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¶

ELEMENTOS DEL LENGUAJE

PL/SQL es un sofisticado lenguaje de programación que se utiliza para acceder a bases de datos Oracle desde distintos entornos. PL/SQL está integrado con el servidor de base de datos, de modo que el código puede ser procesado de forma rápida y eficiente. También se encuentra disponible en varias de las herramientas de cliente que posee Oracle, entre ellas SQL*PLUS, Developer Suite, JDeveloper, etc.

Si nos preguntamos por qué utilizar PL/SQL, la conclusión la encontramos en el propio SQL. Tenemos que recordar que Oracle es una base de datos relacional, que utiliza como lenguaje de datos el propio SQL. SQL es un lenguaje flexible y eficiente, con características muy potentes para la manipulación y examen de los datos relacionales, pero que presenta deficiencias a la hora de realizar programaciones procedimentales.

SQL es un lenguaje de cuarta generación (4GL), que como el resto de lenguajes de esta generación, presenta como característica el hecho de que describen lo que debe hacerse, pero no la manera de llevarlo a cabo. Por ejemplo si analizamos la siguiente instrucción:

```
DELETE FROM estudiantes WHERE nombre like 'Pep%'
```

Esta instrucción determina que queremos borrar de la tabla estudiantes todos aquellos cuyo nombre comience por "Pep", pero no dice como va a realizar el gestor de base de datos el proceso para conseguir eliminar dichos registros. Parece presumible que recorrerá los datos de dicha tabla en un cierto orden para determinar qué elementos debe borrar y luego los eliminará, no obstante es algo que no nos interesa para la instrucción.

En contraposición a los lenguajes 4GL nos encontramos los lenguajes de tercera generación (3GL), como C y Visual Basic. Son lenguajes más procedimentales donde se implementan algoritmos para resolver unos problemas. Estas estructuras procedimentales y de ejecución paso a paso no se pueden implementar en SQL, así que Oracle necesitaba de un lenguaje que pudiese resolver este tipo de problemas y que estuviera más enfocado no solo al manejo de datos sino a la resolución de

problemáticas de todo tipo, así que creo el lenguaje PL/SQL (Lenguaje Procedimental / SQL). Este lenguaje no es solo un lenguaje de tipo 3GL, sino que permite utilizar la flexibilidad de SQL como lenguaje de 4GL.

Esta característica que le define, es posible dado que es un lenguaje particular del sistema gestor de bases de datos Oracle y no un lenguaje estándar.

Por tanto el lenguaje PL/SQL potencia el lenguaje SQL agregando estructuras y objetos del siguiente tipo:

- El bloque
- Manejo de errores y excepciones
- Creación de procedimientos y funciones
- Definición de variables y tipos
- Estructuras de bucle
- Cursores
- Objetos

El bloque

Es la unidad básica de todo programa en PL/SQL. Todo programa al menos debe poseer un bloque.

Todo bloque consta de una sección declarativa optativa, una sección de ejecución obligatoria y una sección de control de errores optativa.

La sintaxis es la siguiente:

```
[DECLARE]
        <sección declarativa>
BEGIN
        <sección de ejecución>
[EXCEPTION]
```

```
        <sección de control de errores>
END;
```

SECCIÓN DECLARATIVA

En esta sección se definen las variables, constantes y cursores que se van a utilizar dentro de la sección de ejecución.

SECCIÓN DE EJECUCIÓN

La sección de ejecución presenta las siguientes particularidades:

- Toda instrucción `SELECT` (no dinámica) que aparezca en esta sección, deberá llevar incorporado el parámetro `INTO`, como se muestra en el siguiente ejemplo:

  `SELECT` columnas `INTO` variables `FROM` tablas `WHERE` criterios.

- En esta sección sólo se admiten instrucciones de SQL del tipo DML (select, insert, update y delete) o instrucciones SQL dinámicas, el resto no están permitidas implementarlas directamente (por ejemplo: alter, create, drop, etc.), salvo que se indique dentro de la instrucción `EXECUTE INMEDIATE`.

SECCIÓN DE CONTROL DE ERRORES

En esta sección se definen los controles programados para detectar los errores de ejecución del bloque y la solución adoptada para los mismos.

Tipos de bloque

Se diferencian 3 tipos de bloques:

- Los *bloques anónimos* se construyen de manera dinámica y se ejecutan una sola vez.

