OBJETOS DE GRAN TAMAÑO (LOB)

Objetos de gran tamaño - Large Objects (LOB)

- Oracle provee un tipo de dato llamado LOB
- Permite almacenar información estructurada y no estructurada como texto , gráficos, audio y video de gran tamaño.
- La información multimedia puede residir tanto en la misma base de datos como archivos del sistema operativo.
- Este tipo de datos se crea en reemplazo a los tipos de datos LONG, RAW y LONG RAW debido a todos las restricciones y problemas de mantenimiento que presentaban.
- La capacidad máxima que un LOB puede albergar es de 4 GB.

Algunas características de los tipos de datos LOB con respecto a los tipos LONG:

- Una tabla puede contener múltiples columnas LOB, pero solamente una columna de tipo LONG.
- El tamaño máximo de un campo LOB es de 4 GB. Los campos LONG solo pueden tener 2 GB
- Los tipos de datos LOB (Excepto NCLOB) pueden ser atributos de un tipo de datos (objeto) definido por el usuario.
- Las tablas con columnas LOB pueden ser replicadas, no así las tablas con columnas LONG.

El tipo de dato LOB

Todo tipo de datos LOB tiene dos partes:

a) LOB Value : el cual constituye el valor a almacenar; por ejemplo: un texto o contenido multimedia.

b) LOB Locator : es un puntero a la ubicación del valor LOB (LOB Value) almacenado en la base de datos.

- Si el texto o información multimedia se guarda dentro de la base de datos, el contenido se almacena en un segmento separado de la tabla. Este segmento es de tipo LOB y almacena solo el LOB Value; mientras la tabla que se definió con el campo LOB solo lleva el LOB Locator como puntero al segmento LOB.

El tipo de dato LOB

- En la tabla generalmente se almacena un localizador del objeto LOB (LOB locator), y los objetos mismos se almacenan en una ubicación diferente (salvo que explícitamente se indique lo contrario durante la creación)
- Cuando se crea una tabla relacional, se definen como cualquier otro tipo de dato.

Tipos de LOB en ORACLE

Tipo	Descripción
CLOB	Permite almacenar datos de tipo carácter que pertenecen al conjunto de caracteres definido para la base de datos. El tamaño máximo es de **(4GB - 1) *** parámetro de inicio **DB_BLOCK_SIZE** (8TB a 128TB)
NCLOB	Almacena datos del conjunto de caracteres nacional definido para la base de datos (usando el estándar de 2 Bytes UNICODE). El tamaño máximo es igual al del CLOB.
BLOB	Puede contener datos binarios no estructurados, que no son interpretados por la base de datos. El tamaño máximo es también de **(4GB - 1) *** parámetro de inicio **DB_BLOCK_SIZE** (8TB a 128TB)
BFILE	Permiten acceso de sólo lectura a los archivos binarios de gran tamaño almacenados fuera de la base de datos ORACLE. La BD almacena tan solo un puntero al archivo externo, y por tanto todas las modificaciones que se hagan a tales archivos no forman parte de las transacciones de ORACLE. Para acceder a tales datos, se debe crear un tipo especial de objeto llamado DIRECTORIO, que no es más que un alias lógico para una ruta de acceso existente en el sistema de archivos. El tamaño máximo es de **4GB**

El tipo CLOB

- Los CLOB almacenan texto que contienen grandes cantidad de bytes. Reemplaza al tipo de dato LONG.
- Existe automáticamente una conversión implícita entre los CLOB y VARCHAR2.
- Al crear una tabla con un campo CLOB este se debe inicializar con un Locator vacío, no dejarlo NULL.
- Para realizar esto lo hacemos mediante la función EMPTY_CLOB().

