NEWSLETTER
www.datum.com.gt

DATUM

Newsletter Marzo 2015
Edición Trimestral

Contenido

Página:

Enmascarando Datos Confidenciales con Oracle Data Redaction

Por Ing. Juan Luis Rodríguez
jrodriguez@datum.com.gt

Oracle Data Redaction permite agregar una máscara a los datos devueltos por consultas a la base de datos, hechas desde cualquier aplicación. Con esto, la información no se muestra tal cual es, sino es reemplazada por otros caracteres (incluyendo el espacio en blanco). Por ejemplo, consultas a la información de un empleado podría exponer datos confidenciales como, salario, número de DPI, fecha de nacimiento, etc.

Oracle Data Redaction permite enmascarar la información por medio de los siguientes métodos:

- **FULL REDACTION**. Enmascarar todos los datos de la columna. La máscara a aplicar depende del tipo de datos. Así, columnas tipo NUMBER son enmascaradas con cero (0) y en columnas tipo VARCHAR2 los datos son enmascarados con espacio.

- **PARTIAL REDACTION**. Enmascarar una parte de los datos en la columna. Por ejemplo, cambiar los cuatro primeros números del DPI por asterisco (*), excepto los siguientes dígitos.

- **REGULAR EXPRESSIONS**. Se puede usar expresiones para buscar patrones. Por ejemplo, al enmascarar direcciones de e-mail se debe tomar en cuenta que el número de caracteres puede variar. Este método aplica solamente para datos de tipo caracter.

- **RANDOM REDACTION**. La información enmascarada se presenta como un valor generado al azar cada vez que es desplegada, dependiendo del tipo de datos de la columna.

¿Cómo funciona Oracle Data Redaction?

Oracle aplica la “redacción” o máscara a los datos en tiempo de ejecución, es decir, antes de ser desplegados en pantalla. Todo el procesamiento de datos es realizado normalmente.

¿Qué se necesita para usar Oracle Data Redaction?

Se requiere tener instalada la versión de base de datos 11.2.0.4 o superior. Se debe contar con la opción Oracle Advanced Security, esto se puede verificar a través la siguiente consulta a la base de datos:

```
SQL> select parameter, value from v$option where upper(parameter) like
'%SECURITY%';

PARAMETER                      VALUE
------------------------------ ----------
Enterprise User Security       TRUE
Oracle Label Security          FALSE
```

El valor esperado es TRUE en la columna VALUE para el parámetro Enterprise User Security. Con esto se confirma la opción requerida en la base de datos.

¿Cómo se usa Oracle Data Redaction?

El uso de Oracle Data Redaction implica elegir los datos o columnas de tablas a enmascarar en el despliegue de la aplicación y, hacer uso del paquete DBMS_REDACT para establecer las políticas y tipo de enmascarado (Full, Partial, Regular Expression, Random) para la columna.

5a Av. 5-55 Zona14, Europlaza World Business Center, Torre II, Nivel 12
PBX (502) 2364-5300 Fax (502) 2364-5311
newsletter@datum.com.gt

Oracle Data Redaction en Acción !

Para obtener ejemplos de los tipos de enmascarado antes mencionados, tomaremos la tabla EMP del esquema SCOTT.

1) FULL REDACTION. Para la columna SAL (Salario). Normalmente, un consulta a esta tabla se muestra así:

```
SQL> select empno, ename, hiredate, sal, deptno
   2 from emp where rownum < 6;

     EMPNO ENAME      HIREDATE          SAL     DEPTNO
---------- ---------- --------- ---------- ----------
      7369 SMITH      17-DEC-80         800         20
      7499 ALLEN      20-FEB-81        1600         30
      7521 WARD       22-FEB-81        1250         30
      7566 JONES      02-APR-81        2975         20
      7654 MARTIN     28-SEP-81        1250         30
```

Enmascarando la columna SAL:

```
SQL> begin
     dbms_redact.add_policy(
     object_schema => 'SCOTT',
     object_name => 'EMP',
     column_name => 'SAL',
     policy_name => 'redact_emp_sal',
     function_type => DBMS_REDACT.FULL,
     expression => '1=1');
   end;
 /

PL/SQL procedure successfully completed.
```

