oracle 10g 研究ORACLE_HOME rdbms admin 下的脚本的功能（1）a0801070.sql

oracle 10g 研究ORACLE_HOME rdbms admin 下的脚本的功能（1）a0801070.sql

```
#upgrade from 8.1.7 to 9.0.1
#upgrade from 9.0.1 to the new release
#upgrade from 9.0.1 to the new release

Rem
Rem $Header: a0801070.sql 04-jun-2001.12:42:18 rburns Exp $
Rem
Rem a0801070.sql
Rem
Rem  Copyright (c) Oracle Corporation 1999, 2000. All Rights Reserved.
Rem
Rem    NAME
Rem      a0801070.sql - additional ANONYMOUS BLOCK dictionary upgrade.
Rem        Upgrade Oracle RDBMS from 8.1.7 to the new release
Rem
Rem
Rem    DESCRIPTION
Rem      Additional upgrade script to be run during the migration of an
Rem      8.1.7 database to the new release.
Rem
Rem  This script is called from u08020x0.sql and a0801050.sql
Rem
Rem  Put any anonymous block related changes here.
Rem  Any dictionary create, alter, updates and deletes
Rem   that must be performed before catalog.sql and catproc.sql go
Rem  in c0801070.sql
Rem
Rem      The upgrade is performed in the following stages:
Rem        STAGE 1: additional steps to upgrade from 8.1.7 to 9.0.1
Rem        STAGE 2: upgrade from 9.0.1 to the new release
Rem
Rem    NOTES
Rem      * This script must be run using SQL*PLUS.
Rem      * You must be connected AS SYSDBA to run this script.
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    rburns      06/04/01 - add 9.0.1 upgrade
Rem    twtong      04/20/01 - retrict granting on commit refresh to mv owner
Rem    rburns      03/11/01 - add drop constraint
Rem    rburns      03/08/01 - fix group_column_pk
Rem    fputzolu    02/27/01 - bug 1660689: pass PLS/INTEGERs to kkpod820Upgrade
Rem    araghava    02/28/01 - change part820_upgrade to part817_upgrade.
Rem    apadmana    02/20/01 - fix local complex mv upgrade
Rem    arrajara    02/16/01 - local complex mv upgrade
Rem    arrajara    01/18/01 - Obsolete 'r' scripts. Move the contents here
Rem    najain      01/17/01 - handle exception queues
Rem    arrajara    01/10/01 - Move replication specific stmts to c0801070.sql
Rem    twtong      01/04/01 - grant on commit refresh privilege to mv owner
Rem    dalpern     11/30/00 - privileges for kga debugger
Rem    celsbern    12/06/00 - fixing replication upgrade
Rem    nbhatt      11/29/00 -  add transformation column to all subscriber tables
Rem    rburns      11/20/00 - update to 9.0
Rem    elu         11/11/00 - modify repcat$_parameter_column
Rem    fputzolu    09/22/00 - fix for SQL*PLUS invocation, eliminate
Rem                           DBMS_OUTPUT calls.
Rem    fputzolu    09/12/00 - upgrade bhiboundval for part tables & indexes
Rem    nshodhan    09/01/00 - add support for mlog.oldest_new column
Rem    svivian     09/01/00 - plan stability upgrades
Rem    liwong      05/17/00 - add_master_db w/o quiesce
Rem    jdavison    04/11/00 - Modify usage notes for 9.0 changes.