El tipo CLOB - Ejemplo

- Durante la creación de la tabla que alberga campos de tipo CLOB, podemos indicar que el LOB Value (contenido) sea almacenado en otro tablespace. Ejemplo:

```
CREATE TABLE TABLA_CLOB
      ( ID          NUMBER,
        DOCUMENTO CLOB DEFAULT EMPTY_CLOB());
```

En otro tablespace:

```
CREATE TABLE TABLA_CLOB2
  ( ID          NUMBER,
    DOCUMENTO CLOB DEFAULT EMPTY_CLOB())
    LOB(DOCUMENTO) STORE AS BASICFILE CTAB
(TABLESPACE BASED2TP );
```

El tipo CLOB - Ejemplo

- Para insertar un valor sobre un campo CLOB lo hacemos tan similar como si fuera un campo VARCHAR2, ejemplo:

```
INSERT INTO TABLA_CLOB2
        VALUES(2 , 'Esto es un texto' ) ;

COMMIT;
```

DBMS_LOB

- El paquete DBMS_LOB proporciona un juego de operaciones para objetos LOB almacenadas en las bases de datos.
- Todas las rutinas toman como argumento al menos un localizador LOB, los cuales pueden ser nulos, o apuntar a objetos LOB inicializados y almacenados en la base de datos.

Subprogramas del paquete DBMS_LOB (Lectura y Escritura)

Subprograma	Descripción
APPEND	Añade los contenidos de un objeto LOB de origen al final de un objeto LOB de destino
COMPARE	Determina si 2 objetos LOB son iguales
COPY	Copia datos desde una posición dada en el objeto LOB de origen a una posición dada en el objeto LOB de destino
ERASE	Borra todo el objeto LOB o una parte del mismo a partir de una posición especificada
GETLENGTH	Devuelve la longitud de un objeto LOB
READ	Devuelve los datos de un objeto LOB a partir de una posición
WRITE	Escribe datos en un objeto LOB a partir de la posición dada
TRIM	Borra datos del final de un objeto LOB
SUBSTR	Devuelve una parte de los datos del objeto LOB a partir de la posición dada
LOADFROMFILE	Lee un determinado número de bytes de una variable definida como BFILE en un objeto de tipo BLOB

Subprogramas del paquete DBMS_LOB (Apertura y cierre)

Subprograma	Descripción
CLOSE	Cierra un objeto LOB abierto con anterioridad
ISOPEN	Determina si un objeto LOB se encuentra abierto
OPEN	Abre un objeto LOB en el modo especificado

Funciones DBMS_LOB para tipos CLOB

Si deseamos obtener una parte del contenido de un CLOB usamos la función DBMS_LOB.SUBSTR(), ejemplo:

```
SELECT DBMS_LOB.SUBSTR(DOCUMENTO,5,12 )
FROM TABLA_CLOB2;
```

Ejemplo: del campo valor se obtiene 5 caracteres desde la posición 12 del texto.

Funciones DBMS_LOB para tipos CLOB

Función DBMS_LOB.INSTR(), ejemplo:

```
SELECT DBMS_LOB.INSTR(DOCUMENTO, 'E',1,2)
            FROM TABLA_CLOB;
```

En este ejemplo conseguimos la posición de la letra `'E'` en su segunda ocurrencia a partir del primer carácter del texto.

Funciones DBMS_LOB para tipos CLOB

Para agregar texto a un CLOB se utiliza la función DBMS_LOB.WRITEAPPEND(), sus parámetros son la variable CLOB a modificar, la cantidad de caracteres y el texto a añadir.

```
DECLARE
 v_clob CLOB;
BEGIN
  SELECT documento INTO v_clob
   FROM  TABLA_CLOB
   WHERE  id=1  FOR UPDATE;
   DBMS_LOB.WRITEAPPEND(v_clob,  12, ' ejemplo ' ) ;
   DBMS_OUTPUT.PUT_LINE(v_clob) ;
  COMMIT;
END;
```

Tipo de datos BLOB

- BLOB: Reemplaza al tipo de dato LONG RAW y almacena el contenido multimedia dentro de la base de datos.
- Para trabajar con BLOB y BFILES se requiere de Objetos Directorios en la base de datos. Los Objetos Directorios no son objetos que le pertenecen a un esquema, todos los directorios creados son de propiedad del usuario SYS.
- Se requiere el privilegio CREATE ANY DIRECTORY.
- Los Objetos Directorios serán una referencia a una ubicación de un directorio del sistema operativo.

