SWISS ORACLE USER GROUP

SOUG www.soug.ch

Newsletter 4/2013

Oktober 2013

- Oracle 12c Consolidation Planer
- Data Redaction & Transparent Sensitive Data Protection
- Oracle Forms Migration
- Oracle 12c “IDENTITY table clause” Feature und APEX

Miguel Anjo, Trivadis AG

Data Redaction & Transparent Sensitive Data Protection

One of the most expected security features of the latest version of Oracle Database is the real-time data masking solution, called Data Redaction. This amazing feature changes the data just before sending it to the user, being targeted to be used in production environments. As the data redaction is transparent to the user and application, it can be used without any change on the application level. There are powerful options like to use regular expressions as masking parameters and to use context variables for including or excluding certain use cases from the masking. As an add-on for Data Redaction and Virtual Private Database, Oracle 12c present us with the Transparent Sensitive Data Protection feature, which allows to categorize sensitive columns across a database and then to set a redaction or VPD policy to each defined category of columns.

The article will describe more in detail these new features, showing small examples how to use them.

Data Redaction

Until now if you wanted to mask the data on real time you needed to do it on the application layer or to use either custom made views or Virtual Private Database, all these solutions lacking functionalities that Data Redaction finally brings. With Data Redaction is now possible to easily totally or partially mask the data, randomize the data and set the masking conditions based on SYS_CONTEXT variables.

The idea behind is to stop masking the data at the application level and do it in a more secure way, at the database level. We can imagine a front-end where some private information is displayed – identification number, account number or even credit card number. Data Redaction allows to display only part of the information. There are also situations that, depending on certain condition, we might want to completely hide or show random information. This new feature is made for that.

It is designed to protect the low-privileged users (application users) and applies by default to all the DB users, with exception of the those who had been granted the EXEMPT REDACTION POLICY privilege.

Data Redaction versus...

Data Redaction is similar but different to several other features also present on the Enterprise Edition. Here we make a short overview of those dissimilarities:

Virtual Private Database (VPD) – allows to control data accesses at row and column level, by adding a dynamic WHERE clause to a SQL statement. VPD always redacts the data to NULL value. When the application does not support that, then one should use Data Redaction. Virtual Private Database is a free option of Enterprise Edition;

Oracle Label Security – Lets to add user defined labels to rows (for example: 'sensitive', 'private') and use Virtual Private Database to control access based on labels;

Transparent Sensitive Data Protection – Permits to create sets of columns with the same sensitive type (like credit card number) on the database level. Data Redaction is used on the policies for masking sets of columns the same way across a database;

Data Masking – permanently masks the data, which is suitable for non-production environments. Also the masking is not made real time but before in a staging environment. The data is masked for all users, including the high-privileged ones. Advanced Data Masking features like shuffling the data of a column and keep foreign keys relationship, are not part of Data Redaction.

Database Vault – Data Redaction does not prevent high privileged users like DBAs to see the contents of columns being redacted. You can use Data Vault realm to prevent users with EXCEPT REDACTION POLICY to see object data.

Licensing

As most of the Oracle Database features, this one is available only with Enterprise Edition and requires the acquisition of the Advanced Security Option. Oracle will register the usage of Data Redaction option if you use the DBMS_REDACT package.

How it works

The package used to create redaction rules is the new DBMS_REDACT package. It includes five procedures to manage the redaction policies and one procedure to change the default values of a full redaction policy. More information about these functions can be found later on.

After selecting the column you want to have the data redacted, you choose how you want to have it masked. There are four possible options: none, full, partial, random, regexp. The "none" does not redact the data. "Full" option fully

redacts the data. The default masking values are described on Table 1. The “random” redaction shows a random string, number or date. While with Data Masking you can randomize from existing values in the column, in the case of Data Redaction the random values are independent from what is there. The format of the random values is shown on Table 2. The last possible option – “regexp” – allows the use of regular expressions both for searching and displaying the redacted data.

The supported datatypes are: NUMBER, BINARY_FLOAT, BINARY_DOUBLE, CHAR, VARCHAR2, NCHAR, NVARCHAR2, DATE, TIMESTAMP, TIMESTAMP WITH TIME ZONE, BLOB, CLOB and NCLOB.

