

Knowing how much time left to complete rollback operation.

Author: David Gornshtein

WisdomForce Technologies, Inc

<http://www.wisdomforce.com>

Note*: The latest copy of this document is available at
http://www.wisdomforce.com/dweb/resources/docs/complete_rollback_script.pdf

The script below is written for Oracle 9+.

You can rewrite it for version 8+, but please note it comes without support for killed session lookup. Also lookup for rollbacks continues after issuing of "shutdown abort".

*Note: There is known issue with Oracle prior 9i version, that there is a lack of support for "left outer join" statement defined by SQL standard.

Preliminaries:

fixed view sys.x\$ktuxe

view: x\$ktuxe
[k]ernel layer
[t]ransaction layer
[u]ndo
transaction [e]ntry

This view holds entry for each active undo slot.

```
DECLARE
  CURSOR tx
  IS
    SELECT /*+ USE_NL(S,T,X) */
      NVL
        (s.username,
         'session no more exists or running on the other node of RAC'
        ),
      x.ktuxeusn, x.ktuxeslt, x.ktuxesqn, x.ktuxesiz
  FROM ((sys.x_$ktuxe x LEFT JOIN sys.gv_$transaction t ON
        t.xidusn = x.ktuxeusn AND t.xidslot = x.ktuxeslt AND
        t.xidsqn = x.ktuxesqn AND x.inst_id = t.inst_id) LEFT JOIN
        sys.gv_$session s ON
        s.saddr = t.ses_addr AND
        s.inst_id = t.inst_id)
```

```

        WHERE x.ktuxesta = 'ACTIVE' AND x.ktuxesiz > 1;

user_name      VARCHAR2 (80);
xid_usn        NUMBER;
xid_slot       NUMBER;
xid_sqn        NUMBER;
used_ublk1     NUMBER;
used_ublk2     NUMBER;
BEGIN
  OPEN tx;

  LOOP
    FETCH tx
      INTO user_name, xid_usn, xid_slot, xid_sqn, used_ublk1;

    EXIT WHEN tx%NOTFOUND;

    IF tx%ROWCOUNT = 1
    THEN
      sys.DBMS_LOCK.sleep (120);

      SELECT SUM (ktuxesiz)
        INTO used_ublk2
        FROM sys.x_$ktuxe
        WHERE ktuxeusn = xid_usn
          AND ktuxeslt = xid_slot
          AND ktuxesqn = xid_sqn
          AND ktuxesta = 'ACTIVE';

      IF used_ublk2 < used_ublk1
      THEN
        sys.DBMS_OUTPUT.put_line ('session (' || user_name || ')');
        sys.DBMS_OUTPUT.put_line
          ( 'transaction '
            | xid_usn
            | '.'
            | xid_slot
            | '.'
            | xid_sqn
            | ' will finish rolling back at approximately '
            | TO_CHAR ( SYSDATE
                      + used_ublk2
                      / (used_ublk1 - used_ublk2)
                      / 30
                      / 24,
                      'HH24:MI:SS DD-MON-YYYY'
                    )
          );

        END IF;
      END IF;
    END LOOP;

    IF user_name IS NULL
    THEN
      sys.DBMS_OUTPUT.put_line ('No transactions appear to be rolling back.');
```

Assume the following situation - you have several unique indexes but want automatically create the appropriate unique constraints. See the following script :

```

DECLARE
TYPE TableName IS TABLE OF USER_IND_COLUMNS.TABLE_NAME%TYPE;
TYPE IndexName IS TABLE OF USER_IND_COLUMNS.INDEX_NAME%TYPE;
TYPE ColumnName IS TABLE OF USER_IND_COLUMNS.COLUMN_NAME%TYPE;

-- define arrays
aTableName TableName;
aIndexName IndexName;
aColumnName ColumnName;

-- define helpers
ColumnList VARCHAR2(256);
CurrentIndex VARCHAR2(256);
CurrentTable VARCHAR2(256);

BEGIN
SELECT TABLE_NAME, INDEX_NAME, COLUMN_NAME
BULK COLLECT INTO aTableName, aIndexName, aColumnName
FROM USER_IND_COLUMNS
WHERE (TABLE_NAME, COLUMN_NAME, COLUMN_POSITION) IN
(
(
SELECT TABLE_NAME, COLUMN_NAME, COLUMN_POSITION
FROM USER_IND_COLUMNS
WHERE INDEX_NAME IN
(
SELECT INDEX_NAME
FROM USER_INDEXES
WHERE UNIQUENESS = 'UNIQUE'
)
)
MINUS
SELECT TABLE_NAME, COLUMN_NAME, POSITION
FROM USER_CONS_COLUMNS
WHERE CONSTRAINT_NAME IN
(
SELECT CONSTRAINT_NAME
FROM USER_CONSTRAINTS
WHERE CONSTRAINT_TYPE IN ('U', 'P')
)
)
AND TABLE_NAME NOT IN
(
SELECT DISTINCT TABLE_NAME
FROM USER_TAB_PARTITIONS
)
ORDER BY INDEX_NAME, TABLE_NAME, COLUMN_POSITION;

CurrentIndex := 'NOTHING';
ColumnList := NULL;
FOR i IN aTableName.FIRST .. aTableName.LAST LOOP

-- IS IT FIRST INDEX
IF CurrentIndex = 'NOTHING' THEN
CurrentIndex := aIndexName(i);
CurrentTable := aTableName(i);
ColumnList := aColumnName(i);

-- IS IT ANOTHER INDEX
ELSIF CurrentIndex != aIndexName(i) OR i = aTableName.LAST THEN
DBMS_OUTPUT.PUT_LINE(' ALTER TABLE ' || CurrentTable
' ADD CONSTRAINT ' || CurrentIndex
' UNIQUE ( ' || ColumnList || ' );');
CurrentIndex := aIndexName(i);
CurrentTable := aTableName(i);
ColumnList := aColumnName(i);

```

```
ELSE
  ColumnList := ColumnList || ', ' || aColumnName(i);
END IF;
END LOOP;
END;
```