INTRODUCCIÓN A LOS TIPOS DE DATOS LOB (Large Objects)

ORACLE

Diseño de Bases de Datos Multimedia

Universitat d'Alacant
Universidad de Alicante

Departamento de
Lenguajes y Sistemas Informáticos

TIPOS DE DATOS LOB

- ¿Qué es LOB? *es un tipo de datos binario que puede contener una cantidad de datos variables. LOB significa "large objetcs" (en castellano, objetos grandes).*

CLOB	Un tipo de datos LOB que puede contener grandes cantidades de caracteres. Son compatibles con el carácter de base de datos establecida, tanto de caracteres de ancho fijo (tipo CHAR) y conjuntos de caracteres de ancho variable (VARCHAR),. El tamaño máximo es (4 gigabytes - 1) * (tamaño del bloque de la base de datos), es decir de 8 Terabytes para bloques de 2K, hasta 128 Terabytes para bloques de 32 K.
NCLOB	Lo mismo que el tipo de datos LOB, pero para caracteres UNICODE. Se usan cuando el juego de caracteres a almacenar no es el mismo que el de la base de datos. Por ello, se usará parte del carácter para indicar el juego de caracteres.
BLOB	Una tipo de datos LOB binario. El tamaño máximo es (4 gigabytes - 1) * (tamaño del bloque de la base de datos). El contenido binario está almacenado en la BD.
BFILE	Contiene un localizador (puntero) a un archivo binario almacenado en una carpeta fuera de la base de datos. El tamaño máximo que puede contener una columna o variable de este tipo es de 4 gigabytes.

TIPOS DE DATOS LOB

- **DBMS_LOB** es un paquete que proporciona ORACLE para trabajar con datos binarios.
- Su uso en PL/SQL es DBMS_LOB.función.
- Algunas de las funciones que podemos utilizar son:
  - **DBMS_LOB.fileopen:** Abre un archivo.
  - **DBMS_LOB.loadfromfile:** Lee un determinado número de bytes.
  - **DBMS_LOB.getlength:** Devuelve el tamaño del archivo en bytes.
  - **DBMS_LOB.fileclose:** Cierra el archivo.

