ÍNDICE

Contenido

Antolín·Muñoz·Chaparro¶
Jefe·de·Proyecto·Sistemas·Informáticos·¶
División·III·de·Aplicaciones·de·Costes·de·Personal·y·Pensiones·Públicas¶
Oficina·de·Informática·Presupuestaria·(DIR3:·EA0027952)¶
Intervención·General·de·la·Administración·del·Estado¶

Antolín·Muñoz·Chaparro¶
Jefe·de·Proyecto·Sistemas·Informáticos·¶
División·III·de·Aplicaciones·de·Costes·de·Personal·y·Pensiones·Públicas¶
Oficina·de·Informática·Presupuestaria·(DIR3:·EA0027952)¶
Intervención·General·de·la·Administración·del·Estado¶

Antolín Muñoz Chaparro
Jefe de Proyecto Sistemas Informáticos
División III de Aplicaciones de Costes de Personal y Pensiones Públicas
Oficina de Informática Presupuestaria (DIR3: EA0027952)
Intervención General de la Administración del Estado

Antolín·Muñoz·Chaparro¶
Jefe·de·Proyecto·Sistemas·Informáticos·¶
División·III·de·Aplicaciones·de·Costes·de·Personal·y·Pensiones·Públicas¶
Oficina·de·Informática·Presupuestaria·(DIR3:·EA0027952)¶
Intervención·General·de·la·Administración·del·Estado¶

PAQUETES

Los paquetes son el tercer tipo de bloque PL/SQL nominado después de procedimientos y funciones que se han visto en el capítulo anterior.

El concepto de paquete ha sido introducido por Oracle de acuerdo a las características que tienen los paquetes de lenguaje ADA. De hecho, muchas de las estructuras del PL/SQL provienen de este lenguaje.

Un paquete es una estructura que permite almacenar juntos una serie de objetos relacionados.

Un paquete tiene 2 partes diferenciadas, la especificación y el cuerpo del paquete.

A diferencia de los procedimientos y funciones, los paquetes no pueden estar contenidos dentro de un bloque local, sólo se pueden almacenar en la base de datos.

Un paquete es en esencia una sección declarativa nominada. Cualquier cosa que pueda incluirse en la sección declarativa de un bloque puede incluirse también en un paquete. Esto abarca procedimientos, funciones, cursores, tipos y variables. Una ventaja de incluir estos objetos en un paquete es la posibilidad de referenciarlos desde otros bloques PL/SQL, con lo que los paquetes permiten disponer de variables globales de PL/SQL.

Especificación o cabecera del paquete

Contiene la información acerca del contenido del paquete, pero no el código de ejecución de los procedimientos y funciones que se definen en él.

La sintaxis es la siguiente:

```
CREATE [OR REPLACE] PACKAGE nombre_paquete {IS | AS}
        especificación_procedimiento |
        especificación_función |
        declaración_variable |
        definición_tipo |
        declaración_excepción |
        declaración_cursor
END;
```

Los elementos de la cabecera de un paquete pueden aparecer en cualquier orden salvo porque el elemento referenciado tiene que ser definido antes de utilizarse.

Las declaraciones de procedimientos y funciones sólo incluirán la cabecera de los mismos, donde se indica el tipo de objeto (procedimiento o función), el nombre y los parámetros que utilizan (nombre, tipo y utilización del parámetro).

Las variables y tipos definidos en la cabecera del paquete se podrán utilizar en cualquiera de los procedimientos y funciones que se utilicen en el mismo.

Así mismo esos tipos podrán utilizarse por procedimientos y funciones ajenos al paquete, siempre y cuando se referencie el nombre del paquete y el tipo o variable a utilizar del mismo.

EJEMPLO

```
CREATE OR REPLACE PACKAGE paqueteA IS
      FUNCTION calculaA (param1  NUMBER) RETURN VARCHAR2;
      PROCEDURE calculaB(param1 IN OUT NUMBER);
      v_estudiante    students%ROWTYPE;
      TYPE t_nombres IS TABLE OF v_estudiante
            INDEX BY BINARY_INTEGER;
      TYPE t_textos IS TABLE OF VARCHAR2
            INDEX BY BINARY_INTEGER;
END;
```

Cuerpo del paquete

Contiene el código ejecutable de todos los procedimientos y funciones que se hayan definido en el cabecera del paquete.

No se puede compilar el cuerpo sin haber compilado antes la cabecera.

No puede dejarse sin definir el código ejecutable de ninguno de los procedimientos o funciones que se hayan definido en la cabecera porque generaría errores de compilación en el cuerpo.

Si es posible definir procedimientos y funciones privadas al cuerpo del paquete que no se haya definido en la cabecera, lo que permite que los procedimientos del cuerpo llamen a otros que sólo se haya definido en este espacio. Los procedimientos y funciones exclusivos del cuerpo (privados), no podrán ser invocados por objetos externos al paquete.

La sintaxis es la siguiente:

```
CREATE [OR REPLACE] PACKAGE BODY nombre_paquete
{IS | AS}
      especificación_y_codigo_procedimiento |
      especifacion_y_codigo_funcion
      [declaración_de_var_locales]
[BEGIN
      Código_ejecutable_local_al_paquete]
END;
```

EJEMPLO

```
CREATE OR REPLACE PACKAGE BODY paqueteA IS

      FUNCTION calculaA (param1  NUMBER) RETURN VARCHAR2
      IS BEGIN
            RETURN to_char(param1);
      END;

      PROCEDURE calculaB(param1 IN OUT NUMBER) IS
      BEGIN
            param1 := param1 * 10;
      END;
END paqueteA;
```

Referenciando a los paquetes

Cuando se quiere nombrar a un elemento cualquiera de los que aparece en la cabecera del mismo hay que hacerlo utilizando la siguiente sintaxis:

Nombre_del_paquete . nombre_del_elemento [(param1..paramn)]

EJEMPLO

```
SQL> DECLARE
          var1  NUMBER(5);
          var2  VARCHAR2(6);
          var3  paqueteA.t_textos;
     BEGIN
          paqueteA.calculaB(var1);
          var2 := paqueteA.calculaA(var1);
          var3(1)  := var3;
     END;
```

En este ejemplo en primer lugar se define una variable VAR3 del mismo tipo que la variable T_TEXTOS del PAQUETEA. A continuación en la parte ejecutable del bloque se invoca la ejecución del procedimiento CALCULAB perteneciente al PAQUETEA, y también se asigna el resultado de ejecutar la función CALCULAA del PAQUETEA a la variable VAR2.

EJEMPLO

```
SQL> SELECT paquetea.calculaA(10) FROM DUAL;
```

En este otro ejemplo se invoca la ejecución de la función CALCULAA del PAQUETEA, utilizando una instrucción SELECT… FROM DUAL.

Inicialización de un paquete

Cuando se referencia a cualquiera de los elementos de un paquete por primera vez en una sesión, es posible ejecutar de forma predeterminada un código que se arrancará antes de ejecutar el objeto referenciado. Para conseguirlo hay que insertar dicho código ejecutable en la sección BEGIN..END propia del cuerpo del paquete.

