Data Masking Procedure

Author: **G S Chapman**

Date: **October 2008**

Version: **1.1**

Location of Document:

DOCUMENT HISTORY

Version	Date	Changed By:	Remarks
1.1	20/10/2008	G S Chapman	This version

DOCUMENT DISTRIBUTION

Copy No	Name	Role	Organisation

DOCUMENT REFERENCES

Document Name	Originator	Part Number	Version	Date

TABLE OF CONTENTS

PURPOSE OF DOCUMENT

This document describes a simple implementation of a data masking procedure within an Oracle database.

Data Masking

1 INTRODUCTION

This procedure was developed to meet a requirement of a client that would enable external development of code against sanitised data held upon the client site. The nature of the client data was such that it was not permissible to perform a simple export of the database since they were unwilling (and unable) to permit the data held in the database and the relationships between the data to be held off site.

To enable external application development work to proceed some 'sample data' was required, hence the development of this simple procedure to perform the task.

2 METHOD

To permit the export to be performed easily a new database user is created. Permission is given to the new user to create table and also to be able to read the source tables.

A single table is created to hold details of the tables and the columns that are to be masked. The command to create the table is given below.

```
CREATE TABLE mask_table
  (table_name VARCHAR2(32),
   column_name VARCHAR2(32),
   CONSTRAINT mask_pk PRIMARY KEY (table_name, column_name))
organization index;
```

The mask_table_columns procedure (as given in A1 Procedure mask_table_columns) will when run create an output table of the same name as the source table with a suffix of '_X'. If the source tabvle name is greater than 28 characters then the trailing characters are removed and the '_X' appended.

```
INSERT INTO mask_table
VALUES('INCIDENTREPORT','HOWRECEIVED');
INSERT INTO mask_table
VALUES('INCIDENTREPORT','RESEARCHCRITERIA');
INSERT INTO mask_table
VALUES('INCIDENTREPORT','DATETIMERECEIVED');
INSERT INTO mask_table
VALUES('INCIDENTREPORT','ACTIONINSTANCEID');
```

The procedure is invoked by a call of the following form:

```
BEGIN
  mask_table_columns('INCIDENTREPORT',TRUE);
END;
```

The first parameter, the table name is required. The second optional parameter instructs the procedure to ignore the table creation statement if the masking table already exists. This is required where a source table requires more than one column to be masked. The default is FALSE, i.e. To raise an exception if the masking table exists.

The procedure works upon date and character (VARCHAR2) columns determined from the table definition. Making use of the dbms random package the column values are

randomised. This enables the data resulting in the target tables to be used fro the generation of an user export for transfer to a remote location.

It is always possible to incorporate additional changes to the export data. One common change is to limit data changes on certain columns to within a certain data range. This is usually performed at the remote site since time and circumstances enable some 'important relationships to be established where required by the application developers.

NOTE: No indexes or primary keys are permitted on the target tables. This is a requirement of the client. If it were desired to implement such relationships, then the indexes would need to be disabled and further checks enabled to cross check the various tables to ensure that the constraints would be enabled correctly. This is outside the current client requirements.

3 ENHANCEMENTS

As mentioned above there are a number of enhancements that may be made to the code. All are outside the client requirements an hence have not been incorporated. If time and opportunity exist then these may be implemented at some future time.