- Los *bloques nominados* se construyen identificándolos con un nombre. Al igual que los anteriores, se construyen de forma dinámica y se ejecutan una sola vez.
- Los *subprogramas* son bloques nominados que se almacenan en la base de datos. Nos podemos encontrar con procedimientos, paquetes y funciones de este tipo. Siempre se ejecutan bajo demanda.
- Los *disparadores* son también bloques nominados que se almacenan en la base de datos, pero que no se pueden ejecutar bajo petición de un programa. Se ejecutan cuando tiene efecto el suceso para el que se han programado contra una cierta tabla del sistema.

EJEMPLO DE BLOQUE ANÓNIMO

```
DECLARE
   Var1  NUMBER;
BEGIN
   Var1 := 1;
END;
```

EJEMPLO DE BLOQUE NOMINADO

```
<<Nombre_Bloque>>
DECLARE
   Var1  NUMBER;
BEGIN
   Var1 := 1;
END;
```

EJEMPLO DE BLOQUE SUBPROGRAMA

```
CREATE OR REPLACE PROCEDURE MI_PROGRAMA IS
   Var1  NUMBER;
BEGIN
   Var1 := 1;
END;
```

EJEMPLO DE DISPARADOR

```
CREATE OR REPLACE TRIGGER MI_DISPARADOR
```

```
    BEFORE INSERT OR UPDATE OF numero ON tabla_temporal
    FOR EACH ROW
BEGIN
    IF :new.numero < 0 THEN
         RAISE_APPLICATION_ERROR(-20100,'¡¡¡¡Error!!!');
    END IF;
END;
```

Manejo de errores y excepciones

El lenguaje PL/SQL como muchos otros procedimentales, permite un control sobre los errores que se produzcan en la ejecución del código. Esta gestión de errores presenta como ventaja la claridad para su manipulación, dado que utiliza una sección independiente a la del código ejecutable.

Creación de procedimientos y funciones

El lenguaje PL/SQL permite la creación de procedimientos almacenados y de funciones que nos devuelvan un valor como resultado de su ejecución.

Definición de variables y tipos

El lenguaje PL/SQL también permite la definición de variables para utilizar en nuestros programas y crear tipos de usuarios a partir de otros predefinidos.

Estructuras de bucle

Para desarrollar nuestros programas y poder realizar operaciones de bifurcación, el lenguaje PL/SQL posee estructuras de control.

Cursores

Este tipo de estructura que se define en la sección declarativa de un bloque, nos permite recuperar en memoria un conjunto de filas de una tabla que se recorren una a una para un tratamiento posterior.

Objetos

Oracle dada la tendencia actual de los lenguajes de programación que hay en el mercado, incorpora también la figura del objeto, de forma que PL/SQL se puede convertir también en un lenguaje orientado a objetos.

Salida por pantalla de los resultados de una ejecución

PL/SQL a través de las herramientas propias de Oracle: SQL*Plus y SQL*Plus Worksheet únicamente visualiza el resultado satisfactorio o no de la ejecución de las instrucciones. Para poder realizar una visualización en pantalla de la ejecución y resultados internos del código PL/SQL hay que utilizar la invocación de un paquete incluido en Oracle PL/SQL denominado DBMS_OUTPUT, el cual permite dirigir a pantalla resultados mediante el uso de la función PUT_LINE.

No obstante para que se haga completamente efectiva dicha salida a pantalla, antes hay que activar el comando de ejecución en pantalla del paquete DBMS_OUTPUT, siempre que se inicie una nueva sesión con la herramienta correspondiente. Este comando es el siguiente:

```
SET SERVEROUTPUT ON;
```

EJEMPLO DE UN BLOQUE CON SALIDA A PANTALLA DE LOS RESULTADOS

```
SET SERVEROUTPUT ON;
DECLARE
   Var1  VARCHAR2(50);
BEGIN
   Var1 := 'Antolin';
   DBMS_OUTPUT.PUT_LINE('El nombre del autor es:'||
                        Var1);
END;
/
```

UNIDADES LÉXICAS

Es el conjunto de caracteres y gramática a utilizar en la programación de PL/SQL. Contamos con los siguientes elementos:

- Identificadores
- Palabras reservadas
- Delimitadores
- Literales
- Comentarios

Identificadores

Dan nombre a los distintos objetos de nuestro programa. Por ejemplo los identificadores de variable, cursores, tipos, etc...