EJEMPLO

```
CREATE OR REPLACE PACKAGE BODY paqueteA IS
  var_local     NUMBER(5);

  FUNCTION calculaA(param1      NUMBER DEFAULT var_local)
  RETURN VARCHAR2 IS
  BEGIN
     RETURN to_char(param1);
  END;

  PROCEDURE calculaB(param1 IN OUT NUMBER) IS
  BEGIN
     Param1 := param1 * 10;
  END;
BEGIN
     Var_local := 10;
END;
```

Cuando se invoque por primera vez a cualquiera de los objetos del paquete dentro de una sesión, antes se inicializará la variable VAR_LOCAL a valor 10.

Sobrecarga de paquetes

Dentro de un paquete pueden sobrecargarse los procedimientos y funciones, es decir, puede haber más de un procedimiento o función con el mismo nombre pero con distintos parámetros.

EJEMPLO

```
CREATE OR REPLACE PACKAGE PRUEBAS AS
  PROCEDURE MAS_ESTUDIANTES(CODIGO IN NUMBER);
  PROCEDURE MAS_ESTUDIANTES(NOMBRE    IN VARCHAR2,
                            APELLIDOS IN VARCHAR2);
END PRUEBAS;
```

Este es un ejemplo de sobrecarga de un paquete en el que se han definido dos procedimientos con el mismo nombre pero con distinto número de parámetros, por lo que PL/SQL los considera distintos.

RESTRICCIONES DE LA SOBRECARGA

A continuación se muestran una serie de reglas que restringen la sobrecarga de objetos dentro de un paquete:

1. No se pueden sobrecargar dos subprogramas si sus parámetros sólo difieren en el tipo de parámetro: entrada, salida o entrada/salida. A continuación se muestra un ejemplo erróneo de sobrecarga:

```
CREATE OR REPLACE PACKAGE ERROR_SOBRECARGA AS
     PROCEDURE prueba (p_param IN NUMBER);
     PROCEDURE prueba (p_param OUT NUMBER);
END;
```

2. No se pueden sobrecargar dos funciones basándose sólo en su tipo de dato de retorno. A continuación se muestra un ejemplo erróneo de sobrecarga:

```
CREATE OR REPLACE PACKAGE ERROR_SOBRECARGA AS
      FUNCTION prueba RETURN DATE;
      FUNCTION prueba RETURN NUMBER;
END;
```

3. No se pueden sobrecargar subprograma en los que exista igualdad del número, nombre de los parámetros y familia del tipo de parámetro, y únicamente difiera el tipo del parámetro dentro de la misma familia. A continuación se muestra un ejemplo erróneo de sobrecarga:

```
CREATE OR REPLACE PACKAGE ERROR_SOBRECARGA AS
      PROCEDURE prueba (p_param1  CHAR);
      PROCEDURE prueba (p_param1  VARCHAR2);
END;
```

4. No se pueden sobrecargar subprogramas en los que sólo difiera el nombre del parámetro, sin cambiar el modo o el tipo del mismo. A continuación se muestra un ejemplo erróneo de sobrecarga:

```
CREATE OR REPLACE PACKAGE ERROR_SOBRECARGA AS
      PROCEDURE prueba (p_param1  IN NUMBER);
      PROCEDURE prueba (p_var1    IN NUMBER);
END;
```

Dependencias

Las dependencias nos permiten cambiar el cuerpo del paquete sin tener que cambiar la cabecera del mismo, de forma que no será necesario recompilar otros objetos que dependan de la cabecera del paquete.

Si lo que se recompila o modifica es la cabecera del paquete, el cuerpo de mismo queda automáticamente en estado INVALID puesto que depende ella.

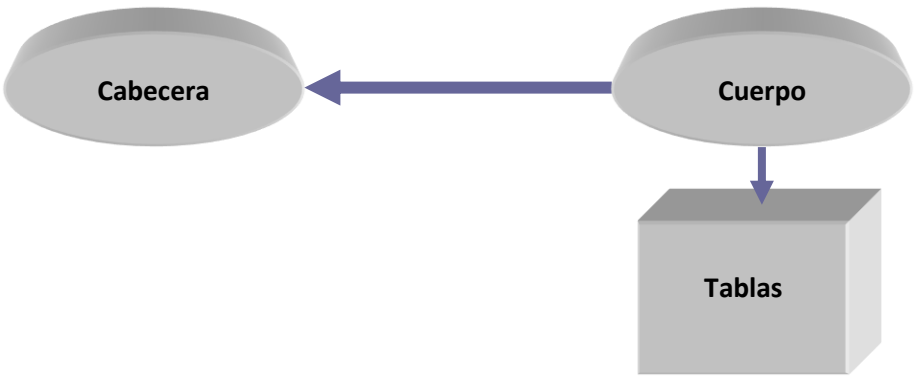

Fig. 8-1 Dependencia entre cabecera y cuerpo de un paquete con las tablas.

Esta figura no muestra la dependencia directa que existe entre la cabecera y el cuerpo de un paquete, así como entre el cuerpo y las tablas de la base de datos.

Repercusión del estado de los objetos con un paquete

Partiendo de la base de las dependencias que existen entre los elementos de un paquete y entre este y las tablas, se pueden presentar las siguientes casuísticas que pueden derivar en un cambio de estado en los objetos dependientes como se indica a continuación:

1. Si se cambia el cuerpo de un paquete sin cambiar la cabecera de procedimientos y funciones incluidos en el mismo, la cabecera del paquete permanecerá como VALID (válida).
2. Si se cambia el cuerpo del paquete cambiando la cabecera de algún procedimiento o función, la cabecera del paquete pasará a estar INVALID (inválida).

3. Si se alteran tablas de la base de datos que afectan al cuerpo del paquete este quedará automáticamente en estado INVALID, mientras que la cabecera del paquete continuará como VALID.
4. Si se borra el cuerpo del paquete, la cabecera permanecerá como VALID.

EJEMPLO DE LA REPERCUSIÓN DEL ESTADO DE OBJETOS

```
SQL> CREATE TABLE SIMPLE
      (campo1    NUMBER(10));

SQL>  CREATE OR REPLACE PACKAGE prueba IS
            PROCEDURE ejemplo (p_val IN NUMBER);
      END prueba;

SQL>  CREATE OR REPLACE PACKAGE BODY prueba IS
            PROCEDURE ejemplo(p_val IN NUMBER) IS
            BEGIN
                  INSERT INTO simple VALUES (p_val);
                  COMMIT;
            END ejemplo;
      END prueba;

SQL>  CREATE OR REPLACE PROCEDURE llamada
      (p_val IN NUMBER) IS
      BEGIN
            Prueba.ejemplo(p_val + 1);
      END;

SQL>  -- Alteramos el cuerpo del paquete cambiando VALUES
      -- (p_val) por VALUES (p_val - 1);

      CREATE OR REPLACE PACKAGE BODY prueba IS
            PROCEDURE ejemplo(p_val IN NUMBER) IS
            BEGIN
                  INSERT INTO simple VALUES (p_val - 1);
                  COMMIT;
            END ejemplo;
      END prueba;
```

```
SQL>  select  object_name,  object_type,  status  from
user_objects
where object_name in ('PRUEBA','SIMPLE','LLAMADA')
order by object_name, object_type;
```