Rem    liwong      07/30/99 - replicated objects
Rem    liwong      07/27/99 - Created
Rem

Rem =========================================================================
Rem BEGIN STAGE 1: upgrade from 8.1.7 to 9.0.1
Rem =========================================================================

REM ===================================================================
REM Grant on commit refresh system privilege to the owner of on
REM commit refresh materialized view which references tables
REM outside of the owner schema
REM ===================================================================
declare
  owner varchar(30);
  cursor c_mv_owner is
     select distinct u.name
        from sum$ s, sumdep$ d, obj$ o1, obj$ o2, user$ u
        where s.obj# = d.sumobj# and
              bitand(s.mflags, 65536) != 0 and
              d.p_obj# = o1.obj# and
              s.obj# = o2.obj# and
              o1.owner# != o2.owner# and
              o2.owner# = u.user#;
begin
   open c_mv_owner;
   loop
     fetch c_mv_owner into owner;
     exit when c_mv_owner%NOTFOUND;
     execute immediate 'GRANT ON COMMIT REFRESH TO ' || owner;
   end loop;
   close c_mv_owner;
end;
/

REM ========== grant debug privileges to JAVADEBUGPRIV role ========
declare
  n number;
begin
  select count(*) into n from user$ where name='JAVADEBUGPRIV' and type#=0;
  if (n > 0) then
    execute immediate 'GRANT DEBUG CONNECT SESSION TO JAVADEBUGPRIV';
    execute immediate 'GRANT DEBUG ANY PROCEDURE TO JAVADEBUGPRIV';
  end if;
end;
/



REM ===================== begin of replication upgrade =============
REM
REM ORA-06550 expected if Advanced Replication is not installed
REM THESE ARE OK IF ADVANCED REPLICATION IS NOT INSTALLED
```

```
REM
REM If the log contains new values, set oldest_new to older of oldest and
REM oldest_pk otherwise set it to 01/01/4000

DECLARE
  oldest_new_val DATE;                          -- temp place holder for update
  date_4k        DATE := to_date('4000-01-01:00:00:00',
                                 'YYYY-MM-DD:HH24:MI:SS');
  CURSOR mlog_cur IS
    SELECT mowner, master, oldest, oldest_pk,
           DECODE(bitand(flag, 16), 16, 1, 0) inv_val
    FROM   sys.mlog$
    FOR UPDATE;
BEGIN
  -- open cursor
  FOR mlogcur IN mlog_cur LOOP

    -- There is a bug which prevents one to alter MV log to include/exclude
    -- new values once the MV log has been created. i.e. the include/exclude
    -- new values option specified at the MV log creation time can not be
    -- modified. Its highly unlikely that this bug will be fixed in 817 patch
    -- releases. So we are assuming that during upgrade, we can set the
    -- older of the oldest and oldest_pk time-stamps to mlog$.oldest_new if
    -- new values are included in the log.
    IF (mlogcur.inv_val = 1) THEN
      -- As we are selecting the older of oldest and oldest_pk, we do not have
      -- to worry about about 4K timestamp here.
      IF (mlogcur.oldest > mlogcur.oldest_pk) THEN
        oldest_new_val := mlogcur.oldest_pk;
      ELSE
        oldest_new_val := mlogcur.oldest;
      END IF;
    ELSE                                         -- new values are not included
      oldest_new_val := date_4k;
    END IF;

    -- update mlog$
    UPDATE sys.mlog$ m
    SET    m.oldest_new = oldest_new_val
    WHERE  m.mowner     = mlogcur.mowner
    AND    m.master     = mlogcur.master;
  END LOOP;

  COMMIT;

END;
/

---------------------------------------------------------------------------
---- populate toid, lcname correctly for repcat$_repcolumn
---------------------------------------------------------------------------
BEGIN
  dbms_repcat_mig_internal.fix_repcolumn;
END;
/

---------------------------------------------------------------------------
-- populate column_pos, attribute_sequence_no for repcat$_parameter_column
```

```
---------------------------------------------------------------------------

ALTER TABLE system.repcat$_parameter_column
  DROP CONSTRAINT repcat$_parameter_column_pk
/
BEGIN
  dbms_repcat_mig_internal.fix_parameter_column;
END;
/
ALTER TABLE system.repcat$_parameter_column
  ADD CONSTRAINT repcat$_parameter_column_pk
          PRIMARY KEY (sname,
                       oname,
                       conflict_type_id,
                       reference_name,
                       sequence_no,
                       parameter_table_name,
                       parameter_sequence_no,
                       column_pos)
/

---------------------------------------------------------------------------
-- populate pos, for repcat$_grouped_column
---------------------------------------------------------------------------

ALTER TABLE system.repcat$_grouped_column
  DROP CONSTRAINT repcat$_grouped_column_pk
/

BEGIN
  dbms_repcat_mig_internal.fix_grouped_column;
END;
/

ALTER TABLE system.repcat$_grouped_column
  ADD CONSTRAINT repcat$_grouped_column_pk
   PRIMARY KEY (sname, oname, group_name, column_name, pos)
/

REM
REM Local complex materialized view upgrade
REM

DECLARE
  new_flag  NUMBER;
  CURSOR complex_summary IS
   SELECT s.obj#, s.mflags
   FROM sys.sum$ s, sys.obj$ o, sys.user$ u, sys.snap$  mv
   WHERE mv.instsite           = 0                          /* non-repapi mv */
     AND mv.mlink             IS NULL                               /* local */
     AND bitand(mv.flag, 256) != 0                                /* complex */
     AND u.name = mv.sowner
     AND u.user# = o.owner#
     AND o.name  = mv.vname
     AND o.type# = 42
     AND o.obj#  = s.obj#;
BEGIN
  FOR rec IN complex_summary LOOP
```

```
      new_flag := rec.mflags;

      IF dbms_ijob.bit(new_flag, 64) THEN         /* unusable, leave it alone */
        goto next_summ;
      END IF;

      IF dbms_ijob.bit(new_flag, 16) THEN                             /* fresh */
        new_flag := new_flag - 16;
      END IF;

      IF dbms_ijob.bit(new_flag, 32) THEN                           /* unknown */
        new_flag := new_flag - 32;
      END IF;

      IF dbms_ijob.bit(new_flag, 512) = FALSE THEN                 /* staleful */
        new_flag := new_flag + 512;
      END IF;

      IF dbms_ijob.bit(new_flag, 1) = FALSE THEN                /* known_stale */
        new_flag := new_flag + 1;
      END IF;

      IF new_flag != rec.mflags THEN
        UPDATE sys.sum$ s SET s.mflags = new_flag WHERE s.obj# = rec.obj#;
        COMMIT;
      END IF;

      <<next_summ>>
      NULL;
  END LOOP;
EXCEPTION WHEN OTHERS THEN
  DBMS_SYSTEM.KSDWRT(2, 'Exception:a0801070.sql:' ||
           TO_CHAR(SQLCODE) || ': ' || SQLERRM);
END;
/

REM ===================== end of replication upgrade =============
REM At this point, the replication upgrade is over.  Any ORA-00942
REM errors beyond this point require closer scrutiny.