Un identificador tiene que comenzar obligatoriamente con una letra seguida por una secuencia de caracteres, entre los que se pueden incluir:

- Letras
- Números
- El símbolo $
- El carácter de subrayado _
- El símbolo #

La longitud máxima de un identificador es de 30 caracteres hasta la versión 11g y a partir de la versión 12c aumenta hasta 128 caracteres.

También hay que tener en cuenta que el lenguaje PL/SQL no es un lenguaje sensible en cuanto a los caracteres, de manera que no distingue mayúsculas o minúsculas, salvo que el nombre vaya encerrado entre comillas dobles (").

Cada objeto tiene un espacio de nombres.

Los espacios de nombres de objeto aglutinan una serie de nombres de objeto, en algunos casos de forma independiente, y en otros común para distintos nombres de objetos diferentes.

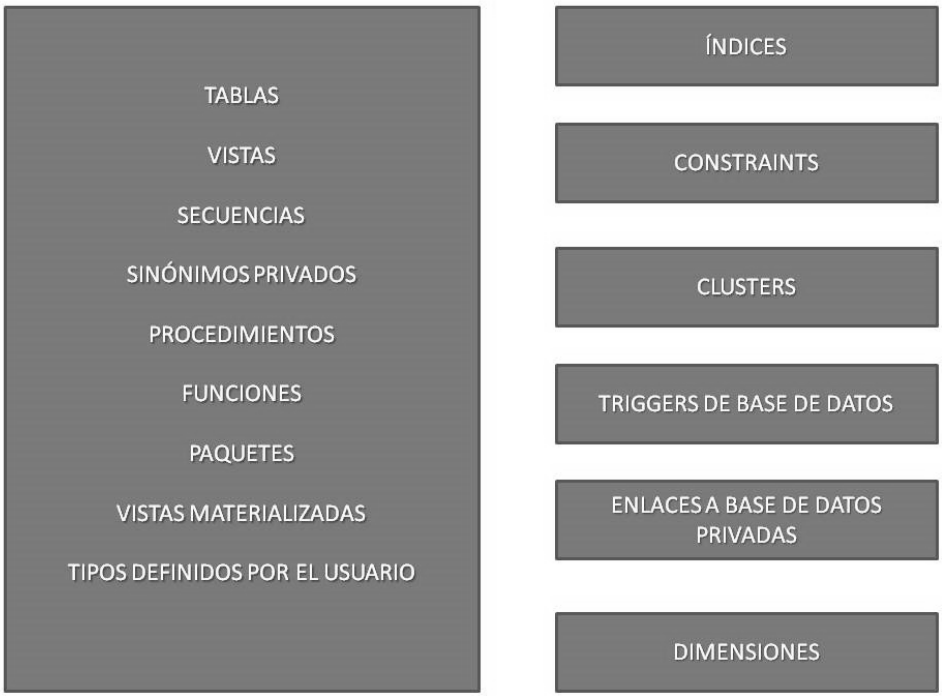

Fig. 1-1 Namespaces (Esquemas de nombre) de objetos de un esquema.

Dentro del mismo espacio de nombres no se puede repetir un nombre de objeto aunque sea de distinto tipo.

Por ejemplo una tabla y una vista no se pueden denominar con el mismo nombre, porque comparten el mismo esquema de nombres. En cambio una restricción y un trigger si se pueden llamar igual, porque cada uno de estos objetos del esquema de base de datos poseen esquemas independientes.

Fig. 1-2 Namespaces de objetos que no pertenecen a un esquema.

Ejemplos de nombres de esquema válidos son los siguientes:

```
EMP
"Emp"
SCOTT.FECHAALTA
"INCLUSO ESTO & VALE!"
UN_NOMBRE_LARGO_Y_VALIDO
```

Palabras reservadas

Son todas aquellas que define Oracle como restringidas dentro del lenguaje PL/SQL y cuyo nombre no podrá llevar ningún otro objeto de la base de datos.