Full redactions	
character datatypes	single space
number datatypes	0
datetime datatypes	01-JAN-01

Table 1 - Full redaction default values

Random redactions	
character datatypes	random string, max length of the type
number datatypes	random number, size number of digits as original
datetime datatypes	random date, never same as original

Table 2 - Random redaction values format

DBMS_REDACT package

DBMS_REDACT.ADD_POLICY – is the main procedure of the package. The possible input variables are described in the image below.

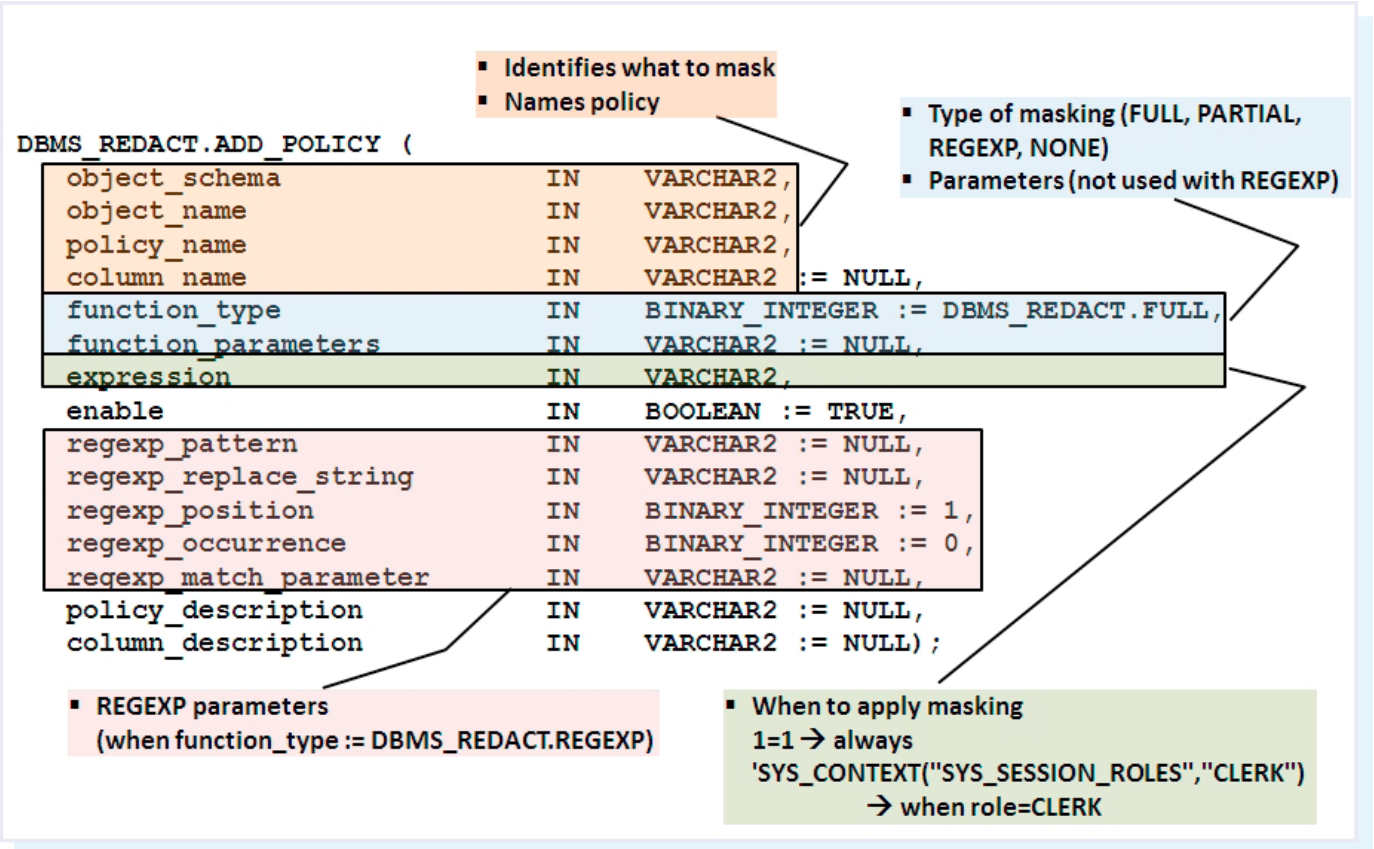

Table 3 - Redaction policy parameters

Other functions are:

DBMS_REDACT.ALTER_POLICY – allows to do change existing policies by doing one of the following:

- By changing the policy expression
- By changing the type of redaction for a specified column
- By changing the parameters to the redaction function for a specified column
- By adding a column to the redaction policy (the redaction type and any parameters must be specified).
- By removing a column from the redaction policy

DBMS_REDACT.DISABLE_POLICY – disables existing policy
DBMS_REDACT.DROP_POLICY – drops existing policy
DBMS_REDACT.ENABLE_POLICY – enables existing policy
DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES – changes the default values for full redaction. Requires the restart of the database to activate.

Dictionary views

Information about the policies can be found within the dictionary views:
REDACTION_COLUMNS, REDACTION_POLICIES and REDACTION_VALUES_FOR_TYPE_FULL

Data Redaction and Data Pump utility

The DATAPUMP_EXP_FULL_DATABASE role includes the powerful EXEMPT REDACTION POLICY system privilege. This role part of the DBA role.

For that reason, as DBA, when you export tables for which the data is being redacted, you are exporting the real – not masked - data. However the redaction policies will be also exported. After re-importing the data, the data redaction policies will be also enable for the same conditions.

In order to export the metadata related to the data redaction policies, you can use the following object paths within the INCLUDE parameter of the expdp command:

CONTENT=METADATA_ONLY
INCLUDE=RADM_FPTM,RADM_POLICY
This can be useful to deploy the redaction policies across different environments.

Examples

Example 1 - regular expressions

The first example shows how to create a policy using expression filters that masks the three first characters of a column called **MASKED** on the table **TEST_TABLE** belonging to **TEST_USER**:

```
BEGIN
 DBMS_REDACT.ADD_POLICY (
   object_schema          => 'TEST_USER',
   object_name            => 'TEST_TABLE',
   column_name            => 'MASKED',
   policy_name            => 'redact_regexp_objname',
   function_type          => DBMS_REDACT.REGEXP,
   expression             => '1=1',
   regexp_pattern         => '\w{3}_(\w+)',
   regexp_replace_string => 'xxx_\1');
END;
/
```

The policy affects all the users (**expression => '1=1'**); searches values in the form "3 letters followed by an under-score" like DBA_TABLES (**regexp_pattern => '\w{3}_(\w+)'**); and replaces the first 3 characters with 'xxx', leaving the rest unchanged (**regexp_replace_string => 'xxx_\1'**). Below the output of a query against that table.

```
create table test_user.test_table as
  (select object_name original, object_name masked from all_objects
   where object_name like ('___\_%') escape '\')

connect test_user/test_user

select original, masked from test_table where rownum < 2;

ORIGINAL                       MASKED
------------------------------ ------------------------------
ALL_TABLES                     xxx_TABLES
ALL_OBJECT_TABLES              xxx_OBJECT_TABLES
```

Example 2 - full redaction

This example shows the full redaction usage and defining exceptions when not applying the policy:

```
BEGIN
 DBMS_REDACT.ADD_POLICY(
   object_schema=> 'TEST_APP',
   object_name  => 'SALARY',
   column_name  => 'SALARY',
   policy_name  => 'redact_full_salary',
   function_type=> DBMS_REDACT.FULL,
   expression    => 'SYS_CONTEXT(''USERENV'',''SESSION_
USER'')!=''BOSS''');
END;
/
```

This policy fully masks the column by replacing the value with a 0, as the database is a number (**function_type => DBMS_REDACT.FULL**) and is applied to all users except BOSS (**expression => 'SYS_CONTEXT(''USERENV'',''SESSION_USER'')!=''BOSS'''**)).

```
insert into test_app.salary values ('General Guisan', 75000);
insert into test_app.salary values ('Henri Dunant', 101000);

connect clerk/clerk;
select * from test_app.salary;

NAME                           SALARY
------------------------------ ----------
Henri Dunant                            0
General Guisan                          0

connect boss/boss;
select * from test_app.salary;

NAME                           SALARY
------------------------------ ----------
Henri Dunant                       101000
General Guisan                      75000
```

Good to know

There are some aspects that worth to be aware when you will start using Data Redaction.