Los resultados que obtenemos al consultar la tabla USER_OBJECTS para los objetos de este ejemplo son los siguientes:

```
OBJECT_NAME     OBJECT_TYPE     STATUS
-----------     -----------     -------
LLAMADA         PROCEDURE       VALID
PRUEBA          PACKAGE         VALID
PRUEBA          PACKAGE BODY    VALID
SIMPLE          TABLE           VALID
```

```
SQL> DROP TABLE SIMPLE;
SQL>  select  object_name,  object_type,  status  from
user_objects
where object_name in ('PRUEBA','SIMPLE','LLAMADA')
order by object_name, object_type;
```

Los resultados que obtenemos al consultar la tabla USER_OBJECTS para los objetos de este ejemplo son los siguientes:

```
OBJECT_NAME     OBJECT_TYPE     STATUS
-----------     -----------     -------
LLAMADA         PROCEDURE       VALID
PRUEBA          PACKAGE         VALID
PRUEBA          PACKAGE BODY    INVALID
```

```
SQL> DROP PACKAGE BODY PRUEBA;
SQL>  select  object_name,  object_type,  status  from
user_objects
where object_name in ('PRUEBA','SIMPLE','LLAMADA')
order by object_name, object_type;
```

Los resultados que obtenemos al consultar la tabla USER_OBJECTS para los objetos de este ejemplo son los siguientes:

```
OBJECT_NAME          OBJECT_TYPE     STATUS
-----------     -----------     -------
PRUEBA          PACKAGE         VALID
LLAMADA         PROCEDURE       VALID
```

```
SQL> -- Alteramos la cabecera del procedimiento cambiando
     -- (p_val IN NUMBER) por (p_val IN CHAR).

     CREATE OR REPLACE PACKAGE prueba IS
          PROCEDURE ejemplo (p_val IN CHAR);
     END prueba;
```

Los resultados que obtenemos al consultar la tabla USER_OBJECTS para los objetos de este ejemplo son los siguientes:

```
OBJECT_NAME     OBJECT_TYPE     STATUS
-----------     -----------     -------
LLAMADA         PROCEDURE       INVALID
PRUEBA          PACKAGE         VALID
```

PAQUETES PREDETERMINADOS

En este capítulo se relacionan 3 de los paquetes predeterminados que más habitualmente se utilizan en la programación en PL/SQL y que les podrán resultar de utilidad para la resolución de los supuestos prácticos que se plantean en el curso.

Para poder consultar con más detalle cualquiera de los paquetes predeterminados que se incluyen en la versión 12c de Oracle, pueden acceder a la guía oficial de Oracle *PL/SQL Package and Types Reference (http://docs.oracle.com/database/121/ARPLS/toc.htm).*

Lista de paquetes predeterminados de mayor uso

A continuación, se relacionan los paquetes predetermiandos que más se utilizan y que debe conocer. En un documento anexo se relacionan el resto de paquetes predeterminados.

Nombre paquete	Descripción
`DBMS_FILE_TRANSFER`	Te permite copiar un fichero binario dentro de una base de datos o transferir un fichero binario entre bases de datos.
`DBMS_OUTPUT`	Acumula información en un buffer de modo que pueda ser recuperada posteriormente.
`UTL_FILE`	Habilita tus programas PL/SQL para leer y escribir ficheros de texto.

DBMS_FILE_TRANSFER

El paquete DBMS_FILE_TRANSFER proporciona procedimientos para copiar archivos binarios dentro de una base de datos o a transferirlos entre bases de datos diferentes.

Este paquete consta de los siguientes subprogramas:

Nombre Subprograma	Tipo	Descripción
COPY_FILE	Procedimiento	Lee un fichero del directorio origen y crea una copia en el directorio destino.
GET_FILE	Procedimiento	Contacta con una base de datos remota para leer un fichero remoto y crear una copia del mismo en el file system local o ASM.
PUT_FILE	Procedimiento	Lee un fichero local o ASM y contacta con una base de datos remota para crear una copia del mismo en el file system remoto.

Procedimiento COPY_FILE

Este procedimiento lee un fichero de un directorio origen y crea una copia del mismo en un directorio destino. Tanto el directorio origen como destino están en un FILE SYSTEM local o en un grupo de discos ASM (Automatic Storage Management).

Se puede copiar cualquier tipo de fichero a/desde un FILESYSTEM local. Sin embargo, solo se pueden copiar ficheros de base de datos (tales como DATAFILES, TEMPFILES, CONTROLFILES, etc.…), a/desde un grupo de discos ASM.

El fichero destino no se cerrará hasta que el procedimiento se termine correctamente.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
DBMS_FILE_TRANSFER.COPY_FILE(
```

```
Objeto_directorio_origen      IN VARCHAR2,
Nombre_fichero_origen         IN VARCHAR2,
Objeto_directorio_destino     IN VARCHAR2,
Nombre_fichero_destino        IN VARCHAR2);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento COPY_FILE.

objeto_directorio_origen

Es el objeto directorio que se designa como origen. El objeto directorio debe existir ya.

Para crear un objeto directorio hay que utilizar el comando `CREATE DIRECTORY`.

nombre_fichero_origen

Es el nombre del fichero que se quiere copiar. Este fichero debe existir en el directorio origen.

objeto_directorio_destino

Es el objeto directorio que se designa como destino. El objeto directorio debe existir ya. Si el destino es ASM, el objeto directorio debe designar también un nombre de grupo (por ejemplo, `+diskgroup1`) o un directorio creado a través de un alias. En el caso de un directorio, se debe especificar la ruta completa (por ejemplo: `+diskgroup1/dbs/control`).

nombre_fichero_destino

El nombre que se asigna al fichero en el directorio destino. No puede existir un fichero con el mismo nombre en el directorio destino.

RESTRICCIONES DE USO DEL PROCEDIMIENTO

Para poder ejecutar correctamente este procedimiento, el usuario que lo invoca debe de tener los siguientes privilegios:

- Privilegio de lectura (READ) en el objeto directorio especificado como origen.
- Privilegio de escritura (WRITE) en el objeto directorio especificado como destino.

Este procedimiento convierte los nombres de objeto directorio indicado en los parámetros a mayúsculas a menos que se especifiquen entre dobles comillas. Sin embargo el procedimiento no hace esto mismo con los nombres de fichero.

Los ficheros a copiar deben de cumplir los siguientes requerimientos:

- El tamaño de los ficheros a copiar debe ser un múltiplo de 512 bytes.
- El tamaño de los ficheros a copiar debe ser menor o igual a 2 terabytes.

La transferencia del fichero no es transaccional. El fichero copiado es tratado como un fichero binario, y no se realiza ninguna conversión de juegos de caracteres. Para monitorizar el progreso de fichero de gran tamaño que se está copiando, se puede consultar la vista dinámica `V$SESSION_LONGOPS`.

