Scripts

APPENDIX

C

Scripts

The following scripts are some of the ones that I've used over the years. None of them is terribly complex so you should be able to understand them without much problem. Most are really just canned queries that save typing, therefore you will *not* find them nicely commented or documented. I've found that many of the ad hoc queries I make I end up needing again, so why not put them in a script? I usually put them on a floppy, take them with me on-site, and add to them as I go.

Feel free to modify these as you need, but make sure you understand them before using them. Remember, never blindly run *anyone* else's script before you have had a chance to read them yourself. Truly complex scripts are available via other sources and I recommend using those when possible instead of "reinventing the wheel" and writing your own. However, I wouldn't trust the integrity of *my* system to some mysterious script I picked up on the Internet or anywhere else.

Above all, I hope these scripts prove to be useful and help spawn ideas for your own scripts.

login.sql

```
REM This file provides custom display settings with SQL*Plus.
REM Have it in the directory from where you start SQL*Plus.

set pagesize 25
col member format a60
col file_name format a60
col tablespace_name format a20
col owner format a15
col object_name format a30
col initial_extent format 999,999,999
col next_extent format 999,999,999
col bytes format 999,999,999,999
col sum(bytes) format 999,999,999,999
select name, created, log_mode from v$database;
show user;
```

show_session_short.sql

```
select s.username, osuser, status,  server as "Connect Type",
to_char(logon_time,'fmHH:MI:SS AM') as "Logon Time",
sid, s.serial#, p.spid as "UNIX Proc"
from v$session s, v$process p
where s.paddr = p.addr
and s.username is not null
order by status, s.username, s.program, logon_time
/
```

show_dba_rollback_segs.sql

```
select segment_name, owner, tablespace_name, initial_extent,
next_extent, min_extents, max_extents,
status, instance_num from dba_rollback_segs
/
```

show_filestat.sql

```
set linesize 180
col tablespace_name format a20
col file_name format a52
col PHYRDS format 999,999,999
col PHYWRTS format 999,999
col PHYBLKRD format 999,999,999
col PHYBLKWRT format 999,999
spool show_filestat.lst
select tablespace_name, file_name, PHYRDS, PHYWRTS, PHYBLKRD, PHYBLKWRT
from v$filestat, dba_data_files
where file_id = file#
order by PHYRDS, PHYWRTS  desc
/
spool off
```


show_index_depth.sql

```
REM B*Tree indexes should not go past 4 levels, performance suffers.
REM Rebuild anything greater than 3, but remember it will lock the table from
dml
REM unless you are using 8i online rebuilds (which take space instead).
REM Also remember to run analyze before running this
REM
col owner format a15
accept user_name1 prompt 'Enter index owner to examine: '
select owner, table_name, index_name, blevel, last_analyzed from dba_indexes
where upper(owner) = upper('&user_name1')
order by blevel
/
set heading off
select 'Note: blevel should not be greater than 3' from dual
/
set heading on
```

show_redo_logs.sql

```
set linesize 180
col member format a50
col bytes format 999,999,999,999
```

```
select v$log.group#, members, member, v$log.status, bytes, archived
from v$log, v$logfile
where v$log.group# = v$logfile.group#;
```

show_rollback_contention.sql

```
set linesize 180
col name format a15
select a.name, b.extents, b.rssize, b.xacts "Active X-actions", b.waits,
b.gets,
optsize, status
from v$rollname a, v$rollstat b
where a.usn = b.usn
/
```

show_segments.sql

```
REM Note the hard coded owner, you will have to fix this for your system
set linesize 180
col tablespace_name format a16
col segment_name format a30
col segment_type format a6
col initial_extent format 9,999,999,999
col next_extent format 9,999,999,999
col bytes format 99,999,999,999
spool verify_import-2000.lst
select tablespace_name, segment_name, segment_type, initial_extent,
next_extent, bytes, extents
from dba_segments
where owner = 'CUSTOMER'
order by tablespace_name, segment_type, segment_name
/
spool off
```

show_tablespaces.sql

```
set linesize 132
set pagesize 65
set heading off
set feedback off
set verify off
col tablespace_name format a30
col file_name format a60
col bytes format 999,999,999,999,999
col status format a15
spool tablespaces.lst
select to_char(sysdate,'MM-DD-YYYY HH:MM')from dual;
set heading on
select tablespace_name, file_name, bytes, status from dba_data_files
```

```
order by tablespace_name, file_name
/
spool off
```

compare_users.sql

```
REM Get two database users and show their roles.
REM
accept user_1 prompt 'Enter the first user: '
accept user_2 prompt 'Enter the second: '
select grantee, granted_role from dba_role_privs where
grantee in (upper('&user_1'),upper('&user_2'))
order by granted_role, grantee
/
```

create_analyze_script.sql

```
REM Note the hard coded owner. You need to modify this or use Dynamic SQL.
set heading off
set feedback off
set linesize 180
set pagesize 32767
spool analyze_customer_tables.sql
select 'analyze table CUSTOMER.' || table_name || ' estimate statistics;'
from dba_tables
where owner = 'CUSTOMER'
/
spool off
set heading on
set feedback on
```


tail-alert

```
# The following is a handy shell script to check the end
# of the alert.log for a database identified by $ORACLE_SID
# I normally run this script several times a day and immediately
# whenever problems are reported.
# I usually give this script 755 permissions.

tail -150 $ORACLE_BASE/admin/$ORACLE_SID/bdump/alert*.log | more
```

Hot Backup Script

The following is a small piece of code that enables you to initiate hot backups. Use this as a *sample* for your script. I used scripts to generate this dynamically, but it can be hard coded as well. I also could have made use of Unix environment variables for the copy and compress steps, but I wanted to keep it simple.

This script first spools to create a log. Next it puts a tablespace in hot backup mode. It uses `cp` to copy the file to a backup location. The `-p` option is probably not necessary. Next it uses `gzip` to compress the file. The tablespace is then taken out of hot backup mode. At the end of the script, I make a text copy and a binary copy of the control file. The timestamp for the binary copy is dynamically generated. Finally, I force a log switch and end the spool.

Especially when writing backup scripts, you *must* test the scripts to make sure they work and nothing becomes corrupt.

run_hots.sql

```
spool hot_backup_run.lst
alter tablespace TOOLS begin backup;
!cp -p /u02/app/oracle/oradata/rh1dev1/tools01.dbf
/ubackup/hot_backup_dump/rh1dev1
!gzip -f /ubackup/hot_backup_dump/rh1dev1/tools*.dbf
alter tablespace TOOLS end backup;
alter tablespace USERS begin backup;
!cp -p /u02/app/oracle/oradata/rh1dev1/users01.dbf
/ubackup/hot_backup_dump/rh1dev1
!gzip -f /ubackup/hot_backup_dump/rh1dev1/users*.dbf
alter tablespace USERS end backup;
alter database backup controlfile to trace;
alter database backup controlfile to
'/ubackup/hot_backup_dump/rh1dev1/control.15062000135844';
alter system switch logfile;
spool off
exit
```