1. Data Redaction does not apply to the where clause, so do not redact data which can be queried in the where clause! Someone willing to discover the value behind a column can use trial and error to discover it, if it has access to change the search condition, as it is presented below:

```
-- example 2, user clerk

select * from test_app.salary where salary > 100000;

NAME                           SALARY
------------------------------ ----------
Henri Dunant                            0
```

2. When using views, functions or procedures to access the data, it remains redacted:

```
-- example 2, user clerk

select max(salary) from test_app.salary;

MAX(SALARY)
-----------
          0
```

3. The results of a random redaction are different each time they are queried and can be confused to real values:

```
-- RANDOM redaction policy

select name, salary from salary;

NAME                           SALARY
------------------------------ ----------
Henri Dunant                   127860
General Guisan                  98423
```

4. Random redaction can show impossible values

```
-- RANDOM redaction policy

select min(salary)-max(salary) RESULT from test_app.salary;

RESULT
------
 20453

select name, salary from test_app.salary where salary>100000;

NAME                           SALARY
------------------------------ ----------
Henri Dunant                    69756
```

5. To change the default values of full redactions you use the DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES procedure. However after changing the values it is necessary to restart the database!

Transparent Sensitive Data Protection

Another new security feature of Oracle 12c is basically a complement to Data Redaction and Virtual Private Database. It is a free option available with Enterprise Edition and is better managed using Enterprise Manager than using the actual DBMS packages.

This feature allows to categorize sensitive columns across a database and then to set a policy to each defined category of columns. For instance:

- telephone number columns → policy to redact into a random number
- Address columns → policy to partially hide the data

You use the packages DBMS_TSDP_MANAGE and DBMS_TSDP_PROTECT to manage the categories and policies, like shown on the example below. It is not so straight forward as with Data Redaction, but can be useful to define enterprise wide sensitive data protection strategies.

Contact

Trivadis AG

Miguel Anjo
E-Mail:
miguel.anjo@trivadis.com

Example

First you create a sensitive type:

```
exec DBMS_TSDP_MANAGE.ADD_SENSITIVE_TYPE (
  sensitive_type  => 'address_text_type',
  user_comment    => 'Type for address columns using text');
```

Then you add sensitive columns to the type:

```
exec DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN(
  schema_name        => 'USER1',
  table_name         => 'CUSTOMER',
  column_name        => 'ADDRESS',
  sensitive_type     => 'address_text_type');
```

As next stage you can create a policy to apply to the sensitive type. The example shows how to create a data redaction policy, but you can also create a VPD policy.

```
DECLARE
  redact_feature_options DBMS_TSDP_PROTECT.FEATURE_OPTIONS;
  policy_conditions DBMS_TSDP_PROTECT.POLICY_CONDITIONS;

BEGIN
  redact_feature_options ('expression'):=
      'SYS_CONTEXT(''USERENV'',''SESSION_USER'') =''APPUSER''';
  redact_feature_options ('function_type'):= 'DBMS_REDACT.PARTI-
AL';
  redact_feature_options ('function_parameters'):=
      'STR, VVVVVVVVV,VVVVVVVVV, *,  1, 6';
  policy_conditions(DBMS_TSDP_PROTECT.DATATYPE):= 'VARCHAR2';

DBMS_TSDP_PROTECT.ADD_POLICY (
  policy_name             => 'REDACT_PARTIAL_ADR',
  security_feature        => DBMS_TSDP_PROTECT.REDACT,
  policy_enable_options   => redact_feature_options,
  policy_apply_condition => policy_conditions);
END;
/
```

As last two steps you associate the policy with the sensitive type and enable it:

```
exec DBMS_TSDP_PROTECT.ASSOCIATE_POLICY (
  policy_name             => 'REDACT_PARTIAL_ADR',
  sensitive_type          => 'address_text_type',
  associate               => TRUE);

exec DBMS_TSDP_PROTECT.ENABLE_PROTECTION_TYPE (
  sensitive_type          => 'address_text_type');
```

Conclusion

The Data Redaction feature of Oracle 12c is promising to have success and tests did not show any problem on using them directly on the command line. It adds a real-time layer of security that can help to keep your data secure. Transparent Sensitive Data Protection is a nice complement to simplify the management of both data redaction and virtual private database policies. ■