Consultando nuevamente para ver el efecto de enmascarar la columna SAL:

```
SQL> select empno, ename, hiredate, sal, deptno
   2 from emp where rownum < 6;

     EMPNO ENAME      HIREDATE          SAL     DEPTNO
---------- ---------- --------- ---------- ----------
      7369 SMITH      17-DEC-80          0         20
      7499 ALLEN      20-FEB-81          0         30
      7521 WARD       22-FEB-81          0         30
      7566 JONES      02-APR-81          0         20
      7654 MARTIN     28-SEP-81          0         30
```

En la salida anterior, la columna SAL es totalmente enmascarada, valor en columna SAL se cambia por 0 (cero). Al aplicar método FULL REDACTION para columnas de tipo VARCHAR2 se obtiene espacio en blanco.

Procedemos a borrar la política de enmascarado para definir otro ejemplo.

```
SQL> begin
        dbms_redact.drop_policy(
        object_schema => 'SCOTT',
        object_name => 'EMP',
        policy_name => 'redact_emp_sal');
      end;

   PL/SQL procedure successfully completed.
```

2) PARTIAL REDACTION. En este ejemplo, se definirá una política de enmascarado para reemplazar el valor de la columna SAL por un valor falso. Para ello, definimos la política:

```
SQL> begin
  dbms_redact.add_policy(
  object_schema => 'SCOTT',
  object_name => 'EMP',
  column_name => 'SAL',
  policy_name => 'redact_emp_sal',
  function_type => DBMS_REDACT.PARTIAL,
  function_parameters => '9,1,4',
  expression => '1=1');
end;
/

PL/SQL procedure successfully completed.
```

Al ejecutar la consulta a los datos obtenemos:

```
SQL> select empno, ename, hiredate, sal, deptno
   2 from emp where rownum < 6;

     EMPNO ENAME      HIREDATE          SAL     DEPTNO
---------- ---------- --------- ---------- ----------
      7369 SMITH      17-DEC-80        999         20
      7499 ALLEN      20-FEB-81       9999         30
      7521 WARD       22-FEB-81       9999         30
      7566 JONES      02-APR-81       9999         20
      7654 MARTIN     28-SEP-81       9999         30
```

Procedemos a borrar la política de enmascarado.

```
SQL> begin
      dbms_redact.drop_policy(
      object_schema => 'SCOTT',
      object_name => 'EMP',
      policy_name => 'redact_emp_sal');
    end;

PL/SQL procedure successfully completed.
```

3) RANDOM REDACTION. En el siguiente ejemplo, se hará el enmascarado de la columna SAL (Salario) con números aleatorios. Definimos la política:

```
SQL> begin
      dbms_redact.add_policy
      ( object_schema => 'scott',
        object_name => 'emp',
        policy_name => 'emp_redact_sal',
        column_name => 'sal',
        function_type => dbms_redact.random,
        expression => '1=1' );
     end;

PL/SQL procedure successfully completed.
```

Al consultar nuevamente, obtenemos los valores del salario, cambiados así:

```
SQL> select empno, ename, hiredate, sal, deptno from emp where rownum
< 6;

     EMPNO ENAME      HIREDATE          SAL     DEPTNO
---------- ---------- --------- ---------- ----------
      7369 SMITH      17-DEC-80         83         20
      7499 ALLEN      20-FEB-81       1412         30
      7521 WARD       22-FEB-81         38         30
      7566 JONES      02-APR-81       1520         20
      7654 MARTIN     28-SEP-81        889         30
```

Finalmente, borramos la política de enmascarado:

```
SQL> begin
      dbms_redact.drop_policy(
      object_schema => 'SCOTT',
      object_name => 'EMP',
      policy_name => 'redact_emp_sal');
    end;

PL/SQL procedure successfully completed.
```

4) REGULAR EXPRESION. En este ejemplo, se enmascaran algunos caracteres al consultar la columna ENAME, haciendo visibles solo los tres primeros:

Creamos la política:

```
SQL> BEGIN
    DBMS_REDACT.ADD_POLICY(
      object_schema => 'scott',
      object_name => 'emp',
      column_name => 'ename',
      policy_name => 'redact_emp_ename',
      function_type => DBMS_REDACT.REGEXP,
      expression => '1=1',
```

```
      regexp_pattern => '(\S{3})(\S+)',
      regexp_replace_string => '\1***');
 END;
 /

PL/SQL procedure successfully completed.
```