Delimitadores

Son símbolos con un significado especial dentro de PL/SQL, tal y como se especifica en la siguiente tabla:

Símbolo	Descripción	Símbolo	Descripción
+	Operador de suma	–	Operador de resta
*	Operador de multiplicación	/	Operador de división
=	Operador de igualdad	<	Operador menor que
>	Operador mayor que	(	Delimitador inicial de una expresión
)	Delimitador final de una expresión	;	Fin de una orden
%	Indicador de atributo	,	Separador de elementos
.	Selector de componente	@	Indicador de enlace a base de datos
'	Delimitador de cadena de caracteres	"	Delimitador de cadena entrecomillada
:	Indicador de variable de asignación	**	Operador de exponenciación
<>	Operador distinto de	!=	Operador distinto de
~=	Operador distinto de	^=	Operador distinto de
<=	Operador menor o igual que	>=	Operador mayor o igual que
:=	Operador de asignación	=>	Operador de asociación
..	Operador de rango	||	Operador de concatenación
<<	Delimitador de comienzo de etiqueta	>>	Delimitador fin de etiqueta
--	Indicador de comentario en una línea	/*	Comienzo de comentario multilineal
*/	Fin de comentario multilineal	<space>	Espacio
<tab>	Carácter de tabulación	<cr>	Retorno de carro

Literales

Los literales pueden ser de los siguientes tipos:

- Booleanos: TRUE, FALSE, NULL.
- Numéricos: 123, -7, +12, 0
- Carácter: '123', 'hola'
- De fecha: '1998-12-25' (formato solo fecha)

'1997-10-22 13:01:01' (formato fecha y hora)
'1997-01-31 09:26:56.66 +02:00' (formato fecha y hora)

Comentarios

Si se quieren incluir comentarios en una sola línea, se tiene que escribir la siguiente sintaxis:

```
-- texto
```

Si el texto a incluir ocupa más de una línea, se tiene que escribir la siguiente sintaxis:

```
/* texto
   Prueba en varias líneas */
```

TIPOS DE DATOS

Los tipos de datos que se pueden utilizar en PL/SQL se dividen en las siguientes categorías:

- Tipos escalares
- Tipos compuestos
- Tipos puntero
- Tipos lob

Tipos escalares

Los tipos escalares se dividen en las siguientes categorías:

- Numéricos
- De caracteres
- Booleanos
- De fecha y hora
- Raw
- Rowid
- Urowid

A continuación se muestra una lista con cada uno de los tipos que se pueden utilizar en cada categoría y una breve descripción de los mismos.

TIPOS NUMÉRICOS

Tipo	Descripción
BINARY_DOUBLE	Tipo de dato numérico que almacena números en coma flotante de 64 bits.
BINARY_FLOAT	Tipo de dato numérico que almacena números en coma flotante de 32 bits.
DEC	Tipo de dato ANSI equivalente al tipo NUMBER.
DECIMAL	Tipo de dato ANSI equivalente al tipo NUMBER.
DOUBLE PRECISION	Tipo de dato ANSI con una precisión de 126 en binario.
FLOAT	Es un subtipo del tipo NUMBER con una precisión en el rango de 1 a 126 dígitos binarios.
INT	Tipo de dato ANSI que equivale al tipo NUMBER(38).
INTEGER	Tipo de dato ANSI que equivale al tipo NUMBER(38).
NUMBER	Tipo de dato numérico que admite una precisión de 1 a 38 y una escala de -84 a 127.
NUMERIC	Tipo de dato ANSI equivalente al tipo NUMBER.
PLS_INTEGER	Tipo de dato numérico que almacena enteros con signo en un rango de -2.147.483.648 y 2.147.483.647.
REAL	Tipo de dato ANSI con una precisión de 63 en binario o 18 en decimal.
SMALLINT	Tipo de dato ANSI que equivale al tipo NUMBER(38).

TIPOS CARÁCTER

Tipo	Descripción
CHAR	Admite un string (conjunto de caracteres) de longitud fija. El tamaño máximo admitido es de 2000 bytes o caracteres y el mínimo es de 1 byte. En PL/SQL admite strings con un rango de 1 a 32.676 bytes.
CHARACTER	Tipo de dato ANSI equivalente al tipo CHAR.
LONG	Admite un string de longitud variable de hasta 2 gigabytes en SQL. En PL/SQL solo admite strings con un rango de 1 a 32.670 bytes.
LONG VARCHAR	Tipo de dato DB2 equivalente al tipo LONG.
NATIONAL CHARACTER	Tipo de dato ANSI equivalente al tipo NCHAR.
NCHAR	Igual que el tipo CHAR pero codificado en el juego de caracteres nacional que se haya definido.
NATIONAL CHAR	Tipo de dato ANSI equivalente al tipo NCHAR.
NVARCHAR2	Igual que el tipo VARCHAR2 pero codificado en el juego de caracteres nacional que se haya definido.
VARCHAR	Tipo de dato ANSI equivalente al tipo VARCHAR2.
VARCHAR2	Admite un string (conjunto de caracteres) de longitud variable con un tamaño máximo de 4000 bytes o caracteres y un tamaño mínimo de 1 byte en SQL. En PL/SQL admite strings con un rango de 1 a 32.676 bytes.

