Top DBA Shell Scripts for Monitoring the Database

In this article, I would like to focus on the DBA's daily responsibilities for monitoring Oracle databases, and share some tips and techniques on how DBAs can turn their manual, reactive monitoring activities into a set of proactive shell scripts. First I'll review some commonly used Unix commands by DBAs and explain the Unix Cron jobs that are used as part of the scheduling mechanism to execute DBA scripts. Then I will cover these eight important scripts for monitoring the Oracle database:

- Check instance availability
- Check listener availability
- Check alert log files for error messages
- Clean up old log files before log destination gets filled
- Analyze tables and indexes for better performance
- Check tablespace usage

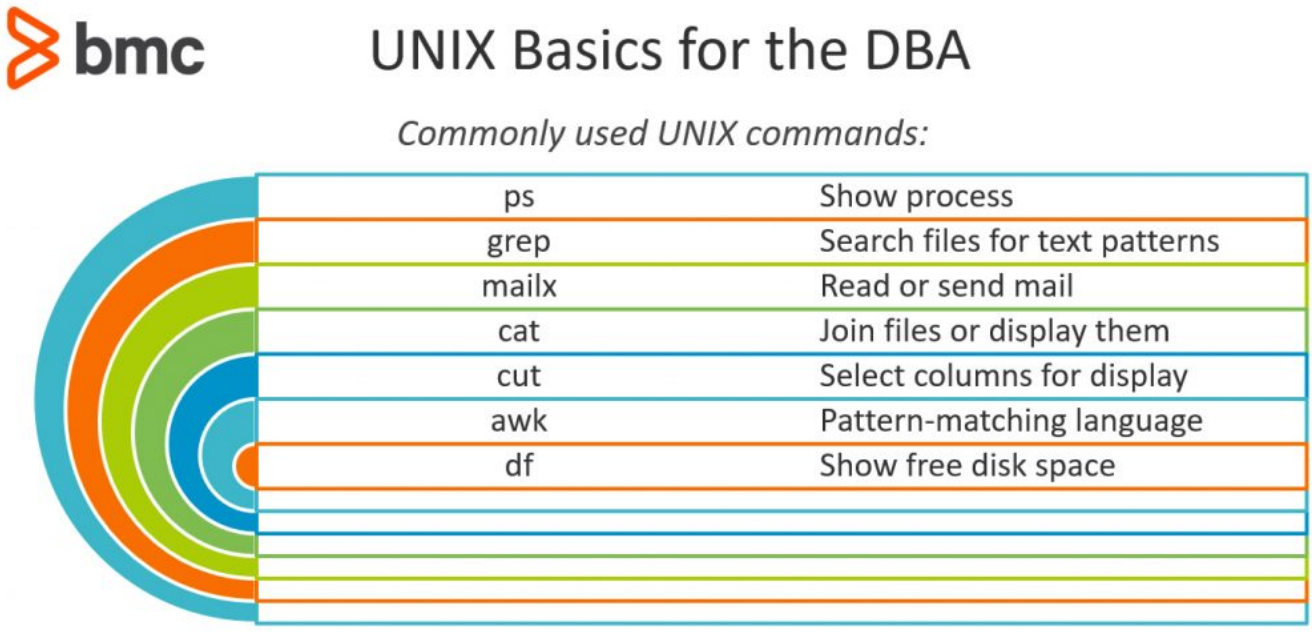
- Find out invalid objects
- Monitor users and transactions

UNIX Basics for the DBA

Basic UNIX Command

The following is a list of commonly used Unix commands:

- **ps** – Show process
- **grep** – Search files for text patterns
- **mailx** – Read or send mail
- **cat** – Join files or display them
- **cut** – Select columns for display
- **awk** – Pattern-matching language
- **df** – Show free disk space



Here are some examples of how the DBA uses these commands:

- List available instances on a server:

```
$ ps -ef | grep smon
oracle 21832     1  0   Feb 24 ?       19:05 ora_smon_oradb1
```

```
oracle   898     1  0   Feb 15 ?        0:00 ora_smon_oradb2
dliu 25199 19038  0 10:48:57 pts/6    0:00 grep smon
oracle 27798     1  0 05:43:54 ?        0:00 ora_smon_oradb3
oracle 28781     1  0   Mar 03 ?        0:01 ora_smon_oradb4
```

