA-errors have appeared since the previous time you looked, note them in the Database Recovery Log and investigate each one. The recovery log is in <file>.

C. Verify DBSNMP is running

1. Log on to each managed machine to check for the 'dbsnmp' process.

   For Unix: at the command line, type ps –ef | grep dbsnmp. There should be two dbsnmp processes running. If not, restart DBSNMP.

D. Verify success of database backup

E. Verify success of database archiving to tape

F. Verify enough resources for acceptable performance

1. Verify free space in tablespaces.

   For each instance, verify that enough free space exists in each tablespace to handle the day's expected growth. As of <date>, the minimum free space for <repeat for each tablespace>: [< tablespace > is < amount >]. When incoming data is stable, and average daily growth can be calculated, then the minimum free space should be at least <time to order, get, and install more disks> days' data growth.

   a) Go to each instance, run free.sql to check free mb in tablespaces.

   Compare to the minimum free MB for that tablespace. Note any low-space conditions and correct.

b) Go to each instance, run space.sql to check percentage free in tablespaces.

Compare to the minimum percent free for that tablespace. Note any low-space conditions and correct.

2. Verify rollback segment.

   Status should be ONLINE, not OFFLINE or FULL, except in some cases you may have a special rollback segment for large batch jobs whose normal status is OFFLINE.

   a) Optional: each database may have a list of rollback segment names and their expected statuses.

   b) For current status of each ONLINE or FULL rollback segment (by ID not by name), query on V$ROLLSTAT.

   c) For storage parameters and names of ALL rollback segment, query on DBA_ROLLBACK_SEGS. That view's STATUS field is less accurate than V$ROLLSTAT, however, as it lacks the PENDING OFFLINE and FULL statuses, showing these as OFFLINE and ONLINE respectively.

3. Identify bad growth projections.

   Look for segments in the database that are running out of resources (e.g. extents) or growing at an excessive rate. The storage parameters of these segments may need to be adjusted. For example, if any object reached 200 as the number of current extents, upgrade the max_extents to unlimited.

   a) To gather daily sizing information, run daily_01.sql

   b) To check current extents, run nr_extents.sql

   c) Query current table sizing information

   d) Query current index sizing information

   e) Query growth trends

4. Identify space-bound objects.

   Space-bound objects' next_extents are bigger than the largest extent that the tablespace can offer. Space-bound objects can harm database performance. If we get such object, first need to investigate the situation. Then we can use ALTER TABLESPACE <tablespace> COALESCE. Or add another datafile.

a) Run spacebound.sql. If all is well, zero rows will be returned.

5. Processes to review contention for CPU, memory, network or disk resources.

a) To check CPU utilization, go to x:\web\phase2\default.htm =>system metrics=>CPU utilization page. 400 is the maximum CPU utilization because there are 4 CPUs on phxdev and phxprd machine. We need to investigate if CPU utilization keeps above 350 for a while.

G. Read DBA manuals for one hour

II. Weekly Procedures

A. Look for objects that break rules

For each object-creation policy (naming convention, storage parameters, etc.) have an automated check to verify that the policy is being followed.

1. Every object in a given tablespace should have the exact same size for NEXT_EXTENT, which should match the tablespace default for NEXT_EXTENT. As of 12/14/98, default NEXT_EXTENT for DATAHI is 1 gig (1048576 bytes), DATALO is 500 mb (524288 bytes), and INDEXES is 256 mb (262144 bytes).

    a) To check settings for NEXT_EXTENT, run nextext.sql.

    b) To check existing extents, run existext.sql

2. All tables should have unique primary keys.

    a) To check missing PK, run no_pk.sql.

    b) To check disabled PK, run disPK.sql.

    c) All primary key indexes should be unique. Run nonuPK.sql to check.

3. All indexes should use INDEXES tablespace. Run mkrebuild_idx.sql.

4. Schemas should look identical between environments, especially test and production.

    a) To check data type consistency, run datatype.sql.

    b) To check other object consistency, run obj_coord.sql.