TIPOS BOOLEANOS

Tipo	Descripción
BOOLEAN	Tipo de dato condicional que sólo admite los valores TRUE, FALSE o NULL.

TIPOS DE FECHA Y HORA

Tipo	Descripción
DATE	Tipo de dato de fecha y hora que almacena los valores con un tamaño máximo de 7 bytes.
TIMESTAMP	Tipo de datos de fecha y hora con una precisión de 0 a 9 dígitos.
TIMESTAMP WITH TIME ZONE	Igual que tipo TIMESTAMP pero almacenando también los valores de fecha y hora correspondientes a la zona horaria de la base de datos.
TIMESTAMP WITH LOCAL TIME ZONE	Igual que tipo TIMESTAMP pero almacenando también los valores de fecha y hora correspondientes a la zona horaria del servidor.
INTERVAL YEAR TO MONTH	Tipo de dato que almacenará un periodo de tiempo en años y meses.
INTERVAL DAY TO SECOND	Tipo de dato que almacenará un periodo en horas, minutos y segundos.

TIPOS RAW

Tipo	Descripción
RAW	Tipo de datos binario con un tamaño máximo de 2000 bytes.
LONG RAW	Tipo de datos binario con un tamaño máximo de 2 gigabytes.

TIPOS ROWID

Tipo	Descripción
ROWID	Tipo de dato que almacena la dirección lógica de ubicación del registro en la tabla. Se representa en formato carácter con base 64.

TIPOS UROWID

Tipo	Descripción
UROWID	Equivalente al tipo ROWID pero para un tipo de tabla INDEX-ORGANIZED.

Tipos compuestos

Los tipos compuestos se componen a partir de uno o varios de los tipos escalares anteriores, y se distinguen los siguientes tipos:

- RECORD
- TABLE
- VARRAY

En el capítulo 4 se explican de una manera pormenorizada, cada uno de estos tipos.

Tipos punteros

Los tipos punteros se utilizan para referenciar a cursores en memoria o tipos objeto, y se distinguen los siguientes tipos:

- REF CURSOR
- REF tipo objeto

Tipos LOB

Permiten referenciar a ficheros de gran tamaño almacenados externamente a la base de datos, pero referenciados desde la misma, y se distinguen los siguientes tipos:

- BFILE: enlaza ficheros binarios de hasta un máximo de 4 GB.
- BLOB: enlaza ficheros binario de hasta (4 GB – 1) * (bloque b.d.)
- CLOB: enlaza ficheros de tipo carácter de hasta (4 GB -1) * (bloque b.d.)
- NLOB: es un subtipo del tipo CLOB, para almacenar ficheros en el juego de caracteres nacional definido.

VARIABLES

La declaración de variables se realiza dentro de la sección DECLARE de un bloque y su sintaxis es la siguiente:

```
<nombre_variable> tipo [CONSTANT tipo | DEFAULT]
[NOT NULL] [:= valor]
```

A continuación se muestra una serie de ejemplos con los distintos tipos de definición de variables:

```
DECLARE
 Var1          NUMBER(5);
 Var2          NUMBER := 10;
 Var3          NUMBER(5) NOT NULL := 0;
 Var4          CONSTANT VARCHAR2(10) := 'Hola';
 Var5          NUMBER DEFAULT 45;
 Intervalo1    INTERVAL YEAR(3) TO MONTH;
 Intervalo2    INTERVAL DAY(3) TO SECOND(3);
 Var6          TIMESTAMP;
BEGIN
      NULL;
END;
```

Por defecto una variable que no se ha inicializado en la sección DECLARE a un valor concreto, tendrá valor NULL al comenzar la sección ejecutable (BEGIN).

Asignación de valores a variables

La sintaxis para la asignación de un valor a las variables es la siguiente:

```
<variable>  := <valor>;
```

Tipos utilizados con variables.