- List available listeners on a server:

```
$ ps -ef | grep listener | grep -v grep
oracle 23879    1  0   Feb 24 ?  33:36 /8.1.7/bin/tnslsnr
listener_db1 -inherit
oracle 27939    1  0 05:44:02 ?  0:00  /8.1.7/bin/tnslsnr
listener_db2 -inherit
oracle 23536    1  0   Feb 12 ?  4:19  /8.1.7/bin/tnslsnr
listener_db3 -inherit
oracle 28891    1  0   Mar 03 ?  0:01  /8.1.7/bin/tnslsnr
listener_db4 -inherit
```

- Find out file system usage for Oracle archive destination:

```
$ df -k | grep oraarch
/dev/vx/dsk/proddg/oraarch 71123968 4754872 65850768   7%
/u09/oraarch
```

- List number of lines in the alert.log file:

```
$ cat alert.log | wc -l
2984
```

- List all Oracle error messages from the alert.log file:

```
$ grep ORA- alert.log
ORA-00600: internal error code, arguments: [kcrrrfswda.1], [],
[], [], [], []
ORA-00600: internal error code, arguments: [1881], [25860496],
[25857716], []
```

CRONTAB Basics

A crontab file is comprised of six fields:

Minute	0-59
Hour	0-23

Day of month	1-31
Month	1 – 12
Day of Week	0 – 6, with 0 = Sunday
Unix Command or Shell Scripts	

- To edit a crontab file, type:

```
Crontab -e
```

- To view a crontab file, type:

```
Crontab -l
```

```
0  4 * * 5       /dba/admin/analyze_table.ksh
30 3  * * 3,6    /dba/admin/hotbackup.ksh /dev/null 2>&1
```

In the example above, the first entry shows that a script to analyze a table runs every Friday at 4:00 a.m. The second entry shows that a script to perform a hot backup runs every Wednesday and Saturday at 3:00 a.m.

The Eight Most Important DBA Shell Scripts for Monitoring the Database

The eight shell scripts provided below cover 90 percent of a DBA's daily monitoring activities. You will need to modify the UNIX environment variables as appropriate.

Check Oracle Instance Availability

The oratab file lists all the databases on a server:

```
$ cat /var/opt/oracle/oratab
#################################################################
#####
##
/var/opt/oracle/oratab
##
#################################################################
```

```
#####
oradb1:/u01/app/oracle/product/8.1.7:Y
oradb2:/u01/app/oracle/product/8.1.7:Y
oradb3:/u01/app/oracle/product/8.1.7:N
oradb4:/u01/app/oracle/product/8.1.7:Y
```

The following script checks all the databases listed in the oratab file, and finds out the status (up or down) of databases:

```
#############################################################
#####
## ckinstance.ksh ##
#############################################################
#####
ORATAB=/var/opt/oracle/oratab
echo "`date`   "
echo  "Oracle Database(s) Status `hostname` :\n"

db=`egrep -i ":Y|:N" $ORATAB | cut -d":" -f1 | grep -v "\#" |
grep -v "\*"`
pslist="`ps -ef | grep pmon`"
for i in $db ; do
echo  "$pslist" | grep  "ora_pmon_$i"  > /dev/null 2>$1
if (( $? )); then
echo "Oracle Instance - $i:       Down"
else
echo "Oracle Instance - $i:       Up"
fi
done
```

Use the following to make sure the script is executable:

```
$ chmod 744 ckinstance.ksh
$ ls -l ckinstance.ksh
-rwxr--r--    1 oracle      dba      657 Mar  5 22:59
ckinstance.ksh*
```

Here is an instance availability report:

```
$ ckinstance.ksh
Mon Mar  4 10:44:12 PST 2002
```

```
Oracle Database(s) Status for DBHOST server:
Oracle Instance - oradb1:   Up
Oracle Instance - oradb2:   Up
Oracle Instance - oradb3:   Down
Oracle Instance - oradb4:   Up
```

Check Oracle Listener’s Availability

A similar script checks for the Oracle listener. If the listener is down, the script will restart the listener:

```
##############################################################
#########
##
cklsnr.sh
    ##
##############################################################
#########
#!/bin/ksh
DBALIST="primary.dba@company.com,
another.dba@company.com";export DBALIST
cd /var/opt/oracle
rm -f lsnr.exist
ps -ef | grep mylsnr | grep -v grep  > lsnr.exist
if [ -s lsnr.exist ]
then
echo
else
echo "Alert" | mailx -s "Listener 'mylsnr' on `hostname` is
down" $DBALIST
TNS_ADMIN=/var/opt/oracle; export TNS_ADMIN
ORACLE_SID=db1; export ORACLE_SID
ORAENV_ASK=NO; export ORAENV_ASK
PATH=$PATH:/bin:/usr/local/bin; export PATH
. oraenv
LD_LIBRARY_PATH=${ORACLE_HOME}/lib;export LD_LIBRARY_PATH
lsnrctl start mylsnr
fi
```