B. Look for security policy violations

*C. Look in SQL*Net logs for errors, issues*

1. Client side logs

2. Server side logs

D. Archive all Alert Logs to history

E. Visit home pages of key vendors

1. Andersen Consulting
   http://www.ac.com/index.html

2. Oracle Corporation
   http://www.oracle.com
   http://technet.oracle.com
   http://www.oracle.com/support
   http://www.oramag.com

3. Quest Software
   http://www.quests.com

4. IBM SP2
   http://www.rs6000.ibm.com/sp.html

5. Sun Microsystems
   http://www.sun.com

III. Monthly Procedures

A. Look for Harmful Growth Rates

1. Review changes in segment growth when compared to previous reports to identify segments with a harmful growth rate.

B. Review Tuning Opportunities

1. Review common Oracle tuning points such as cache hit ratio, latch contention, and other points dealing with memory management. Compare with past reports to identify harmful trends or determine impact of recent tuning adjustments.

C. Look for I/O Contention

1. Review database file activity. Compare to past output to identify trends that could lead to possible contention.

D. Review Fragmentation

1. Investigate fragmentation (e.g. row chaining, etc.).

E. Project Performance into the Future

1. Compare reports on CPU, memory, network, and disk utilization from both Oracle and the operating system to identify trends that could lead to contention for any one of these resources in the near future.

2. Compare performance trends to Service Level Agreement to see when the system will go out of bounds

F. Perform Tuning and Maintenance

1. Make the adjustment necessary to avoid the contention for system resources. This may include scheduled down time or request for additional resources.

IV. Appendix

A. Daily Procedures

1. Free.sql

```
--
-- free.sql
--
--   To verify free space in tablespaces
--   Minimum amount of free space
--   document your thresholds:
--   <tablespace_name> = <amount> m
--

SELECT  tablespace_name, sum ( blocks ) as free_blk , trunc ( sum ( bytes ) /
(1024*1024) ) as free_m
, max ( bytes ) / (1024) as big_chunk_k, count (*) as num_chunks
FROM dba_free_space
GROUP BY tablespace_name
```

2. Space.sql

```
--
-- space.sql
--
-- To check free, pct_free, and allocated space within a tablespace
--
-- 11/24/98

SELECT tablespace_name, max_blocks, count_blocks, sum_free_blocks
     , to_char(100*sum_free_blocks/sum_alloc_blocks, '99.99') || '%'
       AS pct_free
FROM ( SELECT tablespace_name
            , sum(blocks) AS sum_alloc_blocks
       FROM dba_data_files
       GROUP BY tablespace_name
     )
   , ( SELECT tablespace_name AS fs_ts_name
            , max(blocks) AS max_blocks
            , count(blocks) AS count_blocks
            , sum(blocks) AS sum_free_blocks
               FROM dba_free_space
               GROUP BY tablespace_name )
WHERE tablespace_name = fs_ts_name
```

3. Daily_01.sql

```
--
-- daily_01.sql
--
-- To analyze tables and indexes
--
```

```
-- 11/30/98

BEGIN
   dbms_utility.analyze_schema ( '&OWNER', 'ESTIMATE', NULL, 5 ) ;
END ;
/
```

4. nr_extents.sql

```
--
-- nr_extents.sql
--
-- To find out any object reaching <threshold>
-- extents, and manually upgrade it to allow unlimited
-- max_extents (thus only objects we *expect* to be big
-- are allowed to become big)
--
-- 11/30/98

SELECT e.owner, e.segment_type , e.segment_name , count(*) as nr_extents ,
s.max_extents
, to_char ( sum ( e.bytes ) / ( 1024 * 1024 ) , '999,999.90') as MB
FROM dba_extents@phxdst  e , dba_segments@phxdst s
WHERE e.segment_name = s.segment_name
GROUP BY e.owner, e.segment_type , e.segment_name , s.max_extents
HAVING count(*) > &THRESHOLD
    OR ( ( s.max_extents - count(*) ) < &&THRESHOLD )
ORDER BY count(*) desc
```

5. spacebound.sql

```
--
-- spacebound.sql
--
-- To identify space-bound objects.  If all is well, no rows are returned.
-- If any space-bound objects are found, look at value of NEXT extent
-- size to figure out what happened.
-- Then use coalesce (alter tablespace <foo> coalesce;).
-- Lastly, add another datafile to the tablespace if needed.
--
-- 11/30/98

SELECT a.table_name, a.next_extent, a.tablespace_name
FROM all_tables a,
    ( SELECT tablespace_name, max(bytes) as big_chunk
      FROM dba_free_space
      GROUP BY tablespace_name ) f
WHERE f.tablespace_name = a.tablespace_name
               AND a.next_extent > f.big_chunk
```