EJEMPLO

```
SQL> CREATE DIRECTORY DGROUP AS '+diskgroup1/dbs/backup';

Directory create.

SQL> BEGIN
       DBMS_FILE_TRANSFER.COPY_FILE('SOURCEDIR',
                                    't_xdbtmp.f',
                                    'DGROUP',
                                    't_xdbtmp.f');
       END;
       /

PL/SQL procedure successfully completed.
```

```
SQL> EXIT

$ASMCMD
ASMCMD> ls
DISKGROUP1/
ASMCMD> cd diskgroup1/dbs/backup
ASMCMD> ls
t_xdbtmp.f => +DISKGROUP1/ORCL/TEMPFILE/COPY_FILE.267.546
```

Procedimiento GET_FILE

Este procedimiento contacta con una base de datos remota para leer un fichero remoto y crear una copia del mismo en el FILE SYSTEM local o ASM. El fichero que es copiado es el fichero origen, y el nuevo fichero que resulta de la copia es el fichero destino.

El fichero destino no se cerrará hasta que el procedimiento se termine correctamente.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
DBMS_FILE_TRANSFER.GET_FILE(
      Objeto_directorio_origen      IN VARCHAR2,
      Nombre_fichero_origen         IN VARCHAR2,
      Base_datos_origen             IN VARCHAR2,
      Objeto_directorio_destino     IN VARCHAR2,
      Nombre_fichero_destino        IN VARCHAR2);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento GET_FILE.

objeto_directorio_origen

Es el objeto directorio que se designa como origen. El objeto directorio debe existir ya.

nombre_fichero_origen

Es el nombre del fichero que se quiere copiar. Este fichero debe existir en el directorio origen.

base_datos_origen

Es el nombre de una base de datos para enlazar a una base de datos remota donde se localiza el fichero.

objeto_directorio_destino

Es el objeto directorio que se designa como destino.

nombre_fichero_destino

El nombre que se asigna al fichero en el directorio destino. No puede existir un fichero con el mismo nombre en el directorio destino.

RESTRICCIONES DE USO DEL PROCEDIMIENTO

Las mismas que se indicaron en el procedimiento anterior COPY_FILE.

Procedimiento PUT_FILE

Este procedimiento lee un fichero local o ASM y contacta con una base de datos remota para crear una copia del mismo en un FILE SYSTEM remoto. El fichero que es copiado es el fichero origen, y el nuevo fichero que resulta de la copia es el fichero destino. El fichero destino no se cerrará hasta que el procedimiento se termine correctamente.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
DBMS_FILE_TRANSFER.PUT_FILE(
      Objeto_directorio_origen      IN VARCHAR2,
      Nombre_fichero_origen         IN VARCHAR2,
```

```
Objeto_directorio_destino     IN VARCHAR2,
Nombre_fichero_destino        IN VARCHAR2,
Base_datos_destino            IN VARCHAR2);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento PUT_FILE.

objeto_directorio_origen

Es el objeto directorio que se designa como origen. El objeto directorio debe existir ya.

nombre_fichero_origen

Es el nombre del fichero que se quiere copiar. Este fichero debe existir en el directorio origen.

objeto_directorio_destino

Es el objeto directorio que se designa como destino.

nombre_fichero_destino

El nombre que se asigna al fichero en el directorio destino. No puede existir un fichero con el mismo nombre en el directorio destino.

base_datos_destino

Es el nombre de una base de datos para enlazar a una base de datos remota en la que se copiará el fichero.

RESTRICCIONES DE USO DEL PROCEDIMIENTO

Las mismas que se indicaron en el procedimiento COPY_FILE.

DBMS_OUTPUT

El paquete DBMS_OUTPUT se utiliza normalmente por el usuario para realizar procesos de revisión del código (debug) o para visualizar mensajes.

Para visualizar los mensajes que se lanzan con el procedimiento PUT_LINE de este paquete dentro de SQL*PLUS, antes hay que ejecutar el comando `SET SERVEROUPUT ON` que tiene el mismo efecto que invocar a `DBMS_OUTPUT.ENABLE (buffer_size => NULL)`. En ambos casos no hay límite en la salida del buffer.

Este paquete consta de los siguientes subprogramas:

Nombre Subprograma	Tipo	Descripción
`DISABLE`	Procedimiento	Deshabilita la salida de mensajes.
`ENABLE`	Procedimiento	Habilita la salida de mensajes.
`GET_LINE`	Procedimiento	Recupera una línea del buffer.
`GET_LINES`	Procedimiento	Recupera un array de líneas del buffer.
`NEW_LINE`	Procedimiento	Termina una línea creada con `PUT`.
`PUT`	Procedimiento	Coloca una línea parcial en el buffer.
`PUT_LINE`	Procedimiento	Coloca una línea en el buffer.

Errores de la sección EXCEPTION

Los subprogramas de DBMS_OUTPUT pueden generar errores de aplicación del tipo ORA-20000, y los mensajes que se definen con los procedimientos pueden devolver los siguientes errores:

Error	Descripción
`ORU-10027`	*Buffer overflow*. Sobrepasado el tamaño del buffer de salida de mensajes.
`ORU-10028`	*Line length overflow.* Sobrepasado la longitud de salida de una línea.

Procedimiento DISABLE

Este procedimiento deshabilita las llamadas a PUT, PUT_LINE, NEW_LINE, GET_LINE y GET_LINES y limpia el buffer de cualquier información que aún permaneciese en el mismo.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
DBMS_OUTPUT.DISABLE;
```

Procedimiento ENABLE

Este procedimiento habilita las llamadas a PUT, PUT_LINE, NEW_LINE, GET_LINE y GET_LINES. Las llamadas a estos procedimientos son ignoradas si el paquete DBMS_OUTPUT no está activado.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
DBMS_OUTPUT.ENABLE(Tamaño_buffer IN INTEGER DEFAULT
                   20000);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento ENABLE.

tamaño_buffer

Es el límite en bytes, de la información que puede contener el buffer. Si se indica NULL para este parámetro indica que no debería haber límite.

RESTRICCIONES DE USO DEL PROCEDIMIENTO

No es necesario llamar a este procedimiento cuando se usa la instrucción `SET SERVEROUTPUT ON` en SQL*Plus.

Si se producen varias llamadas a este procedimiento, el tamaño del buffer queda fijado al valor de la última llamada. El tamaño máximo es de 1.000.000 y el mínimo es de 2.000.

Normalmente se espera un NULL para el parámetro del tamaño del buffer de forma que por defecto asuma 20.000, por compatibilidad con versiones anteriores de las bases de datos Oracle que no soportaban un buffer ilimitado.

Procedimiento GET_LINE

Este procedimiento recupera una única línea de información del buffer.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
DBMS_OUTPUT.GET_LINE(line       OUT VARCHAR2,
                     Status     OUT INTEGER);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento GET_LINE.

line

Devuelve una única línea de información del buffer, excluyendo el carácter final de salto de línea. La variable que recoja el valor de este parámetro se debería definir con un tamaño y tipo de VARCHAR2(32767) para evitar el riesgo de un error del tipo `ORA-06502: PL/SQL numeric or value error: character string buffer too small`.

status

Si la llamada se completa correctamente, entonces este parámetro devuelve 0. Si no hay más líneas en el buffer, entonces devuelve 1 .

Procedimiento GET_LINES

Este procedimiento recupera un array de líneas de información del buffer.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
DBMS_OUTPUT.GET_LINES (lineas      OUT CHARARR,
                       numlineas   IN OUT INTEGER);

DBMS_OUTPUT.GET_LINES (lineas OUT
DBMSOUTPUT_LINESARRAY,
                       numlineas IN OUT INTEGER);
```