Rem ==============================================================

REM ================== begin of upgrade for partition feature =================
REM This is the second part of the upgrade, the first part is in c0801070.sql.
REM The new 9.0.0 column bhiboundval of tabpart$, tabcompart, indpart$,
REM is derived from the hiboundval column using a C trusted
REM callout part817_upgrade.
REM If generation of some bhiboundval values fails, then error messages are
REM generated. In any case generation of bhiboundval values may be
REM idempotently re-requested.
SET SERVEROUTPUT ON SIZE 10000;
BEGIN
  DBMS_OUTPUT.PUT_LINE
       ('  Begin of conversion of partitioned tables and indexes');
END;
/
-- create library PART817_UPGRADE_LIB containing part817_upgrade
CREATE OR REPLACE LIBRARY PART817_UPGRADE_LIB TRUSTED AS STATIC
```

```
/

-- create type of array of data types of part columns
CREATE OR REPLACE TYPE dtypes_type AS VARRAY(16) of NUMBER;
/

-- create function part817_upgrade (supporting the trusted callback)
CREATE OR REPLACE FUNCTION part817_upgrade
 (textkey IN LONG, textkeyl IN PLS_INTEGER,
  numkeycols IN PLS_INTEGER, dtypes IN dtypes_type) RETURN RAW
IS EXTERNAL
NAME "PART_817_UPGRADE"
WITH CONTEXT
PARAMETERS(CONTEXT, textkey, textkey INDICATOR sb2, textkeyl ub2,
           numkeycols ub2, dtypes OCIColl,
           RETURN LENGTH ub2, RETURN INDICATOR sb2, RETURN)
LIBRARY PART817_UPGRADE_LIB;
/

-- initialize the new binary column bhiboundval from the text column
-- hiboundval in tabpart$, indpart$, tabcompart$.
-- bhiboundval columns of indcompart$ are left null because composite
-- indexes are always local, hence their partition keys are null.
DECLARE
  -- constants: table object, range type, IOT overflow bit in tab$
  table_object   CONSTANT NUMBER := 2;
  range_type     CONSTANT NUMBER := 1;
  overflow_table CONSTANT NUMBER := 512;

  -- cursor used to get info about a range/composite partitioned object:
  --   from partobj$: obj#, number of partition key columns,
  --                  spare2(sub partitioning info)
  --   from obj$: owner#, name, object type (index, table).
  -- local indexes are excluded, because they have null part keys.
  CURSOR c_obj IS
    SELECT obj$.obj#, partkeycols, partobj$.spare2, owner#, name, type#
    FROM partobj$, obj$
    WHERE parttype = range_type AND partobj$.obj# = obj$.obj#
    AND (BITAND(partobj$.flags, 1) = 0);       -- exclude local indexes

  objnum      NUMBER;               -- object #
  numkeycols  PLS_INTEGER;          -- # of partition key columns

  -- cursor used to get data types of part columns of an object
  -- in the varray called dtypes using loop index ix
  CURSOR c_col IS
    SELECT type# FROM partcol$ WHERE obj# = objnum ORDER by POS#;
  -- array of data types of part cols
  dtypes      dtypes_type := dtypes_type(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
  ix          PLS_INTEGER;      -- index over array

  user_name   VARCHAR2(30);     -- schema user name used for reporting

  -- cursors used to select partition text key and text length
  -- of a partition, to be resubmitted as parameters for updating.