B. Weekly Procedures

1. nextext.sql

```
--
-- nextext.sql
--
-- To find tables that don't match the tablespace default for NEXT extent.
-- The implicit rule here is that every table in a given tablespace should
-- use the exact same value for NEXT, which should also be the tablespace's
-- default value for NEXT.
--
-- This tells us what the setting for NEXT is for these objects today.
--
-- 11/30/98

SELECT segment_name, segment_type, dt.tablespace_name, ds.next_extent
FROM dba_tablespaces dt, dba_segments ds
WHERE dt.tablespace_name = ds.tablespace_name
  AND dt.next_extent !=ds.next_extent
  AND ds.owner = '&OWNER'
```

2. existext.sql

```
--
-- existext.sql
--
-- To check existing extents
--
-- This tells us what the setting for NEXT was for these objects at the
-- time the extent was allocated.  If this report shows a lot of different
-- sized extents, your free space is likely to become fragmented.  If so,
-- this tablespace is a candidate for reorganizing.
--
-- 12/15/98

SELECT count(*), segment_name, segment_type, dt.tablespace_name
FROM dba_tablespaces dt, dba_extents dx
WHERE dt.tablespace_name = dx.tablespace_name
  AND dt.next_extent != dx.byte  AND dx.owner = '&OWNER'
GROUP BY segment_name, segment_type, dt.tablespace_name
```

3. No_pk.sql

```
--
-- To find tables without PK constraint
--
-- 11/2/98

SELECT table_name
FROM all_tables
WHERE owner = '&OWNER'
MINUS
```

```
SELECT table_name
FROM all_constraints
WHERE owner = '&&OWNER'
AND constraint_type = 'P'
```

4. disPK.sql

```
--
-- disPK.sql
--
-- To find out which primary keys are disabled
--
-- 11/30/98

SELECT owner, constraint_name, table_name, status
FROM all_constraints
WHERE owner = '&OWNER' AND status = 'DISABLED’ AND constraint_type = 'P'
```

5. nonuPK.sql

```
--
--  nonuPK.sql
--
-- To find tables with nonunique XPKs
--
-- 11/2/98


SELECT index_name, table_name, uniqueness
FROM all_indexes@PHXDSD
WHERE index_name like '&PKNAME%'
  AND owner = '&OWNER' AND uniqueness = 'NONUNIQUE'
/
```

6. mkrebuild_idx.sql

```
--
--  mkrebuild_idx.sql
--
-- Rebuild indexes to have correct storage parameters
--
-- 11/2/98

SELECT 'alter index ' || index_name || ' rebuild '
     , 'tablespace INDEXES storage ( initial 256 K next 256 K ) ; '
FROM all_indexes
WHERE ( tablespace_name != 'INDEXES'
        OR next_extent != ( 256 * 1024 )
      )
  AND owner = '&OWNER'
/
```

7. datatype.sql

```
--
-- datatype.sql
```

```
--
-- To check datatype consistency between two environments
--
-- 11/30/98


SELECT
   table_name,
   column_name,
   data_type,
   data_length,
   data_precision,
   data_scale,
   nullable
FROM all_tab_columns -- first environment
WHERE owner = '&OWNER'
MINUS
SELECT
   table_name,
   column_name,
   data_type,
   data_length,
   data_precision,
   data_scale,
   nullable
FROM all_tab_columns@&my_db_link  -- second environment
WHERE owner = '&OWNER2'
order by table_name, column_name
```

8. obj_coord.sql

```
--
-- obj_coord.sql
--
-- To find out any difference in objects between two instances
--
-- 12/08/98

SELECT object_name, object_type
FROM user_objects
MINUS
SELECT object_name, object_type
FROM user_objects@&my_db_link
```

V. References

1. Loney, Kevin *Oracle8 DBA Handbook*
2. Cook, David *Database Management from Crisis to Confidence* [http://www.europa.com/~orapub]
3. Cox, Thomas B. *The Database Administration Capability Maturity Model*