El tipo CHARARR es un tipo table con la siguiente sintaxis:

```
TYPE CHARARR IS TABLE OF VARCHAR2(32767) INDEX BY
BINARY_INTEGER;
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento GET_LINES.

lineas

Devuelve un array de líneas de información del buffer. La máxima longitud de cada línea en el array es de 32.767 bytes.

numlineas

Número de líneas que se quieren recuperar del buffer. Después de recuperar el número especificado de líneas, el procedimiento devuelve el número de líneas

actualmente recuperadas. Si este número es menor que el de las líneas solicitadas, entonces no hay más líneas en el buffer.

Procedimiento NEW_LINE

Este procedimiento coloca un carácter de fin de línea.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
DBMS_OUTPUT.NEW_LINE;
```

Procedimiento PUT

Este procedimiento coloca una línea parcial en el buffer.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
DBMS_OUTPUT.PUT (cadena      IN VARCHAR2);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento PUT.

cadena

Cadena a incluir en el buffer.

Procedimiento PUT_LINE

Este procedimiento coloca una línea en el buffer.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
DBMS_OUTPUT.PUT_LINE (cadena      IN VARCHAR2);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento PUT_LINE.

cadena

Cadena a incluir en el buffer.

Reglas y límites

El tamaño máximo de una línea de salida de mensajes es de 32.767 bytes.

El tamaño por defecto del buffer es de 20.000 bytes. El tamaño mínimo es de 2.000 bytes y el máximo es ilimitado.

UTL_FILE

El paquete UTL_FILE proporciona mecanismos para leer y escribir sobre ficheros de textos del sistema operativo.

Este paquete consta de los siguientes subprogramas:

Nombre Subprograma	Tipo	Descripción
FCLOSE	Procedimiento	Cierra un fichero.
FLOSE_ALL	Procedimiento	Cierra los manejadores de todos los ficheros abiertos.
FCOPY	Procedimiento	Copia una porción contigua de un fichero a un nuevo fichero.
FFLUSH	Procedimiento	Escribe físicamente todas las salidas pendientes a un fichero.
FGETATTR	Procedimiento	Lee y devuelve los atributos de un fichero de disco.
FGETPOS	Función	Devuelve en bytes la posición relativa (offset) dentro de un fichero.
FOPEN	Función	Abre un fichero para entrada o salida.
FOPEN_NCHAR	Función	Abre un fichero en Unicode para entrada o salida.
FREMOVE	Procedimiento	Borra un fichero de disco asumiendo que se tienen los suficientes privilegios para hacerlo.
FRENAME	Procedimiento	Renombra un fichero existente a un nuevo nombre. Es similar a la función `mv` de UNIX.
FSEEK	Procedimiento	Ajusta el puntero del fichero hacia delante o hacia atrás dentro del mismo, a partir del número de bytes especificados.

GET_LINE	Procedimiento	Lee un texto desde un fichero abierto.
GET_LINE_NCHAR	Procedimiento	Lee un texto en formato Unicode desde un fichero abierto.
GET_RAW	Procedimiento	Lee un string RAW de un fichero y ajusta el puntero del fichero por el número de bytes leídos.
IS_OPEN	Función	Determina si el manejador de fichero referencia a un fichero abierto.
NEW_LINE	Procedimiento	Escribe uno o más finalizadores de línea en el fichero.
PUT	Procedimiento	Escribe una cadena de caracteres en el fichero.
PUT_LINE	Procedimiento	Escribe una línea entera (cadena de caracteres + finalizador de línea) en el fichero.
PUT_LINE_NCHAR	Procedimiento	Escribe una línea Unicode en el fichero.
PUT_NCHAR	Procedimiento	Escribe una cadena de caracteres Unicode en el fichero.
PUTF	Procedimiento	Un procedimiento PUT formateado.
PUTF_NCHAR	Procedimiento	Un procedimiento PUT_NCHAR formateado y escribe una cadena de caracteres Unicode en el fichero también formateada.
PUT_RAW	Procedimiento	Acepta una entrada de datos tipo RAW y escribe el valor en el buffer de salida.

Para realizar operaciones de lectura y escritura sobre ficheros del sistema operativo, el usuario tiene que tener estos permisos sobre el objeto directorio que apunta a la ruta de red donde se encuentran físicamente los ficheros.

Para crear este tipo de objetos directorio se utiliza la instrucción CREATE DIRECTORY.

Para manejar un fichero se tiene que definir una variable de tipo UTL_FILE.FILE_TYPE.

Errores de la sección EXCEPTION

Los subprogramas de UTL_FILE pueden generar alguno de los errores que se especifican en la siguiente tabla:

Error	Descripción
INVALID_PATH	La ruta de localización del fichero es incorrecta.
INVALID_MODE	El parámetro que indica el modo de apertura del fichero en FOPEN es incorrecto.
INVALID_FILEHANDLE	El manejador del fichero es incorrecto.
INVALID_OPERATION	El fichero podría no estar abierto o no se puede operar contra el, como se solicita.
READ_ERROR	El buffer de destino es demasiado pequeño u ocurrió un error durante la operación de lectura del fichero.
WRITE_ERROR	Un error ocurrió durante la operación de escritura en el fichero.
INTERNAL_ERROR	Error PL/SQL inespecífico.
CHARSETMISMATCH	Se abrió un fichero con la función FOPEN_NCHAR pero se están usando operaciones incompatibles como PUTF o GET_LINE.
FILE_OPEN	La operación solicitada falló porque el fichero ya está abierto.
INVALID_MAXLINESIZE	El valor MAX_LINESIZE de la función FOPEN es incorrecto. Debería estar en un rango de 1 a 32767.
INVALID_FILENAME	El parámetro correspondiente al nombre del fichero es incorrecto.

ACCESS_DENIED	Se han denegado los permisos para acceder al fichero en la ruta especificada.
INVALID_OFFSET	Las causas por las que se puede producir este error son: • ABSOLUTE_OFFSET = NULL y RELATIVE_OFFSET = NULL. • ABSOLUTE_OFFSET < 0 • El valor de offset causa la búsqueda más allá del final del fichero.
DELETE_FAILED	La operación de borrado solicitada falló.
RENAME_FAILED	La operación de renombrado solicitada falló.

El procedimiento UTL_FILE también puede devolver errores predefinidos de PL/SQL como pueden ser NO_DATA_FOUND cuando se intente leer un fichero que está vacío o VALUE_ERROR cuando se intente asignar el contenido de una línea leída del fichero a un tipo de variable incompatible.