Ejecutando la siguiente consulta obtenemos:

```
SQL> select empno, ename, hiredate, sal, deptno
   2 from emp where rownum < 6;

     EMPNO ENAME           HIREDATE          SAL     DEPTNO
---------- --------------- --------- ---------- ----------
      7369 SMI***          17-DEC-80        800         20
      7499 ALL***          20-FEB-81       1600         30
      7521 WAR***          22-FEB-81       1250         30
      7566 JON***          02-APR-81       2975         20
      7654 MAR***          28-SEP-81       1250         30
```

Finalmente, borramos la política de enmascarado:

```
SQL> begin
        dbms_redact.drop_policy(
        object_schema => 'SCOTT',
        object_name => 'EMP',
        policy_name => 'redact_emp_ename');
      end;

PL/SQL procedure successfully completed.
```

CONCLUSIONES

- Oracle Data Redaction es transparente. No es necesario realizar cambios en las aplicaciones para enmascarar la información. Como se vio en los ejemplos anteriores, la configuración se realiza en los objetos (columnas) a nivel de base de datos.

- El role DATAPUMP_EXP_FULL_DATABASE incluye el privilegio EXEMPT REDACTION POLICY, permitiendo con ello “exportar” los datos reales, no enmascarados (utilitario expdp).

- Si se requiere un usuario, o un grupo de usuarios que no se vean afectados por las políticas de redacción, basta con asignarles el privilegio EXEMPT REDACTION POLICY.

Decisiones: Desarrollo Web Utilizando Oracle Forms u Oracle ADF

Por Ing. Daniel Caciá
dcacia@datum.com.gt

Al inicio de cualquier proyecto, luego del levantado de requerimientos, se hace necesario decidir cuál es la mejor tecnología para implementarlo. Muchas veces esta decisión se toma una sola vez, debido entre otras cosas a la curva de aprendizaje que conlleva cambiar de tecnología o bien a las opciones de productos de desarrollo que ofrece nuestro proveedor.

Hace unos años, Oracle decidió fomentar el desarrollo de aplicaciones Web, remplazando la arquitectura cliente/servidor. Este cambio se hizo evidente a partir de la versión de Oracle Forms 9i. Hoy en día ya existe la versión 11g que continua con el mismo paradigma.

La migración de Oracle Forms 6i a 10g u 11g no acarrea muchas complicaciones, más allá de pequeñas o casi ninguna modificación a cierta lógica de programación. Por lo anterior es evidente que los sistemas que ya estaban desarrollados en 6i pueden ser utilizados en el Web de una manera casi transparente para los usuarios finales y programadores.

Actualmente Oracle ofrece varias opciones para desarrollar aplicaciones Web, entre las más populares se encuentran Oracle Forms y Oracle ADF. Ambas herramientas permiten desarrollar aplicaciones de manera rápida que permiten manipular los datos almacenados. En base a lo anterior surge la necesidad de replantearse que tecnología nos permitiría sacar el mayor beneficio para utilizarla al momento de desarrollar nuevas aplicaciones.

La intención de este artículo es dar a conocer una serie de criterios, características y ventajas de utilizar una u otra tecnología. Partamos de la siguiente premisa: Oracle ADF no sustituirá jamás a Oracle Forms ni viceversa, más bien se complementan. Existen cierto tipo de aplicaciones que por su naturaleza deberían ser programadas en una u otra herramienta.

Comencemos describiendo de manera puntual cada una de estas herramientas de desarrollo. Oracle Forms es un ambiente RAD (Desarrollo rápido de aplicaciones, por sus siglas en inglés) de cuarta generación. Oracle Forms Builder es utilizado para crear aplicaciones que permitan insertar, acceder, cambiar o eliminar datos almacenados en una base de datos Oracle (u otra).

Oracle Application Development Framework (Oracle ADF), provee un framework de Java que permite desarrollar aplicaciones empresariales que interactúen con la base de datos. Oracle ADF permite el desarrollo visual y declarativo para aplicaciones Java EE. Es una tecnología RAD, ya que ofrece asistentes, patrones de diseño listos para usar y herramientas visuales basadas en metadata para facilitar el desarrollo.