  -- we have to read the partition text key and resubmit it
  -- as a local variable hival because LONG columns can't be passed
  -- as arguments of SQL functions.
  CURSOR ct_par IS
    SELECT hiboundval, hiboundlen FROM tabpart$ WHERE bo# = objnum
    FOR UPDATE;
  CURSOR ctc_par IS
    SELECT hiboundval, hiboundlen FROM tabcompart$ WHERE bo# = objnum
    FOR UPDATE;
  CURSOR ci_par IS
    SELECT hiboundval, hiboundlen FROM indpart$
    WHERE bo# = objnum FOR UPDATE;
  hival          LONG(30000);  -- local copy of hiboundval
  hilen          PLS_INTEGER;  -- local copy of hiboundlen
  overflow_flag  NUMBER;       -- 1 if overflow table of IOT, else 0

BEGIN
  -- loop reading information about partitioned objects
  FOR r_obj IN c_obj LOOP
    BEGIN                        -- loop over partitioned objects
      objnum := r_obj.obj#;
      numkeycols := r_obj.partkeycols;

      -- loop initializing dtypes varray for current object
      ix := 1;
      FOR r_col IN c_col LOOP
        dtypes(ix) := r_col.type#;
        ix := ix + 1;
      END LOOP;

      -- initialize new bhiboundval columns in appropriate xxxpart$ table
      -- skip IOT overflow tables, for which overflow_flag = 1
      -- an object is composite if its spare2 column of partobj$ is non-zero
      IF r_obj.type# = table_object THEN
        BEGIN                 -- table
          SELECT TRUNC(MOD(property/overflow_table, 2)) INTO overflow_flag
           FROM tab$ WHERE obj# = objnum;
          IF overflow_flag = 0 THEN  -- not an IOT overflow table
            IF r_obj.spare2 = 0 THEN   -- non composite table
              OPEN ct_par;
              LOOP
                FETCH ct_par INTO hival, hilen;
                EXIT WHEN ct_par%NOTFOUND;
                UPDATE tabpart$ SET bhiboundval = part817_upgrade
                 (hival, hilen, numkeycols, dtypes)
                 WHERE CURRENT OF ct_par;
              END LOOP;
              CLOSE ct_par;
            ELSE                       -- composite table
              OPEN ctc_par;
              LOOP
                FETCH ctc_par INTO hival, hilen;
                EXIT WHEN ctc_par%NOTFOUND;
                UPDATE tabcompart$ SET bhiboundval = part817_upgrade
                 (hival, hilen, numkeycols, dtypes)
                 WHERE CURRENT OF ctc_par;
              END LOOP;
              CLOSE ctc_par;
            END IF;
          END IF;             -- not an IOT overflow table
        END;                  -- table
      ELSE
        BEGIN                 -- index, must be global hence non composite
          OPEN ci_par;
          LOOP
            FETCH ci_par INTO hival, hilen;
            EXIT WHEN ci_par%NOTFOUND;
            UPDATE indpart$ SET bhiboundval = part817_upgrade
             (hival, hilen, numkeycols, dtypes)
             WHERE CURRENT OF ci_par;
          END LOOP;
          CLOSE ci_par;
        END;                  -- index, must be global hence non composite
      END IF;
    EXCEPTION WHEN OTHERS THEN
      SELECT name INTO user_name FROM user$ WHERE user# = r_obj.owner#;
      DBMS_OUTPUT.PUT_LINE
       ('Error ' || TO_CHAR(SQLCODE) || ': ' || SQLERRM);
      DBMS_OUTPUT.PUT_LINE
       ('Error in creating partition bhiboundval for table/index:');
      DBMS_OUTPUT.PUT_LINE
       ('  ' || user_name || '.' || r_obj.name );
    END;                         -- loop over partitioned objects
  END LOOP;

  COMMIT;
END;
/

-- drop the function, library and type used by bhiboundval column update

DROP FUNCTION part817_upgrade;

DROP LIBRARY PART817_UPGRADE_LIB;

DROP TYPE dtypes_type;

BEGIN
  DBMS_OUTPUT.PUT_LINE
       ('  End of conversion of partitioned tables and indexes');
END;
/
SET SERVEROUTPUT OFF;

REM ================== end of upgrade for partition feature ================

REM ==============================================================
REM Plan Stability Changes
REM ==============================================================

BEGIN
   DBMS_OUTLN.UPDATE_SIGNATURES;
END;
/

REM ==============================================================
REM AQ upgrade - transformations
REM ==============================================================
DECLARE

 CURSOR get_old81_queue_tables_c IS
   SELECT t.schema, t.name FROM system.aq$_queue_tables t
   WHERE bitand(t.flags,8) = 8 AND   bitand(t.flags,1) = 1 ;
 old81_queue_tables  get_old81_queue_tables_c%ROWTYPE;
 add_col_sql   VARCHAR2(300);
BEGIN
   FOR old81_queue_tables IN get_old81_queue_tables_c LOOP
     add_col_sql := 'ALTER TABLE '
                    || old81_queue_tables.schema || '.'