TIPOS DE DATOS BLOB

- **Todas las funciones de DBMS_BLOB son:**

Summary of DBMS_LOB Subprograms

Table 82-9 DBMS_LOB Package Subprograms

Subprogram	Description
APPEND Procedures	Appends the contents of the source LOB to the destination LOB
CLOSE Procedure	Closes a previously opened internal or external LOB
COMPARE Functions	Compares two entire LOBs or parts of two LOBs
CONVERTTOBLOB Procedure	Reads character data from a source CLOB or NCLOB instance, converts the character data to the specified character, writes the converted data to a destination BLOB instance in binary format, and returns the new offsets
CONVERTTOCLOB Procedure	Takes a source BLOB instance, converts the binary data in the source instance to character data using the specified character, writes the character data to a destination CLOB or NCLOB instance, and returns the new offsets
COPY Procedures	Copies all, or part, of the source LOB to the destination LOB
COPY_DBFS_LINK Procedures	Copies the DBFS link in the source LOB to the destination LOB
COPY_FROM_DBFS_LINK	Retrieves the data for the LOB from the DBFS store
CREATETEMPORARY Procedures	Creates a temporary BLOB or CLOB and its corresponding index in the user's default temporary tablespace
DBFS_LINK_GENERATE_PATH Functions	Returns a unique file path name for use in creating a DBFS Link
ERASE Procedures	Erases all or part of a LOB
FILECLOSE Procedure	Closes the file
FILECLOSEALL Procedure	Closes all previously opened files
FILEEXISTS Function	Checks if the file exists on the server
FILEGETNAME Procedure	Gets the directory object name and file name
FILEISOPEN Function	Checks if the file was opened using the input BFILE locators
FILEOPEN Procedure	Opens a file
FRAGMENT_DELETE Procedure	Deletes the data at the specified offset for the specified length from the LOB
FRAGMENT_INSERT Procedures	Inserts the specified data (limited to 32K) into the LOB at the specified offset
FRAGMENT_MOVE Procedure	Moves the amount of bytes (BLOB) or characters (CLOB/NCLOB) from the specified offset to the new offset specified
FRAGMENT_REPLACE Procedures	Replaces the data at the specified offset with the specified data (not to exceed 32k)
FREETEMPORARY Procedures	Frees the temporary BLOB or CLOB in the default temporary tablespace
GET_DBFS_LINK Functions	Returns the DBFS Link path associated with the specified SecureFile
GET_DBFS_LINK_STATE Procedures	Retrieves the current DBFS Link state of the specified SecureFile
GETCHUNKSIZE Functions	Returns the amount of space used in the LOB chunk to store the LOB value
GETCONTENTTYPE Functions	Returns the content ID string previously set by means of the SETCONTENTTYPE Procedure
GETLENGTH Functions	Gets the length of the LOB value
GETOPTIONS Functions	Obtains settings corresponding to the option_type field for a particular LOB
GET_STORAGE_LIMIT Function	Returns the storage limit for LOBs in your database configuration
INSTR Functions	Returns the matching position of the *nth* occurrence of the pattern in the LOB
ISOPEN Functions	Checks to see if the LOB was already opened using the input locator
ISTEMPORARY Functions	Checks if the locator is pointing to a temporary LOB
LOADBLOBFROMFILE Procedure	Loads BFILE data into an internal BLOB
LOADCLOBFROMFILE Procedure	Loads BFILE data into an internal CLOB
LOADFROMFILE Procedure	Loads BFILE data into an internal LOB
MOVE_TO_DBFS_LINK Procedures	Writes the specified SecureFile data to the DBFS store
OPEN Procedures	Opens a LOB (internal, external, or temporary) in the indicated mode
READ Procedures	Reads data from the LOB starting at the specified offset
SET_DBFS_LINK Procedures	Links the specified SecureFile to the specified path name. It does not copy the data to the path
SETCONTENTTYPE Procedure	Sets the content type string for the data in the LOB
SETOPTIONS Procedures	Enables CSCE features on a per-LOB basis, overriding the default LOB column settings
SUBSTR Functions	Returns part of the LOB value starting at the specified offset
TRIM Procedures	Trims the LOB value to the specified shorter length
WRITE Procedures	Writes data to the LOB from a specified offset
WRITEAPPEND Procedures	Writes a buffer to the end of a LOB

Acceso a documentación sobre DBMS_BLOB

TIPOS DE DATOS LOB: Insertar datos (I)

- La función **EMPTY_BLOB() o EMPTY_CLOB()** nos devuelve un puntero vacío que nos permitirá insertar un valor en un campo BLOB, CLOB y NCLOB.

Ejemplo 1:

- Supongamos la tabla T con las columnas (id Number, Texto CLOB)
- El código ORACLE para insertar un valor en esa tabla sería:

```
Declare
MiPuntero CLOB;
Begin
insert into tabla (id, Texto)  VALUES(9244, empty_clob()) returning Texto into MiPuntero;
dbms_lob.write( MiPuntero, longitud texto a añadir, posición desde la que se inserta , texto a insertar);
……
```

TIPOS DE DATOS LOB: Insertar datos (II)

- La función **BFILENAME()** crea un puntero que asocia un archivo existente en el sistema de ficheros del servidor con el ORACLE DIRECTORY (carpeta) que lo contiene.

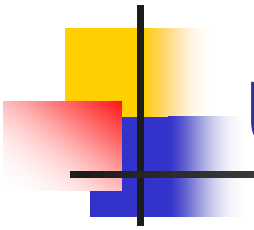
  Ejemplo 2:

- Supongamos tabla T con campos (id Number, foto BFILE)
- Supongamos que la foto pepe.jpg está en la carpeta /img/fotos del S.O .
- Además, hemos creado el siguiente objeto directorio de ORACLE:

```
CREATE DIRECTORY MIDIR AS '/img/fotos'
```

Entonces, para insertar (asociar ya que en la BD sólo se almacena el puntero) bastará con:

```
insert into tabla (id, Texto) VALUES(9244, BFILENAME('MIDIR','pepe.jpg');
```



Usos prácticos de los tipos de dato LOB

- Cuando almacenemos información multimedia que tengan tamaño grande en nuestras bases de datos, usaremos los tipos de datos LOB.
- Recordad que la información se puede almacenar directamente en la base de datos, o incluir en ella una referencia a dónde se encuentra la información (en una carpeta del sistema operativo, o en una URL).