El término RAD no se utiliza a la ligera, ambas herramientas de desarrollo ofrecen un ambiente sencillo de utilizar para construir aplicaciones Web, guiado por asistentes y complementado por el uso de propiedades. Ambas ofrecen un ambiente visual de desarrollo, construcción de “pantallas” a partir de tablas y la posibilidad de codificar procedimientos o funciones para aumentar o modificar el comportamiento por defecto que se ofrece (inserciones, actualizaciones, eliminaciones y consultas).

Ambas herramientas permiten desarrollar aplicaciones de forma sencilla, completamente funcionales, pero cuya forma de ejecución difiere grandemente y que podría ser la razón por la que nos decidamos por una u otra.

Describamos algunas de las diferencias. Oracle Forms permite agregar código de programación utilizando lenguaje PL/SQL mientras que Oracle ADF utiliza Java. La idea de poner al equipo de IT a aprender un nuevo lenguaje y herramienta puede ser un pensamiento descabellado, ya que requiere invertir tiempo. Pero lo anterior no debe ser un factor determinante por sí solo.

Oracle ha realizado un esfuerzo grande para que los ambientes de desarrollo Forms Builder y JDeveloper (herramienta que utilizamos para desarrollar aplicaciones Oracle ADF) sean bastante parecidos y la experiencia de cambiar entre uno y otro no sea demasiado complicada. Además, el código de Java que utilizamos para personalizar la aplicaciones Oracle ADF que desarrollemos es bastante reducido y por lo tanto sólo es necesario tener conocimiento básico de Java.

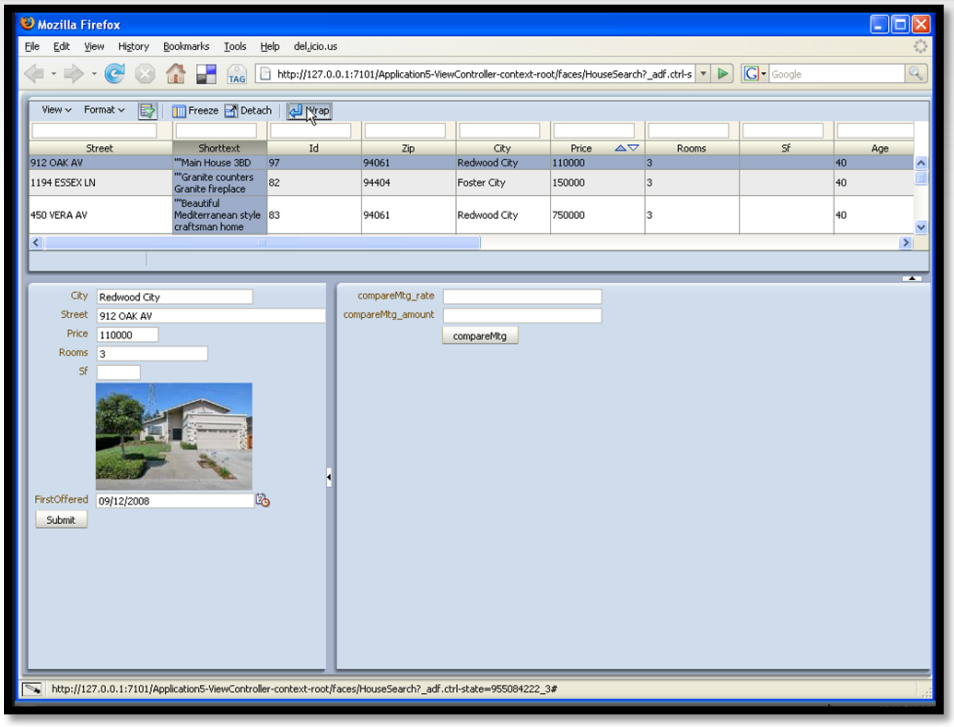
Otra diferencia entre las herramientas es la forma de ejecución. Oracle Forms ejecuta la lógica de la aplicación (triggers) en el servidor de aplicaciones, mientras que la lógica de navegación y de presentación es ejecutada en el cliente. Oracle Forms utiliza applets como GUI. Por otra parte, Oracle ADF ejecuta tanto la lógica de la aplicación, como la lógica de presentación en el servidor de aplicación, utilizando HTML para el GUI.