EJEMPLO

```
SQL> CREATE DIRECTORY user_dir AS '/app1/g1/user';
SQL> GRANT READ ON DIRECTORY user_dir TO PUBLIC;
SQL> GRANT WRITE ON DIRECTORY user_dir TO PUBLIC;
```

En esta parte del ejemplo se crea un objeto directorio denominado USER_DIR que apunta a una ruta física de red Unix denomina 'app1/g1/user'. A continuación se le otorgan permisos de lectura (READ) y escritura (WRITE) a todos los usuarios existentes en la base de datos (PUBLIC).

```
DECLARE
      V1    VARCHAR2(32767);
      F1    UTL_FILE.FILE_TYPE;
BEGIN
F1 := UTL_FILE.FOPEN('USER_DIR','f_prueba.tmp','R',256);
UTL_FILE.GET_LINE(F1,V1,32767);
UTL_FILE.FCLOSE(F1);

F1 := UTL_FILE.FOPEN('USER_DIR','f_prueba.tmp','R');
UTL_FILE.GET_LINE(F1,V1,32767);
```

```
UTL_FILE.FCLOSE(F1);

F1 := UTL_FILE.FOPEN('USER_DIR','f_prueba.tmp','A');
UTL_FILE.PUT_LINE(F1,'Esta es la nueva linea insertada');
UTL_FILE.FCLOSE(F1);

END;
```

En esta parte del ejemplo, se define un bloque sin nomina (DECLARE..BEGIN..END) en el que en primer lugar se define una variable V1 para almacenar la ruta íntegra del fichero (incluido también el propio nombre del mismo), y la variable necesaria para manejar el fichero: F1 de tipo UTL_FILE.FILE_TYPE. Dentro del bloque ejecutable (BEGIN..END), en el primer grupo de sentencias se abre el fichero "f_prueba.tmp" únicamente para lectura con un tamaño máximo de buffer de lectura de 256 caracteres, por lo que aunque se define la lectura de una línea del fichero con tamaño de 32767 en el comando GET_LINE, prevalece el ancho del buffer y solo se leerán los primero 256 caracteres.

En el segundo grupo de sentencias del bloque ejecutable, se vuelve a abrir el mismo fichero para lectura pero sin indicación del ancho del buffer. Como el valor por defecto del ancho del buffer es 1024 si no se especifica ningún valor, pues pese a que otra vez se define la lectura de una línea de 32767, sólo se leerán los primeros 1024 caracteres.

Por último en el tercer grupo de sentencias del bloque ejecutable, se abre el mismo fichero de los dos casos anteriores pero para escritura añadiendo las líneas al final del mismo (APPEND) y se inserta una línea con el texto `'Esta es la nueva linea insertada'` añadiendo un retorno de carro (final de línea) tras el mismo.

Procedimiento FCLOSE

Este procedimiento cierra un fichero abierto, identificado por un manejador de fichero.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.FCLOSE(file IN OUT FILE_TYPE);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento FCLOSE.

file

Manejador de fichero activo devuelto por las llamadas a las funciones FOPEN o FOPEN_NCHAR.

Procedimiento FCLOSE_ALL

Este procedimiento cierra un todos los manejadores de fichero de la sesión en curso. Este procedimiento se debería usar como un procedimiento de limpieza de emergencia, por ejemplo, cuando un programa PL/SQL aborta por un error/excepción.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.FCLOSE_ALL;
```

Procedimiento FCOPY

Este procedimiento copia una porción contigua de un fichero a un nuevo fichero creado. Por defecto, el fichero completo es copiado si los parámetros `start_line` y `end_line` son omitidos. El fichero origen se abre en modo lectura. El fichero destino es abierto en modo escritura. Se puede especificar opcionalmente una línea de comienzo y finalización para seleccionar una porción desde el centro del fichero origen a copiar.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.FCOPY(src_location IN VARCHAR2,
```

```
src_filename      IN VARCHAR2,
dest_location     IN VARCHAR2,
dest_filename     IN VARCHAR2,
start_line        IN BINARY_INTEGER
                     DEFAULT 1,
end_line          IN BINARY_INTEGER
                     DEFAULT NULL);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento FCOPY.

src_location

Directorio de localización del fichero origen. Uno de los objetos directorio definidos en la base de datos.

src_filename

Fichero origen que será copiado.

dest_location

Directorio destino donde el fichero destino será creado.

dest_filename

El fichero destino creado a partir del fichero origen.

start_line

Número de línea de comienzo de la copia. El valor por defecto es 1 para la primera línea si se omite este parámetro.

end_line

Número de línea de finalización de la copia. El valor por defecto es NULL que equivale a indicar el final del fichero, cuando se omite este parámetro.

Procedimiento FFLUSH

Este procedimiento físicamente escribe los datos pendientes al fichero identificado por el manejador. Normalmente, los datos que son escritos al fichero son los que se encuentran en el buffer. Los datos deben de terminarse con un retorno de carro (nueva línea).

Este procedimiento es útil cuando el fichero debe ser leído mientras está todavía abierto. Por ejemplo cuando se obtienen mensajes de un proceso de debug, estos pueden ser copiados a un fichero de modo que puedan ser leídos inmediatamente.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.FFLUSH(file  IN FILE_TYPE);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento FFLUSH.

file

Manejador de fichero activo devuelto por las llamadas a las funciones FOPEN o FOPEN_NCHAR.

Procedimiento FGETATTR

Este procedimiento lee y devuelve los atributos de un fichero de disco.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.FGETATTR(location      IN VARCHAR2,
                  filename      IN VARCHAR2,
                  fexists       OUT BOOLEAN,
```

```
file_length    OUT NUMBER,
block_size     OUT BINARY_INTEGER);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento FGETATTR.

location

Directorio de localización del fichero origen. Uno de los objetos directorio definidos en la base de datos.

filename

Nombre del fichero que será examinado.

fexists

Un valor booleano que indica si existe o no el fichero.

file_length

Longitud del fichero en bytes. NULL si el fichero no existe.

block_size

Tamaño en bytes del bloque del fichero del sistema. NULL si el fichero no existe.

Función FGETPOS

Esta función devuelve la posición relativa del offset dentro del fichero, en bytes.

SINTAXIS

La sintaxis de esta función es la siguiente:

```
UTL_FILE.FGETPOS(file IN FILE_TYPE)
```

```
RETURN PLS_INTEGER;
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en la función FGETPOS.

file

Directorio de localización del fichero origen. Uno de los objetos directorio definidos en la base de datos.

RETURN PLS_INTEGER

La función devuelve un número de tipo PLS_INTEGER.

Función FOPEN

Esta función abre un fichero. Se puede especificar un máximo de tamaño en líneas y tener un máximo de 50 ficheros abiertos simultáneamente.

