Oracle DBA Checklist

Version 1.7 Generic

Revised: 11-Oct-2003

Authors: Thomas B. Cox, version 1.0 written 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.   
  
The latest version of this paper should always be available on the primary author's home page, <http://www.geocities.com/tbcox23>.

Change Notes: 1.1 Typo in 'existext.sql' identified by Steve DeNunzio, fixed

1.2 Typos fixed

1.3 Gnu Public License added; pctincr 0 in rebuild index added

1. Added pointer to newest version on Geocities home page,   
   Fixed pointer to 'orapub' web site,  
   Added nightly checklist and volumetrics
2. fixed typo in III A 1 (‘bytes’ should be ‘kbytes’) identified by Lee Hallin
3. Fixed WHERE clause in NR\_EXTENTS, identified by Evgueni Balabanov
4. Formatting and other small fixes

Support Information (customize for your site):

Help Desk: <phone>

Physical DBA: <name> <phone>

Application DBA: <name> <phone>

Oracle Support: CSI: <#> <phone>

**Acknowledgements:**

This paper was inspired by the work of David Cook (see References), and Version 1.0 was 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. A version of this Checklist was included in the April 2001 book *Oracle DBA Checklists Pocket Reference* by O’Reilly and Associates.

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, <tbcox23@yahoo.com>

Copyright © 1999-2005 Thomas B. Cox, All Rights Reserved.

This document is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This document is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

For a copy of the GNU General Public License write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

**Index**

I. Daily Procedures 3

A. Verify all instances are up 3

B. Look for any new alert log entries 3

C. Verify DBSNMP is running 3

D. Verify success of database backup 3

E. Verify success of database archiving to tape 3

F. Verify enough resources for acceptable performance 3

G. Copy Archived Logs to Standby Database and Roll Forward 5

H. Read DBA manuals for one hour 5

II. Nightly Procedures 6

A. Collect volumetric data 6

III. Weekly Procedures 7

A. Look for objects that break rules 7

B. Look for security policy violations 7

C. Look in SQL\*Net logs for errors, issues 7

D. Archive all Alert Logs to history 7

E. Visit home pages of key vendors 8

IV. Monthly Procedures 9

A. Look for Harmful Growth Rates 9

B. Review Tuning Opportunities 9

C. Look for I/O Contention 9

D. Review Fragmentation 9

E. Project Performance into the Future 9

F. Perform Tuning and Maintenance 9

V. Appendix 10

A. Daily Procedures 10

B. Nightly Procedures 12

C. Weekly Procedures 14

VI. References 17

# I. II. 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 system.
* 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. (Some sites have this disabled on purpose; if this is the case, remove this item from your list, or change it to "verify that DBSNMP is NOT running".)

## 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, AND it's an object that is supposed to get large, upgrade the max\_extents to unlimited.

#### a) To gather daily sizing information, run analyze5pct.sql. If you are collecting nightly volumetrics, skip this step.

#### 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 operation. 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. Copy Archived Logs to Standby Database and Roll Forward

If you have a Standby Database, copy the appropriate Archived Logs to the expected location on the standby machine and apply those logs (roll forward the changes) to the standby database. This keeps the standby database up-to-date.

The copying of logs, the applying of them, or both, can in some cases be automated. If you have automated them, then your daily task should be to confirm that this happened correctly each day.

## H. Read DBA manuals for one hour

Nothing is more valuable in the long run than that the DBA be as widely experienced, and as widely read, as possible. Readings should include DBA manuals, trade journals, and possibly newsgroups or mailing lists.

# III. IV. Nightly Procedures

Most production databases (and many development and test databases) will benefit from having certain nightly batch processes run.

## A. Collect volumetric data

This example collects table row counts. This can easily be extended to other objects such as indexes, and other data such as average row sizes.

### 1. Analyze Schemas and Collect Data.

The idea here is to use the more time consuming and more accurate ANALYZE COMPUTE command and save the results, which show up in the data dictionary, to a more permanent store.

#### a) If you havent' yet, create the volumetrics table with mk\_volfact.sql

#### b) To gather nightly sizing information, run analyze\_comp.sql.

#### c) To collect the resulting statistics, run pop\_vol.sql

#### d) Examine the data at your leisure, probably weekly or monthly.

I use MS Excel and an ODBC connection to examine and graph data growth.

# V. VI. 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 kbytes), DATALO is 500 mb (524288 kbytes), and INDEXES is 256 mb (262144 kbytes).

