oracle 10g 研究ORACLE_HOME rdbms admin 下的脚本的功能（13）awrextr.sql

oracle 10g 研究ORACLE_HOME rdbms admin 下的脚本的功能（13）awrextr.sql

```
#AWR Extract
#SQL/Plus script to help users extract data from the AWR

Rem
Rem $Header: awrextr.sql 06-apr-2005.18:28:53 mlfeng Exp $
Rem
Rem awrextr.sql
Rem
Rem Copyright (c) 2004, 2005, Oracle. All rights reserved.
Rem
Rem    NAME
Rem      awrextr.sql - AWR Extract
Rem
Rem    DESCRIPTION
Rem      SQL/Plus script to help users extract data from the AWR
Rem
Rem    NOTES
Rem      User must be connected as SYS to run this SQL/Plus script.
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    mlfeng      03/01/05 - Add Disclaimer for support
Rem    mlfeng      06/01/04 - mlfeng_awr_import_export
Rem    mlfeng      05/17/04 - Created
Rem

--   Use local dbid
-- define dbid = '';
--
--   List all snapshots
-- define num_days = '';
--
--   List no (i.e. 0) snapshots
-- define num_days = 0;
--
-- List past 3 day's snapshots
-- define num_days = 3;
--
--  Optionally, set the snapshots to export.  If you do not set them,
--  you will be prompted for the values.
-- define begin_snap = 0;
-- define end_snap   = 10000000;
--
--  Use the default directory name and file name
-- define directory_name = 'DATA_PUMP_DIR'
-- define file_name      = ''
--

set echo off heading on underline on verify off
set feedback off linesize 80 termout on;

prompt ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
prompt ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
prompt  Disclaimer: This SQL/Plus script should only be called under
```

```
prompt              the guidance of Oracle Support.
prompt ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
prompt ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
prompt
prompt
prompt ~~~~~~~~~~~~~
prompt  AWR EXTRACT
prompt ~~~~~~~~~~~~~
prompt ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
prompt ~  This script will extract the AWR data for a range of snapshots  ~
prompt ~  into a dump file.  The script will prompt users for the         ~
prompt ~  following information:                                         ~
prompt ~     (1) database id                                             ~
prompt ~     (2) snapshot range to extract                               ~
prompt ~     (3) name of directory object                                ~
prompt ~     (4) name of dump file                                       ~
prompt ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

--
-- Get the current database information - this will be used as the
-- default for the database ID in the AWR schema to extract from.

set termout off;
column db_name   heading "DB Name" format a12;
column db_dbid   heading "DB Id"   format 9999999999 just c new_value db_dbid;

select d.dbid            db_dbid
     , d.name            db_name
  from v$database d;

set termout on;

column dbb_name   heading "DB Name"   format a12;
column dbbid      heading "DB Id"     format a12 just c;
column host       heading "Host"      format a12;

prompt
prompt
prompt Databases in this Workload Repository schema
prompt ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
select distinct
       (case when cd.dbid = wr.dbid and
                  cd.name = wr.db_name
             then '* '
             else '  '
        end) || wr.dbid   dbbid
     , wr.db_name         dbb_name
     , wr.host_name       host
  from dba_hist_database_instance wr, v$database cd
  order by dbbid desc;

prompt
prompt The default database id is the local one: '&db_dbid'.  To use this
prompt database id, press <return> to continue, otherwise enter an alternative.
prompt

set heading off;
column dbid new_value dbid noprint;
```

```
select 'Using ' || nvl('&&dbid','&db_dbid') || ' for Database ID'
     , nvl('&&dbid','&db_dbid') dbid
  from sys.dual;


-- Set up Bind for database ID
variable dbid       number;

begin
  :dbid      :=  &dbid;
end;
/

--
--  Error reporting

whenever sqlerror exit;
variable max_snap_time char(10);

declare

  cursor cidnum is
     select 'X'
       from dba_hist_database_instance
      where dbid            = :dbid;

  cursor csnapid is
     select to_char(max(end_interval_time),'dd/mm/yyyy')
       from dba_hist_snapshot
      where dbid            = :dbid;

  vx     char(1);

begin

  -- Check Database Id/Instance Number is a valid pair
  open cidnum;
  fetch cidnum into vx;
  if cidnum%notfound then
    raise_application_error(-20200,
      'Database ' || :dbid ||
      ' does not exist in DBA_HIST_DATABASE_INSTANCE');
  end if;
  close cidnum;

  -- Check Snapshots exist for Database Id/Instance Number
  open csnapid;
  fetch csnapid into :max_snap_time;
  if csnapid%notfound then
    raise_application_error(-20200,
      'No snapshots exist for Database ' || :dbid);
  end if;
  close csnapid;

end;
/
```

```
--
--  Ask how many days of snapshots to display

set termout on;
column dbid_fmt noprint;
column db_name     format a12  heading 'DB Name';
column snap_id     format 99999990 heading 'Snap Id';
column snapdat     format a18  heading 'Snap Started' just c;

prompt
prompt
prompt Specify the number of days of snapshots to choose from
prompt ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
prompt Entering the number of days (n) will result in the most recent
prompt (n) days of snapshots being listed.  Pressing <return> without
prompt specifying a number lists all completed snapshots.