Check Alert Logs (ORA-XXXXX)

Some of the environment variables used by each script can be put into one profile:

```
#########################################################################
## oracle.profile ##
#########################################################################
EDITOR=vi;export EDITOR ORACLE_BASE=/u01/app/oracle; export
ORACLE_BASE ORACLE_HOME=$ORACLE_BASE/product/8.1.7; export
ORACLE_HOME LD_LIBRARY_PATH=$ORACLE_HOME/lib; export
LD_LIBRARY_PATH TNS_ADMIN=/var/opt/oracle;export
TNS_ADMIN NLS_LANG=american; export
NLS_LANG NLS_DATE_FORMAT='Mon DD YYYY HH24:MI:SS'; export
NLS_DATE_FORMAT ORATAB=/var/opt/oracle/oratab;export
ORATAB
PATH=$PATH:$ORACLE_HOME:$ORACLE_HOME/bin:/usr/ccs/bin:/bin:/us
r/bin:/usr/sbin:/
sbin:/usr/openwin/bin:/opt/bin:.; export
PATH                DBALIST="primary.dba@company.com,
another.dba@company.com";export
DBALIST
```

The following script first calls oracle.profile to set up all the environment variables. The script also sends the DBA a warning e-mail if it finds any Oracle errors:

```
#########################################################################
######
##
ckalertlog.sh
##
#########################################################################
######
#!/bin/ksh
. /etc/oracle.profile
for SID in `cat $ORACLE_HOME/sidlist`
do
cd $ORACLE_BASE/admin/$SID/bdump
if [ -f alert_${SID}.log ]
```

```
then
mv alert_${SID}.log alert_work.log
touch alert_${SID}.log
cat alert_work.log >> alert_${SID}.hist
grep ORA- alert_work.log > alert.err
fi
if [ `cat alert.err|wc -l` -gt 0 ]
then
mailx -s "${SID} ORACLE ALERT ERRORS" $DBALIST < alert.err
fi
rm -f alert.err
rm -f alert_work.log
done
```

Clean Up Old Archived Logs

The following script cleans up old archive logs if the log file system reaches 90 percent capacity:

```
$ df -k | grep arch
Filesystem                kbytes   used     avail    capacity
Mounted on
/dev/vx/dsk/proddg/archive 71123968 30210248 40594232   43%
/u08/archive

#################################################################
#########
##
clean_arch.ksh
    ##
#################################################################
#########
#!/bin/ksh
df -k | grep arch > dfk.result
archive_filesystem=`awk  -F" "  '{ print $6 }' dfk.result`
archive_capacity=`awk  -F" "  '{ print $5 }' dfk.result`
if [[ $archive_capacity > 90% ] ]
then
echo "Filesystem ${archive_filesystem} is ${archive_capacity}
filled"
# try one of the following option depend on your need
```

```
find $archive_filesystem -type f -mtime +2 -exec rm -r {} \;
tar
rman
fi
```

Analyze Tables and Indexes (for Better Performance)

Below, I have shown an example of how to pass parameters to a script:

```
##############################################################
######
## analyze_table.sh ##
##############################################################
######
#!/bin/ksh #
input parameter: 1: password # 2: SID if (($#<1)) then echo
"Please enter
'oracle'
user password as the first parameter !" exit 0 fi if (($#<2))
then echo
"Please enter
instance name as the second parameter!" exit 0 fi
```

To execute the script with parameters, type:

```
$ analyze_table.sh manager oradb1
```

The first part of script generates a file analyze.sql, which contains the syntax for analyzing table. The second part of the script analyzes all the tables:

```
##############################################################
#######
## analyze_table.sh ##
##############################################################
#######
sqlplus -s <<!
oracle/$1@$2
set heading off
```

```
set feed off
set pagesize 200
set linesize 100
spool analyze_table.sql
select 'ANALYZE TABLE ' || owner || '.' || segment_name ||
' ESTIMATE STATISTICS SAMPLE 10 PERCENT;'
from dba_segments
where segment_type = 'TABLE'
and owner not in ('SYS', 'SYSTEM');
spool off
exit
!
sqlplus -s <<!
oracle/$1@$2
@./analyze_table.sql
exit
!
```

