Prácticas de LOBS

PL/SQL 12c-18c avanzado

NOTA: Aunque siempre pongo las soluciones, os recomiendo que primero intentéis hacer el ejemplo por vosotros mismos y luego lo comparéis con el mío. ¡¡¡En muchas ocasiones, el mismo proceso se puede hacer de varias formas!!!!

LOBS

1. Añadir una nueva columna a la tabla clientes, usada en las prácticas anteriores, denominada FOTO_BLOB de tipo BLOB

```
ALTER TABLE CLIENTES ADD FOTO_BLOB BLOB;


DESC CLIENTES


Nombre      ¿Nulo? Tipo
----------- ------ -------------
CODIGO            NUMBER
NOMBRE            VARCHAR2(100)
FOTO              BLOB
COMENTARIOS       CLOB
FOTO_BLOB         BLOB
```

2. Crear un procedimiento que copie de la columna FOTO a la columna FOTO_BLOB mediante los métodos READ y WRITE del paquete DBMS_LOB.
    - Podemos usar las filas generadas en la práctica anterior
    - No usamos variables intermedias, trabajamos directamente con las dos columnas
    - No es necesario hacer el OPEN ni el CLOSE al tratar directamente con las columnas.
    - Podemos usar el método DBMS_LOB.GETLENGTH para averiguar el tamaño de la imagen
3. Comprobar con SqlDeveloper que la imagen se ha copiado satisfactoriamente

Ejemplo:

```
CREATE OR REPLACE PROCEDURE CARGA_FOTO
IS
   CURSOR CLI IS SELECT * FROM CLIENTES FOR UPDATE;
   TEMPORAL BLOB;
   CANTIDAD INTEGER:=0;
   POSICION INTEGER:=1;
```

```
BEGIN
   FOR C1 IN CLI LOOP
      CANTIDAD:=DBMS_LOB.GETLENGTH(C1.FOTO);
      DBMS_LOB.READ(C1.FOTO,CANTIDAD,POSICION,TEMPORAL);
      DBMS_LOB.WRITE(C1.FOTO_BLOB,CANTIDAD,POSICION,TEMPORAL);
   END LOOP;
END;
/
EXECUTE CARGAR_CLIENTE(2,'JUAN');
EXECUTE CARGA_FOTO;

SELECT * FROM CLIENTES;

COMMIT;
```

4. Crear una función que pasando el código del cliente guarde la imagen de la columna FOTO_BLOB en un fichero denominado cliente-"código_cliente".png
   - Debe devolver TRUE si ha funcionado y FALSE si se produce alguna excepción si no se ha podido grabar por algún motivo.
   - Usamos el directorio creado en la práctica inicial de LOBS, que denominado IMAGENES.
   - Dado que un BFILENAME no puede ser de tipo WRITE, debemos usar el paquete UTL_FILE para escribir, con el método UTL_FILE.PUT_RAW

Ejemplo

```
CREATE OR REPLACE FUNCTION FICHERO_FOTO (COD_CLIENTE NUMBER)
RETURN BOOLEAN
IS
   FICHERO UTL_FILE.FILE_TYPE;
   CANTIDAD INTEGER;
   NOMBRE_FICHERO VARCHAR2(100);
   CLIENTE CLIENTES%ROWTYPE;
BEGIN
   SELECT * INTO CLIENTE FROM CLIENTES  WHERE CODIGO=COD_CLIENTE;

      --CREAR EL NOMBRE DEL FICHERO
      NOMBRE_FICHERO:='foto'|| COD_CLIENTE||'.png';

      --ABRIR EL FICHERO COMO ESCRITURA

FICHERO:=UTL_FILE.FOPEN('IMAGENES',NOMBRE_FICHERO,'wb',DBMS_LOB.GETLENGT
H(CLIENTE.FOTO_BLOB));
```

```
        --VARIALBE CON LA LONGITUD DE LA FOTO
        CANTIDAD:=DBMS_LOB.GETLENGTH(CLIENTE.FOTO_BLOB);

        --ESCRIBIMOS EN EL FICHERO
        UTL_FILE.PUT_RAW(FICHERO,CLIENTE.FOTO_BLOB);

       --CERRAMOS EL FICHEROS
        --);
        RETURN TRUE;
EXCEPTION
   WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLCODE);
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        RETURN FALSE;
END;
/
```

- Para probarlo, podemos usar el siguiente bloque anónimo
-

```
SET SERVEROUTPUT ON
DECLARE
  CORRECTO BOOLEAN;
BEGIN
   CORRECTO:= FICHERO_FOTO(1);
   IF CORRECTO THEN
      DBMS_OUTPUT.PUT_LINE('OK');
   ELSE
      DBMS_OUTPUT.PUT_LINE('ALGO HA FALLADO');
   END IF;
END;
/
```