SINTAXIS

La sintaxis de esta función es la siguiente:

```
UTL_FILE.FOPEN(location     IN VARCHAR2,
               filename     IN VARCHAR2,
               open_mode    IN VARCHAR2,
               max_linesize IN BINARY_INTEGER
           DEFAULT 1024) RETURN FILE_TYPE;
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en la función FOPEN.

location

Directorio de localización del fichero origen. Uno de los objetos directorio definidos en la base de datos.

filename

Nombre del fichero que será examinado.

open_mode

Especifica como se abre el fichero. Los modos permitidos son:

- r: sólo lectura.
- w: solo escritura (sobreescribe todo lo existente).
- a: append (escritura al final del fichero sin borrar lo existente).
- rb: sólo lectura por bytes.
- wb: sólo escritura por bytes.
- ab: append por bytes.

Si se intenta abrir un fichero especificando el modo 'a' o 'ab' pero este fichero no existe ya, entonces el fichero será creado en modo escritura.

max_linesize

Máximo número de caracteres para cada línea, incluyendo el carácter de nueva línea, para este fichero (mínimo valor 1, máximo valor 32767). Si no se especifica, Oracle asume el valor por defecto de 1024.

RETURN FILE_TYPE

Devuelve el manejador del fichero abierto.

Función FOPEN_NCHAR

Esta función abre un fichero en el juego de caracteres nacional instalado en la base de datos, para operaciones de entrada o salida con el máximo de tamaño en número

de líneas que se haya especificado. Con esta función se puede leer o escribir un fichero de texto en formato Unicode en vez de en el juego de caracteres de la base de datos

Aunque el contenido de un buffer de tipo NVARCHAR2 puede tener información en formato AL16UTF16 o UTF8 (dependiendo del juego de caracteres nacional instalado en la base de datos), el fichero siempre será leído o escrito en formato UTF8. UTL_FILE convierte entre UTF8 y AL16UTF16 si es necesario.

SINTAXIS

La sintaxis de esta función es la siguiente:

```
UTL_FILE.FOPEN_NCHAR(location    IN VARCHAR2,
                     filename    IN VARCHAR2,
                     open_mode   IN VARCHAR2,
                     max_linesize IN BINARY_INTEGER
            DEFAULT 1024) RETURN FILE_TYPE;
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en la función FOPEN_NCHAR.

location

Directorio de localización del fichero origen. Uno de los objetos directorio definidos en la base de datos.

filename

Nombre del fichero que será examinado.

open_mode

Especifica como se abre el fichero. Los modos permitidos son:

- r: sólo lectura.
- w: solo escritura (sobreescribe todo lo que exista).
- a: append (escritura al final del fichero sin borrar lo existente).

- rb: sólo lectura por bytes.
- wb: sólo escritura por bytes.
- ab: append por bytes.

Si se intenta abrir un fichero especificando el modo 'a' o 'ab' pero este fichero no existe ya, entonces el fichero será creado en modo escritura.

max_linesize

Máximo número de caracteres para cada línea, incluyendo el carácter de nueva línea, para este fichero (mínimo valor 1, máximo valor 32767). Si no se especifica, Oracle asume el valor por defecto de 1024.

RETURN FILE_TYPE

Devuelve el manejador del fichero abierto.

Procedimiento FREMOVE

Este procedimiento borra un fichero del disco asumiendo que se tienen los permisos (del sistema operativo) necesarios para realizar esta operación.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.FREMOVE( location      IN VARCHAR2,
                  filename      IN VARCHAR2);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento FREMOVE.

location

Directorio de localización del fichero origen. Uno de los objetos directorio definidos en la base de datos.

filename

Nombre del fichero que será eliminado.

Procedimiento FRENAME

Este procedimiento renombra un fichero existente a un nuevo nombre. Esta operación es similar a la función `mv` de Unix, en la que el fichero en realidad se elimina del directorio origen y se copia con en otro directorio distinto con el mismo nombre u otro diferente.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.FRENAME(src_location    IN VARCHAR2,
                 src_filename    IN VARCHAR2,
                 dest_location   IN VARCHAR2,
                 dest_filename   IN VARCHAR2,
                 overwrite       IN BOOLEAN DEFAULT
                                    FALSE);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento FRENAME

src_location

Directorio de localización del fichero origen. Uno de los objetos directorio definidos en la base de datos.

filename

Nombre del fichero origen que será renombrado.

dest_location

Directorio destino del fichero. Uno de los objetos directorio definidos en la base de datos.

dest_filename

Nuevo nombre del fichero.

overwrite

Por defecto es FALSE. Se debe de disponer de los permisos de base de datos necesarios para realizar esta operación en ambos objetos directorio. Se usa este parámetro para especificar si se sobreescribe o no un fichero que ya exista con el mismo nombre en el directorio destino. El valor por defecto FALSE indica que no se sobreescribe.

Procedimiento FSEEK

Este procedimiento ajusta el puntero del fichero hacia delante o hacia atrás dentro del número de bytes que se especifiquen.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.FSEEK( file  IN OUT UTL_FILE.FILE_TYPE,
                absolute_offset IN PLS_INTEGER
                                DEFAULT NULL,
                relative_offset IN PLS_INTEGER
                                DEFAULT NULL);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento FSEEK.

file

Manejador del fichero.

absolute_offset

Localización absoluta del fichero que se busca. Por defecto el valor es NULL.

relative_offset

Número de bytes a buscar hacia delante o hacia atrás. Un valor positivo indica búsqueda hacia delante y un valor negativo búsqueda hacia atrás. Cero indica buscar en la posición actual. Por defecto este parámetro tiene valor NULL.

Procedimiento GET_LINE

Este procedimiento lee un texto del fichero abierto identificado por un manejador de fichero y lo lleva al parámetro del buffer de salida. El texto leído no incluye la marca de final de línea, o final de fichero o retorno de carro. La lectura no puede exceder del tamaño indicado en el parámetro `max_linesize` de la función FOPEN.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.GET_LINE (file     IN FILE_TYPE,
                   buffer  OUT VARCHAR2
                   len      IN PLS_INTEGER DEFAULT
                                 NULL);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento GET_LINE.

file

Manejador del fichero. El fichero debe estar abierto en modo R (lectura) porque si no devolverá un error/excepción del tipo INVALID_OPERATION.

buffer

Buffer de información que recibe el contenido de la línea leía del fichero.

len

El número de bytes leídos del fichero. Por defecto el valor es NULL. Si no se especifica otro valor, se asume el valor del parámetro `max_linesize`.

Procedimiento GET_LINE_NCHAR

Este procedimiento lee un texto del fichero abierto identificado por un manejador de fichero y lo lleva al parámetro del buffer de salida. Con esta función se puede leer o escribir un fichero de texto en formato Unicode en vez de en el juego de caracteres de la base de datos.

El fichero debe estar abierto en el juego de caracteres nacional y debe estar codificado en UTF8. El tipo de dato que se espera para el buffer será NVARCHAR2. Si la variable es de otro tipo, tal como NCHAR, NCLOB o VARCHAR2, PL/SQL realizará un conversión implícita desde NVARCHAR2 después de haber leído el texto.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.GET_LINE_NCHAR(file     IN FILE_TYPE,
                        buffer  OUT VARCHAR2
                        len  IN PLS_INTEGER
                                DEFAULT NULL);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento GET_LINE_NCHAR.

file

Manejador del fichero. El fichero debe estar abierto en modo R (lectura) porque si no devolverá un error/excepción del tipo INVALID_OPERATION.

buffer

Buffer de información que recibe el contenido de la línea leía del fichero.

len

El número de bytes leídos del fichero. Por defecto el valor es NULL. Si no se especifica otro valor, se asume el valor del parámetro `max_linesize`.

Procedimiento GET_RAW

Este procedimiento lee una cadena de tipo RAW desde un fichero y ajusta la posición del puntero al número de bytes leído. UTL_FILE.GET_RAW ignora los finalizadores de línea (retorno de carro, fin de línea o fin de fichero).