Here is an example of analyze.sql:

```
$ cat analyze.sql
ANALYZE TABLE HIRWIN.JANUSAGE_SUMMARY ESTIMATE STATISTICS
SAMPLE 10 PERCENT;
ANALYZE TABLE HIRWIN.JANUSER_PROFILE ESTIMATE STATISTICS
SAMPLE 10 PERCENT;
ANALYZE TABLE APPSSYS.HIST_SYSTEM_ACTIVITY ESTIMATE STATISTICS
SAMPLE 10 PERCENT;
ANALYZE TABLE HTOMEH.QUEST_IM_VERSION ESTIMATE STATISTICS
SAMPLE 10 PERCENT;
ANALYZE TABLE JSTENZEL.HIST_SYS_ACT_0615 ESTIMATE STATISTICS
SAMPLE 10 PERCENT;
ANALYZE TABLE JSTENZEL.HISTORY_SYSTEM_0614 ESTIMATE STATISTICS
SAMPLE 10 PERCENT;
ANALYZE TABLE JSTENZEL.CALC_SUMMARY3 ESTIMATE STATISTICS
SAMPLE 10 PERCENT;
ANALYZE TABLE IMON.QUEST_IM_LOCK_TREE ESTIMATE STATISTICS
SAMPLE 10 PERCENT;
ANALYZE TABLE APPSSYS.HIST_USAGE_SUMMARY ESTIMATE STATISTICS
SAMPLE 10 PERCENT;
ANALYZE TABLE PATROL.P$LOCKCONFLICTTX ESTIMATE STATISTICS
SAMPLE 10 PERCENT;
```

Check Tablespace Usage

This script checks for tablespace usage. If tablespace is 10 percent free, it will send an alert e-mail.

```
#################################################################
#######
## ck_tbsp.sh ##
#################################################################
#######
#!/bin/ksh
sqlplus -s <<!
oracle/$1@$2
set feed off
set linesize 100
set pagesize 200
spool tablespace.alert
SELECT F.TABLESPACE_NAME,
TO_CHAR  ((T.TOTAL_SPACE  -  F.FREE_SPACE),'999,999')  "USED
(MB)",
TO_CHAR (F.FREE_SPACE, '999,999') "FREE (MB)",
TO_CHAR (T.TOTAL_SPACE, '999,999') "TOTAL (MB)",
TO_CHAR ((ROUND ((F.FREE_SPACE/T.TOTAL_SPACE)*100)),'999')||'
%' PER_FREE
FROM   (
SELECT       TABLESPACE_NAME,
ROUND (SUM (BLOCKS*(SELECT VALUE/1024
FROM V\$PARAMETER
WHERE NAME = 'db_block_size')/1024)
) FREE_SPACE
FROM DBA_FREE_SPACE
GROUP BY TABLESPACE_NAME
) F,
(
SELECT TABLESPACE_NAME,
ROUND (SUM (BYTES/1048576)) TOTAL_SPACE
FROM DBA_DATA_FILES
GROUP BY TABLESPACE_NAME
) T
WHERE F.TABLESPACE_NAME = T.TABLESPACE_NAME
AND (ROUND ((F.FREE_SPACE/T.TOTAL_SPACE)*100)) < 10;
```

```
spool off
exit
!
if [ `cat tablespace.alert|wc -l` -gt 0 ]
then
cat tablespace.alert -l tablespace.alert > tablespace.tmp
mailx -s "TABLESPACE ALERT for ${2}" $DBALIST < tablespace.tmp
fi
```

An example of the alert mail output is as follows:

```
TABLESPACE_NAME      USED (MB)   FREE (MB)          TOTAL
(MB)             PER_FREE
------------------- --------- ---------- ------------------
------------------
SYSTEM                2,047             203
2,250                  9 %
STBS01                  302              25
327                   8 %
STBS02                  241              11
252                   4 %
STBS03                  233              19
252                   8 %
```