Annex A

A1 PROCEDURE MASK_TABLE_COLUMNS

```
CREATE OR REPLACE PROCEDURE mask_table_columns(tab_name IN VARCHAR2 DEFAULT NULL,
                                               ignoreflag IN BOOLEAN DEFAULT FALSE)
IS
  /*

  Simple procedure to mask columns values in a table that can be extracted.

  Written 10th August 2004

  G S Chapman.


  Require explicit create table priv to a desired exportable user.

  Requires a table structure as follows for the criteria selection.

    CREATE TABLE mask_table
      (table_name VARCHAR2(32),
       column_name VARCHAR2(32),
       CONSTRAINT mask_pk PRIMARY KEY (table_name, column_name))
    organization index;

  Then typically insert into the above table the table name and columns
  to be masked.

    INSERT INTO mask_table
    VALUES('INCIDENTREPORT','HOWRECEIVED');
    INSERT INTO mask_table
    VALUES('INCIDENTREPORT','RESEARCHCRITERIA');
    INSERT INTO mask_table
    VALUES('INCIDENTREPORT','DATETIMERECEIVED');
    INSERT INTO mask_table
    VALUES('INCIDENTREPORT','ACTIONINSTANCEID');

  When run will create a table named after the first 30 characters of
  the input table, or the whole name if less than 30 character followed
  the characters '_X'.

  No indexes or primary keys are created.

  Typical usage would be as follows:

     BEGIN
       mask_table_columns('INCIDENTREPORT',TRUE);
    END;

  The second optional parameter instructs the procedure to ignore the
  table creation if the masking table already exists.  Default is FALSE,
  i.e.  To object if the masking table exists.

  The first parameter, the table name is required.

  Last change:     Description

  */

  DebugFlag BOOLEAN := FALSE;

  invalid_parameter EXCEPTION;
```

```
no_requirement EXCEPTION;
already_exists EXCEPTION;

CURSOR c1 (tab_name IN VARCHAR2)
IS
   SELECT column_name, data_type,
         data_length, column_id
   FROM user_tab_columns
   WHERE table_name = tab_name
   ORDER BY column_id;

v_tab_name VARCHAR2(32);
v_num INTEGER;
v_sqltext VARCHAR2(4096);
v_len INTEGER;
v_chk INTEGER;

/*
Functions form basis of criteria used in select string.

Could be made part of the database if required.

*/

function mask_char (isize IN INTEGER)
  RETURN VARCHAR2
IS
BEGIN
  RETURN dbms_random.string('a',isize);
END mask_char;

function mask_date
  RETURN date
IS
BEGIN
  RETURN sysdate - dbms_random.random;
END mask_date;

/*

Simplistic string display routine.
Will display the first 960 characters of the supplied string.

*/
PROCEDURE display_string(v_sqltext IN VARCHAR2)
IS
  v_len INTEGER;
BEGIN
  v_len := length (v_sqltext);
  dbms_output.put_line(to_char(v_len)||' string length');
  IF v_len < 120 AND v_len > 0 THEN
    dbms_output.put_line(substr(v_sqltext,1,v_len));
  ELSE
    dbms_output.put_line(substr(v_sqltext,1,120));
    IF v_len < 240 THEN
      dbms_output.put_line(substr(v_sqltext,121,120));
    ELSE
      dbms_output.put_line(substr(v_sqltext,121,120));
      IF v_len < 360 THEN
        dbms_output.put_line(substr(v_sqltext,241,v_len-240));
      ELSE
        dbms_output.put_line(substr(v_sqltext,241,120));
        IF v_len < 480 THEN
          dbms_output.put_line(substr(v_sqltext,361,v_len-360));
```

```
          ELSE
            dbms_output.put_line(substr(v_sqltext,361,120));
            IF v_len < 600 THEN
              dbms_output.put_line(substr(v_sqltext,481,v_len-480));
            ELSE
              dbms_output.put_line(substr(v_sqltext,481,120));
              IF v_len < 720 THEN
                dbms_output.put_line(substr(v_sqltext,601,v_len-600));
              ELSE
                dbms_output.put_line(substr(v_sqltext,601,120));
                IF v_len < 840 THEN