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.GET_RAW(file     IN FILE_TYPE,
                 buffer  OUT NOCOPY RAW
                 len      IN PLS_INTEGER DEFAULT
                             NULL);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento GET_RAW

file

Manejador del fichero.

buffer

Información leída en formato RAW.

len

El número de bytes leídos del fichero. Por defecto el valor es NULL. Si no se especifica otro valor, se asume la máxima longitud del tipo RAW.

Función IS_OPEN

Esta función comprueba el manejador del fichero para ver si identifica a un fichero abierto. IS_OPEN informa si el manejador apunta a un fichero que ha sido abierto y todavía no se ha cerrado.

SINTAXIS

La sintaxis de esta función es la siguiente:

```
UTL_FILE.IS_OPEN(file IN FILE_TYPE)
                 RETURN BOOLEAN;
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en la función IS_OPEN.

file

Manejador del fichero.

RETURN BOOLEAN

Devuelve TRUE si el fichero está abierto o FALSE en caso contrario.

Procedimiento NEW_LINE

Este procedimiento escribe una o más finalizadores de línea (carácter fin de línea) en el fichero especificado por el manejador.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.NEW_LINE(file     IN FILE_TYPE,
                  lines    IN BINARY_INTEGER := 1);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento NEW_LINE

file

Manejador del fichero.

lines

Número de finalizadores de línea que serán escritos en el fichero.

Procedimiento PUT

Este procedimiento escribe una cadena de caracteres en el buffer que se llevará al fichero abierto. El fichero debe estar abierto en modo escritura. Con este procedimiento no se añaden símbolos de fin de línea (para ello hay que usar el procedimiento NEW_LINE o PUT_LINE).

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.PUT( file    IN FILE_TYPE,
              buffer  IN VARCHAR2);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento PUT.

file

Manejador del fichero.

buffer

Buffer que contiene el texto a escribir en el fichero.

Procedimiento PUT_LINE

Este procedimiento escribe una cadena de caracteres en el buffer que se llevará al fichero abierto. El fichero debe estar abierto en modo escritura. Con este procedimiento si que se añade un símbolo de fin de línea después de la cadena de caracteres a escribir.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.PUT_LINE(file      IN FILE_TYPE,
                  buffer    IN VARCHAR2,
                  autoflush IN BOOLEAN DEFAULT
                               FALSE);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento PUT_LINE.

file

Manejador del fichero.

buffer

Buffer que contiene el texto a escribir en el fichero.

autoflush

Limpia el buffer del disco después de la escritura.

Procedimiento PUT_LINE_NCHAR

Este procedimiento escribe una cadena de caracteres en el buffer que se llevará al fichero abierto. Con esta función se puede escribir un fichero de texto en formato Unicode en vez de en el juego de caracteres de la base de datos. El fichero debe estar abierto en modo escritura. Con este procedimiento si que se añade un símbolo de fin de línea después de la cadena de caracteres a escribir.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.PUT_LINE_NCHAR(file      IN FILE_TYPE,
                        buffer    IN VARCHAR2);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento PUT_LINE_NCHAR.

file

Manejador del fichero.

buffer

Buffer que contiene el texto a escribir en el fichero.

Procedimiento PUT_NCHAR

Este procedimiento escribe una cadena de caracteres en el buffer que se llevará al fichero abierto. Con esta función se puede escribir un fichero de texto en formato Unicode en vez de en el juego de caracteres de la base de datos. El fichero debe estar abierto en modo escritura.

El texto a escribir debe estar en el juego de caracteres UTF8. El buffer de salida espera un tipo de dato NVARCHAR2. Si la variable es de otro tipo , se realizará una conversión implícita a NVARCHAR2 antes de escribir el texto.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.PUT_NCHAR(file     IN FILE_TYPE,
                   buffer   IN VARCHAR2);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento PUT_NCHAR.

file

Manejador del fichero.

buffer

Buffer que contiene el texto a escribir en el fichero. El usuario deber abrir el fichero en modo "w" o en modo "a". En caso contrario devolverá el error/excepción INVALID_OPERATION.

Procedimiento PUTF

Este procedimiento es un formateo de su homólogo PUT. Trabaja como un `printf()` limitado.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.PUTF(file      IN FILE_TYPE,
              format    IN VARCHAR2,
              [arg1     IN VARCHAR2 DEFAULT NULL,
                …
               arg5     IN VARCHAR2 DEFAULT NULL]);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento PUTF.

file

Manejador del fichero.

format

Cadena de caracteres formateada que puede contener un texto en el que se hayan formateado los caracteres `\n` y `%s`.

arg1..arg5

Desde uno hasta 5 cadenas de argumentos para operaciones de formateo.

`%s` indica que se sustituya la secuencia con la cadena del próximo argumento en la lista de argumentos.

`\n` indica que se sustituya con el carácter de fin de línea apropiado a la plataforma en la que se esté trabajando.

EJEMPLO

```
Hola mundo!

Yo vengo de Zork con saludos para todos los terrícolas.
```

```
   DECLARE
      My_world   VARCHAR2(4) := 'Zork';
      Manejador  UTL_FILE.FILE_TYPE;
BEGIN
      ….
      PUTF (manejador,'Hola mundo!\nYo vengo de %s con %s.
\n',My_world,'saludos para todos los terrícolas');
      …
   END;
```

En este ejemplo en primer lugar se definen 2 líneas de un fichero cualquiera y a continuación parte de un bloque sin nominar en la que en el código ejecutable se introduce una sentencia PUTF.

Procedimiento PUTF_NCHAR

Este procedimiento es un formateo de su homólogo PUT_NCHAR.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.PUTF_NCHAR(
      file       IN FILE_TYPE,
      format     IN VARCHAR2,
      [arg1      IN VARCHAR2 DEFAULT NULL,
                  …
       arg5      IN VARCHAR2 DEFAULT NULL]);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento PUTF_NCHAR.

file

Manejador del fichero.

format

Cadena de caracteres formateada que puede contener un texto en el que se hayan formateado los caracteres `\n` y `%s`.

arg1..arg5

Desde uno hasta 5 cadenas de argumentos para operaciones de formateo.

`%s` indica que se sustituya la secuencia con la cadena del próximo argumento en la lista de argumentos.

`\n` indica que se sustituya con el carácter de fin de línea apropiado a la plataforma en la que se esté trabajando.

Procedimiento PUT_RAW

Este procedimiento acepta como entrada un valor de tipo RAW y lo escribe en el buffer de salida.

SINTAXIS

La sintaxis de este procedimiento es la siguiente:

```
UTL_FILE.PUT_RAW(file      IN UTL_FILE.FILE_TYPE,
                 buffer    IN RAW,
                 autoflush IN BOOLEAN DEFAULT
                              FALSE);
```

DESCRIPCIÓN DE LOS PARÁMETROS

A continuación se describen cada uno de los parámetros que se usan en el procedimiento PUT_RAW.

file

Manejador del fichero.

buffer

Buffer que contiene la información RAW a escribir.

autoflush

Si es TRUE entonces realiza una operación de limpieza después de escribir el valor en el buffer de salida. El valor por defecto es FALSE.