Find Out Invalid Database Objects

The following finds out invalid database objects:

```
#########################################################
#######
## invalid_object_alert.sh ##
#########################################################
#######
#!/bin/ksh
. /etc/oracle.profile
sqlplus -s <<!
oracle/$1@$2
set          feed off
set heading off
column object_name format a30
spool invalid_object.alert
SELECT  OWNER, OBJECT_NAME, OBJECT_TYPE, STATUS
```

```
FROM    DBA_OBJECTS
WHERE   STATUS = 'INVALID'
ORDER BY OWNER, OBJECT_TYPE, OBJECT_NAME;
spool off
exit
!
if [ `cat invalid_object.alert|wc -l` -gt 0 ]
then
mailx  -s  "INVALID  OBJECTS  for  ${2}"  $DBALIST  <
invalid_object.alert
fi
$ cat invalid_object.alert

OWNER        OBJECT_NAME           OBJECT_TYPE          STATUS
-------------------------------------------------------------
--------
HTOMEH       DBMS_SHARED_POOL           PACKAGE BODY
INVALID
HTOMEH       X_$KCBFWAIT                VIEW
INVALID
IMON         IW_MON                     PACKAGE
INVALID
IMON         IW_MON                     PACKAGE BODY
INVALID
IMON         IW_ARCHIVED_LOG            VIEW
INVALID
IMON         IW_FILESTAT                VIEW
INVALID
IMON         IW_SQL_FULL_TEXT           VIEW
INVALID
IMON         IW_SYSTEM_EVENT1           VIEW
INVALID
IMON         IW_SYSTEM_EVENT_CAT        VIEW
INVALID
LBAILEY      CHECK_TABLESPACE_USAGE     PROCEDURE
INVALID
PATROL       P$AUTO_EXTEND_TBSP         VIEW
INVALID
SYS          DBMS_CRYPTO_TOOLKIT        PACKAGE
INVALID
SYS          DBMS_CRYPTO_TOOLKIT        PACKAGE BODY
```

```
INVALID
SYS           UPGRADE_SYSTEM_TYPES_TO_816 PROCEDURE
INVALID
SYS           AQ$_DEQUEUE_HISTORY_T       TYPE
INVALID
SYS           HS_CLASS_CAPS               VIEW
INVALID
SYS           HS_CLASS_DD                 VIEW
INVALID
```

Monitor Users and Transactions (Dead Locks, et al)

This script sends out an alert e-mail if dead lock occurs:

```
#################################################################
#####
## deadlock_alert.sh ##
#################################################################
#####
#!/bin/ksh
. /etc/oracle.profile
sqlplus -s <<!
oracle/$1@$2
set feed off
set heading off
spool deadlock.alert
SELECT   SID, DECODE(BLOCK, 0, 'NO', 'YES' ) BLOCKER,
DECODE(REQUEST, 0, 'NO','YES' ) WAITER
FROM     V$LOCK
WHERE    REQUEST > 0 OR BLOCK > 0
ORDER BY block DESC;
spool off
exit
!
if [ `cat deadlock.alert|wc -l` -gt 0 ]
then
mailx -s "DEADLOCK ALERT for ${2}" $DBALIST < deadlock.alert
fi
```

Conclusion

```
0,20,40 7-17 * * 1-5 /dba/scripts/ckinstance.sh > /dev/null
2>&1
0,20,40 7-17 * * 1-5 /dba/scripts/cklsnr.sh > /dev/null 2>&1
0,20,40 7-17 * * 1-5 /dba/scripts/ckalertlog.sh > /dev/null
2>&1
30          * * * 0-6 /dba/scripts/clean_arch.sh > /dev/null
2>&1
*          5 * * 1,3 /dba/scripts/analyze_table.sh > /dev/null
2>&1
*          5 * * 0-6 /dba/scripts/ck_tbsp.sh > /dev/null 2>&1
*           5 * * 0-6 /dba/scripts/invalid_object_alert.sh >
/dev/null 2>&1
0,20,40  7-17  *  *  1-5  /dba/scripts/deadlock_alert.sh  >
/dev/null  2>&1
```

Now, my DBA friends, you can finally get a bit more uninterrupted sleep at night. You may also have time for more important things like performance tuning.

References

- Unix in a Nutshell, O'Reilly & Associates, Inc.;
- "Using Oracle9i Application Server to Build Your Web-Based Database Monitoring Tool," Daniel T. Liu; Select Magazine – November 2001 Volume 8, No. 1;
- "Net8: A Step-by-Step Setup of Oracle Names Server," Daniel T. Liu; Oracle Open World 2000, Paper#271.

I would also like to acknowledge the assistance of Johnny Wedekind of ADP, Ann Collins, Larry Bailey, Husam Tomeh and Archana Sharma of FARES.