La interfaz de usuario que utiliza Oracle Forms es un applet, requiere que el explorador tenga instalado un plug-in para poder visualizar la forma, como el *jinitiator* de Oracle o el *jre* de Sun. A las aplicaciones Oracle Forms se les conoce como “pesadas” debido a que el tiempo de descarga e inicialización de los applets consume bastante tiempo y recursos del cliente.

Oracle ADF utiliza código HTML para la interfaz del usuario, este código es generado en el servidor e interpretado por el explorador del cliente. No es necesario tener un plug-in del lado del cliente para visualizar la aplicación, por lo que la aplicación se considera liviana.

En las siguientes imágenes se muestra el GUI de ambas tecnologías.

Oracle Forms GUI 1



Oracle ADF GUI 2

Mi opinión es que ambas soluciones son razonables, pero para diferente tipo de aplicaciones y diferentes tipos de usuarios.

La realidad es que hacer que una página web haga algo realmente interesante es todo un reto. Es muy complejo, por ejemplo, que cuando se llene un campo, inmediatamente otro campo muestre una descripción. Actualmente existen tecnología como AJAX, JavaScripts y páginas JSF que han hecho que esta situación sea más sencilla de implementar.

Para muchas aplicaciones (ej. consulta de saldos, listado de productos, compra de boletos) el usuario no necesita una interfaz de usuario muy poderosa. Además, probablemente deseen utilizar la aplicación desde un café internet o utilizando la computadora de algún compañero, a estos usuarios se les conoce como “breves o pasajeros”. Si el propósito del sitio web es satisfacer a este tipo de usuarios, entonces Oracle ADF es la forma más apropiada de hacerlo.

Si los usuarios utilizan aplicaciones de largo plazo (que permanezcan abiertas todo el día), como un cajero o un punto de venta, utilizar aplicaciones HTML+Javascript no será la mejor opción, debido a que si se está utilizando esta tecnología agregar características especiales es bastante complejo, como un “bip” cuando un valor de un campo no sea el apropiado al momento de ingresarlo.

Como se mencionó anteriormente, Oracle ha logrado bastante avances con Oracle ADF, especialmente aprovechando las características de los Oracle ADF Faces Components para páginas JSF, estos componentes hacen que las características que antes eran difíciles de programar utilizando HTML+Javascript sean mucho más sencillas y rápidas de programar. Los Oracle ADF Faces Components permite desarrollar una interfaz del usuario casi tan poderosa como lo ofrecido por los applets.

Por último, debemos considerar el modelo “desconectado” con el que se ejecutan las aplicaciones Oracle ADF. Utilizando esta tecnología no es posible que el servidor le envíe mensajes a la página HTML para que se refresque, la página HTML solo puede saber que el modelo de datos cambió o que ocurrió un evento en el servidor hasta que se hace un *Submit* de la página hacia el servidor.

Por el contrario, Oracle Forms debe mantener siempre abierto un canal de comunicación entre el servidor y el cliente para mantener la sincronización con el modelo de datos, esto quiere decir que el servidor sí es capaz de indicarle a las formas que un evento ocurrió y por lo tanto debe refrescar su interfaz gráfica.

Oracle Forms necesita un ancho de banda robusto, mientras que Oracle ADF puede aprovechar un enlace más sencillo.

La decisión final depende del tipo de aplicación que se desea crear, pero sobre todo del tipo de usuario que utilizará nuestra aplicación: si son empleados a los que se les puede capacitar en el uso de la aplicación, si son usuarios externos, o bien si son usuarios “breves” o usuarios que requieren que la información en sus pantallas se esté refrescando inmediatamente, o si se trata de usuarios que no quieren usar el mouse (digitadores), etc.

TIP TÉCNICO DEL MES:

Consulta Oracle SQL que muestra el número de conexiones actuales a Oracle agrupado por aplicación que realiza la conexión

```
select program Aplicacion, count(program) Numero_Sesiones
from v$session
group by program
order by Numero_Sesiones desc;
```

Por Lic. Francisco Barrundia
fbarrundia@datum.com.gt

5a Av. 5-55 Zona14, Europlaza World Business Center, Torre II, Nivel 12
PBX (502) 2364-5300 Fax (502) 2364-5311
newsletter@datum.com.gt