CREACION DE DIRECTORIO

```
CREATE DIRECTORY nombre_dir AS ruta_dir
```

Ejemplo:

En ambientes UNIX

```
CREATE DIRECTORY utils as '/home/utils';
```

En ambientes WINDOWS/DOS

```
CREATE DIRECTORY mis_blobs AS 'C:\MIS_BLOBS';
```

```
GRANT WRITE, READ ON DIRECTORY MIS_BLOBS TO
 BASEDATOS1;
```

OBJETOS BFILE

Los archivos BFILE se almacenan externamente, y como tal la BD requiere conocer la ubicación en el SO. Para ello se define un directorio

Subprogramas del paquete DBMS_LOB (Objetos BFILE)

Subprograma	Descripción
FILEOPEN	Abre el archivo del sistema operativo asociado a un localizador BFILE dado
FILECLOSE	Cierra un objeto BFILE abierto tras finalizar su procesamiento
FILECLOSEALL	Cierra todos los objetos BFILE abiertos y libera todos los recursos asociados a los mismos
FILEEXISTS	Determina si un archivo dado existe y se puede leer
FILEGETNAME	Devuelve el directorio y nombre del archivo asociados a un localizador BFILE dado
FILEISOPEN	Determina si el archivo del sistema operativo asociado a un localizador BFILE dado se encuentra abierto

FUNCIONES ESPECIALES

Subprograma	Descripción
BFILENAME	Es una función con dos argumentos ■Nombre de directorio de tipo varchar2 ■Nombre de archivo varchar2 Y devuelve un dato de tipo BFILE Es útil para las sentencias INSERT o UPDATE

Tratamiento de tipo BFILE

```
ALTER TABLE b_empleados ADD foto BFILE;

INSERT INTO B_EMPLEADOS VALUES(767676,'JOSEFINA',
 'ARRUA', SYSDATE,
   TO_DATE('20/02/1970','DD/MM/YYYY'),
  1607843, '33940','4ta.Proyectada',
   'Sajonia',13,
   bfilename('MIS_BLOBS','767676.JPEG'));
  COMMIT;
```

Tratamiento de tipo BFILE

```
DECLARE
   nombrearch VARCHAR2(30);
   arch BFILE;
   CURSOR c_emp IS SELECT * FROM b_empleados
   FOR UPDATE;
 BEGIN
  FOR REG IN  C_EMP LOOP
   BEGIN
    nombrearch := TRIM(BOTH FROM
   TO_CHAR(REG.cedula))||'.jpg';
    arch:= BFILENAME('MIS_BLOBS',nombrearch);
    DBMS_LOB.FILEOPEN(arch);
    UPDATE b_empleados SET foto = arch
            WHERE CURRENT OF c_emp;
    DBMS_LOB.FILECLOSE(arch);
   EXCEPTION
    WHEN OTHERS THEN
          NULL;
   END;
 END LOOP;
END;
/
```

Creación de tablas con columnas BLOB

```
CREATE TABLE  emp_lob
(ID NUMBER,
  NOMBRE VARCHAR2(30),
  DOCUMENTO  BLOB);
```

Procedimiento p/ para insertar una columna BLOB

```
DECLARE
   v_flob BFILE;
   v_blob BLOB;
BEGIN
   INSERT INTO EMP_LOB VALUES
    (1, 'Juan Perez', EMPTY_BLOB())
     RETURNING DOCUMENTO  INTO v_blob;
   v_flob := BFILENAME('MIS_BLOBS','DOCUM01.PDF');
   DBMS_LOB.FILEOPEN(v_flob,
   DBMS_LOB.FILE_READONLY);
   DBMS_LOB.LOADFROMFILE(v_blob, v_flob,
   DBMS_LOB.GETLENGTH(v_flob));
   DBMS_LOB.FILECLOSE(v_flob);
   COMMIT;
END;
/
```

Procedimiento p/ actualizar un campo BLOB

```
INSERT INTO EMP_LOB VALUES(3, 'Pedro Paniagua',
   EMPTY_BLOB())

DECLARE
   v_flob BFILE;
   v_blob BLOB;
BEGIN
   SELECT DOCUMENTO INTO V_BLOB
     FROM EMP_LOB
     WHERE ID = 3 FOR UPDATE OF DOCUMENTO;
   v_flob := BFILENAME('MIS_BLOBS','DOCUM03.PDF');
   DBMS_LOB.FILEOPEN(v_flob,
   DBMS_LOB.FILE_READONLY);
   DBMS_LOB.LOADFROMFILE(v_blob, v_flob,
   DBMS_LOB.GETLENGTH(v_flob));
   DBMS_LOB.FILECLOSE(v_flob);
   COMMIT;
END;
/
```

Ejercicios