prompt
prompt

set heading off;
column num_days new_value num_days noprint;
select    'Listing '
       || decode( nvl('&&num_days', 3.14)
                , 0    , 'no snapshots'
                , 3.14 , 'all Completed Snapshots'
                , 1    , 'the last day''s Completed Snapshots'
                , 'the last &num_days days of Completed Snapshots')
     , nvl('&&num_days', 3.14)  num_days
  from sys.dual;
set heading on;


--
-- List available snapshots

break on db_name;

ttitle off;

select s.dbid                                           dbid_fmt
     , max(di.db_name)                                  db_name
     , s.snap_id                                        snap_id
     , to_char(max(s.end_interval_time), 'dd Mon YYYY HH24:mi') snapdat
  from dba_hist_snapshot s
     , dba_hist_database_instance di
 where s.dbid              = :dbid
   and di.dbid             = s.dbid
   and di.instance_number  = s.instance_number
   and di.startup_time     = s.startup_time
   and s.end_interval_time >= decode( &num_days
                                    , 0   , to_date('31-JAN-9999','DD-MON-YYYY')
                                    , 3.14, s.end_interval_time
                                    , to_date(:max_snap_time,'dd/mm/yyyy') -
                                              (&num_days-1))
 group by s.dbid, snap_id
 order by s.dbid, snap_id;

clear break;
```

```
ttitle off;


--
--  Ask for the snapshots Id's which are to be compared

prompt
prompt
prompt Specify the Begin and End Snapshot Ids
prompt ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
prompt Begin Snapshot Id specified: &&begin_snap
prompt
prompt End   Snapshot Id specified: &&end_snap
prompt

--
--  Set up the snapshot-related binds

variable bid        number;
variable eid        number;

begin
  :bid       :=  &begin_snap;
  :eid       :=  &end_snap;

  /* do a basic check to ensure end_snap >= begin_snap */
  IF (:bid > :eid) THEN
    RAISE_APPLICATION_ERROR(-20019, 'begin_snap must be less than or ' ||
                                    'equal to end_snap.');
  END IF;
end;
/


--
-- Ask User for Directory Name
--

prompt
prompt Specify the Directory Name
prompt ~~~~~~~~~~~~~~~~~~~~~~~~~

set heading on;
column dirname format a30 heading 'Directory Name'
column dirpath format a49 heading 'Directory Path' wrap

select directory_name dirname, directory_path dirpath
  from DBA_DIRECTORIES
 order by directory_name;

set termout off;
column dflt_dir new_value dflt_dir noprint;
select ''  dflt_dir from dual;
set termout on;

prompt
prompt Choose a Directory Name from the above list (case-sensitive).
prompt
```

```
set heading off;
column directory_name new_value directory_name noprint;
select 'Using the dump directory: ' || nvl('&&directory_name','&dflt_dir')
     , nvl('&&directory_name','&dflt_dir') directory_name
  from sys.dual;


variable dmpdir  varchar2(30);
variable dmppath varchar2(4000)

declare

  cursor dirpath (dirname varchar2) is
    select directory_path
      from dba_directories
      where directory_name = dirname;

begin
  :dmpdir  := '&directory_name';

   /* select the directory path into a variable */
   open dirpath(:dmpdir);

   fetch dirpath into :dmppath;

   if (dirpath%NOTFOUND) then
     RAISE_APPLICATION_ERROR(-20103,
                             'directory name ''' || :dmpdir ||
                              ''' is invalid', TRUE);
   end if;

   close dirpath;
end;
/

set termout off;
column dflt_name new_value dflt_name noprint;
select 'awrdat'||'_'||:bid||'_'||:eid  dflt_name from dual;
set termout on;

prompt
prompt Specify the Name of the Extract Dump File
prompt ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
prompt The prefix for the default dump file name is &dflt_name..
prompt To use this name, press <return> to continue, otherwise enter
prompt an alternative.
prompt

set heading off;
column file_name new_value file_name noprint;
select 'Using the dump file prefix: ' || nvl('&&file_name','&dflt_name')
     , nvl('&&file_name','&dflt_name') file_name
  from sys.dual;

variable dmpfile varchar2(30);

begin
```

```
  :dmpfile := '&file_name';
end;
/

set serveroutput on;
exec dbms_output.enable(500000);
set termout on;

column loc    format a80 newline;
column locend format a80;

declare
  begpos   NUMBER;
  numchar  NUMBER  := 74;

begin
  dbms_output.put_line('|');
  dbms_output.put_line('| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~');
  dbms_output.put_line('|  The AWR extract dump file will be located   ');
  dbms_output.put_line('|  in the following directory/file:            ');

  begpos := 1;
  WHILE (begpos <= length(:dmppath)) LOOP
    dbms_output.put_line('|   ' || substr(:dmppath, begpos, numchar));
    begpos := begpos + numchar;
  END LOOP;

  dbms_output.put_line('|   ' || :dmpfile || '.dmp');
  dbms_output.put_line('| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~');
  dbms_output.put_line('|');
  dbms_output.put_line('|  *** AWR Extract Started ...');
  dbms_output.put_line('|');
  dbms_output.put_line('|  This operation will take a few moments. The ');
  dbms_output.put_line('|  progress of the AWR extract operation can be ');
  dbms_output.put_line('|  monitored in the following directory/file: ');

  begpos := 1;
  WHILE (begpos <= length(:dmppath)) LOOP
    dbms_output.put_line('|   ' || substr(:dmppath, begpos, numchar));
    begpos := begpos + numchar;
  END LOOP;

  dbms_output.put_line('|   ' || :dmpfile || '.log');
  dbms_output.put_line('|');
end;
/

whenever sqlerror continue;
set heading off;
set linesize 110 pagesize 50000;
set echo off;
set feedback off;
set termout on;

begin
  /* call PL/SQL routine to extract the data */
  dbms_swrf_internal.awr_extract(dmpfile  => :dmpfile,
                                 dmpdir   => :dmpdir,
```

```
                                bid      => :bid,
                                eid      => :eid,
                                dbid     => :dbid);
end;
/

undefine dbid
undefine num_days
undefine begin_snap
undefine end_snap
undefine directory_name
undefine file_name
```