#### 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.

#### c) Better yet, use a tool like Quest Software's Schema Manager.

## 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. Oracle Corporation

http://www.oracle.com  
http://technet.oracle.com  
http://www.oracle.com/support  
http://www.oramag.com

### 2. Quest Software

http://www.quests.com

### 3. Sun Microsystems

http://www.sun.com

# VII. VIII. 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 adjustments necessary to avoid contention for system resources. This may include scheduled down time or request for additional resources.

# IX. X. 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, largest\_free\_chunk

, nr\_free\_chunks, sum\_alloc\_blocks, sum\_free\_blocks

, to\_char(100\*sum\_free\_blocks/sum\_alloc\_blocks, '09.99') || '%'

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 largest\_free\_chunk

, count(blocks) AS nr\_free\_chunks

, sum(blocks) AS sum\_free\_blocks

FROM dba\_free\_space

GROUP BY tablespace\_name )

WHERE tablespace\_name = fs\_ts\_name

### 3. analyze5pct.sql

--

-- analyze5pct.sql

--

-- To analyze tables and indexes quickly, using a 5% sample size

-- (do not use this script if you are performing the overnight

-- collection of volumetric data)

--

-- 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 e , dba\_segments s

WHERE e.segment\_name = s.segment\_name

AND e.owner = s.owner -- 2001.10.19 Evgueni B. fix

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. Nightly Procedures

### 1. mk\_volfact.sql

--

-- mk\_volfact.sql (only run this once to set it up; do not run it nightly!)

--

-- -- Table UTL\_VOL\_FACTS

CREATE TABLE utl\_vol\_facts

(

table\_name VARCHAR2(30),

num\_rows NUMBER,

meas\_dt DATE

)

TABLESPACE platab

STORAGE (

INITIAL 128k

NEXT 128k

PCTINCREASE 0

MINEXTENTS 1

MAXEXTENTS unlimited

)

/

-- Public Synonym

CREATE PUBLIC SYNONYM utl\_vol\_facts FOR &OWNER..utl\_vol\_facts

/

-- Grants for UTL\_VOL\_FACTS

GRANT SELECT ON utl\_vol\_facts TO public

/

### 2. analyze\_comp.sql

--

-- analyze\_comp.sql

--

BEGIN

sys.dbms\_utility.analyze\_schema ( '&OWNER','COMPUTE');

END ;

/

### 3. pop\_vol.sql

--

-- pop\_vol.sql

--

insert into utl\_vol\_facts

select table\_name

, NVL ( num\_rows, 0) as num\_rows

, trunc ( last\_analyzed ) as meas\_dt

from all\_tables -- or just user\_tables

where owner in ('&OWNER') -- or a comma-separated list of owners

/

commit

/

## C. 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, ds.next\_extent as Actual\_Next

, dt.tablespace\_name, dt.next\_extent as Default\_Next

FROM dba\_tablespaces dt, dba\_segments ds

WHERE dt.tablespace\_name = ds.tablespace\_name

AND dt.next\_extent !=ds.next\_extent

AND ds.owner = UPPER ( '&OWNER' )

ORDER BY tablespace\_name, segment\_type, segment\_name

### 2. existext.sql

--

-- existext.sql

--

-- To check existing extents

--

-- This tells us how many of each object's extents differ in size from

-- the tablespace's default size. 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 segment\_name, segment\_type

, count(\*) as nr\_exts

, sum ( DECODE ( dx.bytes,dt.next\_extent,0,1) ) as nr\_illsized\_exts

, dt.tablespace\_name, dt.next\_extent as dflt\_ext\_size

FROM dba\_tablespaces dt, dba\_extents dx

WHERE dt.tablespace\_name = dx.tablespace\_name

AND dx.owner = '&OWNER'

GROUP BY segment\_name, segment\_type, dt.tablespace\_name, dt.next\_extent

### 3. No\_pk.sql

--

-- 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 PK indexes. Requires that PK names

-- follow a naming convention. An alternative query follows that

-- does not have this requirement, but runs more slowly.

--

-- 11/2/98

SELECT index\_name, table\_name, uniqueness

FROM all\_indexes

WHERE index\_name like '&PKNAME%'

AND owner = '&OWNER' AND uniqueness = 'NONUNIQUE'

SELECT c.constraint\_name, i.tablespace\_name, i.uniqueness

FROM all\_constraints c , all\_indexes i

WHERE c.owner = UPPER ( '&OWNER' ) AND i.uniqueness = 'NONUNIQUE'

AND c.constraint\_type = 'P' AND i.index\_name = c.constraint\_name

### 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 pctincrease 0 ) ; '

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

# XI. References

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