                  dbms_output.put_line(substr(v_sqltext,721,v_len-720));
                ELSE
                  dbms_output.put_line(substr(v_sqltext,721,120));
                  IF v_len < 960 THEN
                    dbms_output.put_line(substr(v_sqltext,841,v_len-840));
                  ELSE
                    dbms_output.put_line(substr(v_sqltext,841,120));
                  END IF;
                END IF;
              END IF;
            END IF;
          END IF;
        END IF;
      END IF;
    END IF;
  END display_string;

--  Main routine follows.

BEGIN
  IF tab_name IS NULL then
     RAISE invalid_parameter;
  END IF;

  SELECT count(*)
  INTO v_num
  FROM mask_table
  WHERE table_name = tab_name;

  IF v_num = 0 THEN
    dbms_output.put_line('No column entries to process.');
    RAISE no_requirement;
  END IF;

  v_tab_name := substr(tab_name, 1,30)||'_X';

  IF DebugFlag THEN
    dbms_output.put_line('Table to be created/used: '||v_tab_name);
  END IF;

  v_num := 0;
  BEGIN
    SELECT 1
    INTO v_num
    FROM user_tables
    WHERE table_name = tab_name;
  EXCEPTION
     WHEN no_data_found THEN
       IF ignoreflag THEN
         NULL;
       ELSE
         RAISE already_exists;
       END IF;
```

```
  END;


  IF v_num = 0 THEN
    v_sqltext := 'create table '||v_tab_name||
                 ' as (select * from '||tab_name||
                 ' where 1=0 )';
    IF DebugFlag THEN
      dbms_output.put_line(v_sqltext);
    END IF;

    EXECUTE IMMEDIATE
      v_sqltext;

    dbms_output.put_line('Table '||v_tab_name||' created.');
  ELSE
    dbms_output.put_line('Using Table '||v_tab_name||' which already exists.');
  END IF;

  -- Build up insert - select string
  v_sqltext := 'insert into '||v_tab_name||
               ' select ';
  FOR r1 IN c1(tab_name) LOOP

     v_chk := 0;
     BEGIN
       SELECT 1
       INTO v_chk
       FROM mask_table
       WHERE table_name = tab_name
       AND column_name = r1.column_name;
     EXCEPTION
       WHEN no_data_found THEN
         v_chk := 0;
     END;

     IF r1.column_id = 1 THEN
       IF v_chk = 1 THEN
         IF r1.data_type = 'VARCHAR2' THEN
             v_sqltext :=
v_sqltext||'dbms_random.string(''a'',nvl(length('||r1.column_name||'),0))';
         ELSIF r1.data_type = 'DATE' THEN
           v_sqltext := v_sqltext||'sysdate-mod(dbms_random.random,1000)';
         END IF;
       ELSE
           v_sqltext := v_sqltext||r1.column_name;
        END IF;
     ELSE
       IF v_chk = 1 THEN
         IF r1.data_type = 'VARCHAR2' THEN
           v_sqltext := v_sqltext||',
dbms_random.string(''a'',nvl(length('||r1.column_name||'),0))';
         ELSIF r1.data_type = 'DATE' THEN
            v_sqltext := v_sqltext||', sysdate-mod(dbms_random.random,1000)';
         END IF;
       ELSE
          v_sqltext := v_sqltext||','||r1.column_name;
       END IF;
     END IF;
  END LOOP;

  -- cludge up the end of the string.
  v_sqltext := v_sqltext ||' from '||tab_name;
  v_sqltext := v_sqltext||' where rownum < 11';
```

```
  IF DebugFlag THEN
    display_string(v_sqltext);
  END IF;

  IF length(v_sqltext) > 0 THEN
    EXECUTE IMMEDIATE
      v_sqltext;
  END IF;

  COMMIT;

EXCEPTION
  WHEN already_exists THEN
    dbms_output.put_line('Table to be created already exists.');
    dbms_output.put_line(' Remove table and rerun.');
    dbms_output.put_line(' Job terminated.');
  WHEN no_requirement THEN
    dbms_output.put_line(' Table specified does not have any masking criteria.');
    dbms_output.put_line(' Job terminated.');
  WHEN invalid_parameter THEN
    dbms_output.put_line('No table name specified.');
    dbms_output.put_line(' Job terminated.');
END mask_table_columns;
```