                    || 'AQ$_'|| old81_queue_tables.name ||'_S'
                    || ' ADD (trans_name VARCHAR2(61))';
     BEGIN
        EXECUTE IMMEDIATE add_col_sql;
     EXCEPTION
       WHEN OTHERS THEN
        RAISE;
     END;
   END LOOP;
END;
/

REM ===============================================================
REM AQ upgrade - handle exception queues
REM ===============================================================

DECLARE

  obj_no          system.aq$_queue_tables.objno%TYPE;
  get_msg         varchar2(512);
  upd_tid         varchar2(512);
  q_cursor        INTEGER;
  q_cursor2       INTEGER;
  ignore     INTEGER;
  msgid    RAW(16);            -- message id of message in queue table
  sqlquery        varchar2(256);

  type rt is REF CURSOR;
  sqlrc  rt;       -- ref cursor for sql statement
  sqlrc2 rt;       -- ref cursor for sql statement

  qt_schema       system.aq$_queue_tables.schema%TYPE;
  qt_name         system.aq$_queue_tables.name%TYPE;
  qt_objno        system.aq$_queue_tables.objno%TYPE;
  qt_flags        system.aq$_queue_tables.flags%TYPE;


BEGIN

  sqlquery := 'SELECT schema, name, objno, flags FROM system.aq$_queue_tables';

  -- loop through all the index entries in the old dequeue IOT
  OPEN sqlrc FOR sqlquery;
  LOOP

    FETCH sqlrc INTO qt_schema, qt_name, qt_objno, qt_flags;
    EXIT WHEN sqlrc%NOTFOUND;
    IF ((bitand(qt_flags,1) = 1) AND (bitand(qt_flags,8) = 8)) THEN
      get_msg := 'SELECT msgid FROM ' || qt_schema || '.' || qt_name || ' WHERE q_name IN ';
      get_msg := get_msg || '(SELECT name FROM system.aq$_queues WHERE table_objno = :obj AND usage = 1)';

      upd_tid := 'UPDATE ' || qt_schema || '.AQ$_' || qt_name || '_H SET transaction_id = ''INVALID_TRANSACTION'' WHERE msgid =
:msg AND ';
      upd_tid := upd_tid || 'transaction_id IS NOT NULL and dequeue_time IS NULL';

      -- loop through all the index entries in the queue table
      OPEN sqlrc2 FOR get_msg USING qt_objno;

      q_cursor2 := dbms_sql.open_cursor;
      dbms_sql.parse(q_cursor2, upd_tid, dbms_sql.v7);

      LOOP

        FETCH sqlrc2 INTO msgid;
        EXIT WHEN sqlrc2%NOTFOUND;

        dbms_sql.bind_variable(q_cursor2, 'msg', msgid);
        ignore := dbms_sql.execute(q_cursor2);

      END LOOP;

      dbms_sql.close_cursor(q_cursor2);

    ELSE

      upd_tid := 'UPDATE ' || qt_schema || '.' || qt_name;
      upd_tid := upd_tid || ' SET deq_tid = ''INVALID_TRANSACTION'' WHERE deq_tid IS NULL AND deq_time IS NULL AND q_name IN ';
      upd_tid := upd_tid || '(SELECT name FROM system.aq$_queues WHERE table_objno = :obj AND usage = 1)';

      EXECUTE IMMEDIATE upd_tid USING qt_objno;

    END IF;

  END LOOP;

END;
/

COMMIT
/

REM ============== End of Plan Stability Changes =================

Rem =====================================================================
Rem END STAGE 1: upgrade from 8.1.7 to 9.0.1
Rem =====================================================================

Rem =====================================================================
Rem BEGIN STAGE 2: upgrade from 9.0.1 to the new release
Rem =====================================================================
Rem

@@a0900010
```

```
Rem =========================================================================
Rem END STAGE 2: upgrade from 9.0.1 to the new release
Rem =========================================================================

Rem *************************************************************************
Rem END a0801070.sql
Rem *************************************************************************
```