A parte de los tipos estándar definidos en ORACLE, y que se han definido con anterioridad, podemos utilizar 2 más:

- %TYPE.
- %ROWTYPE.

%TYPE.

Cuando se utiliza al declarar una variable se asume el tamaño y tipo de la columna de la tabla asociada que se referencia.

A continuación se muestra un ejemplo de uso de este tipo:

```
DECLARE
      Var_emple_nombre      empleados.nombre%TYPE;
```

La variable *var_emple_nombre* asume el tamaño y tipo de la columna *nombre* de la tabla *empleados*.

%ROWTYPE.

Cuando se utiliza al declarar una variable se asume el conjunto de columnas los tipos de la tabla que se referencia

A continuación se muestra un ejemplo de uso de este tipo:

```
DECLARE
      Var_emple_nombre      empleados%ROWTYPE;
```

La variable *var_emple_nombre* se convierte en un tipo compuesto (RECORD) que asume el conjunto de columnas y tipos de la tabla *empleados*.

Subtipos.

Los subtipos son definiciones de tipos basados en otro tipo predefinido.

A continuación se muestra un ejemplo de uso de este tipo:

```
DECLARE
     SUBTYPE    contador IS NUMBER(4);
     Var_conta  contador;
```

Hemos definido en el ejemplo un subtipo *contador* que es de tipo numérico con tamaño de 4.

Petición de valores por pantalla.

En PL/SQL puede resultar de utilidad la petición de valores por pantalla, siempre y cuando se utilicen en bloques anónimos que se ejecutan en el momento. No es conveniente en bloques nominados que se almacenan en la base de datos, dado que la ejecución de los mismos quedaría parada a expensas de introducir un valor.

La sintaxis para la petición de un valor por pantalla asociada a una variable numérica es la siguiente:

```
     <variable>  := &texto_a_mostrar;
```

La sintaxis para la petición de un valor por pantalla asociada a una variable de texto es la siguiente:

```
     <variable>  := '&texto_a_mostrar';
```

EJEMPLO

```
DECLARE
  Var1     NUMBER(3) := &Indique_la_edad;
  Var2     VARCHAR2(100) := '&Indique_nombre_y_apellidos';
BEGIN
  DBMS_OUTPUT.PUT_LINE('Usted se llama: '||VAR2);
  DBMS_OUTPUT.PUT_LINE('Y su edad es: '||
                       TRIM(TO_CHAR(VAR1)));
END;
```

ENTORNOS DE EJECUCIÓN PL/SQL

Los entornos de ejecución son los programas que nos van a permitir lanzar código PL/SQL contra el sistema gestor de base de datos Oracle.

En algunos entornos como Oracle Forms y Procedure Builder los bloques PL/SQL pueden ejecutarse completamente en el cliente sin interacciones con el servidor de la base de datos.

En otros entornos como SQL*Plus, los precompiladores o SQL-Station, se pueden enviar desde un programa cliente para ejecutarse en el servidor.

Sql*Plus / iSql*plus / Sql*Worksheet

SQL*Plus permite al usuario introducir órdenes SQL y bloques PL/SQL interactivamente. Si se producen errores hay que reescribir el código para volverlo a ejecutar.

iSQL*Plus permite al usuario introducir órdenes SQL y bloque PL/SQL a través de un navegador (Explorer, Firefox, etc.) y permite la corrección de los errores sin necesidad de reescribir todo el código.

SQL*Worksheet permite al usuario introducir órdenes SQL y bloques PL/SQL a través de una consulta en java divida en 2 áreas: área de comandos (introducción de sentencias) y área de resultados. Permite la corrección de los errores sin necesidad de reescribir todo el código. Presenta como problema la dificultad para ejecutar bloques PL/SQL donde se soliciten valores por pantalla.

En todos los casos las órdenes se envían directamente a la base de datos y los resultados se visualizan en la pantalla.

Ejecución de código SQL

Cuando queremos que se ejecute un código habrá de introducirse la sentencia y terminar con el símbolo punto y coma (`;`). Esto hace que automáticamente se ejecute la misma dentro de SQL*Plus.

Dentro de iSQL*Plus y SQL*Worksheet habrá de pulsarse sobre el botón ejecutar.

Por ejemplo:

```
SQL> SELECT SYSDATE FROM DUAL;
```

Ejecución de código PL/SQL

Cuando queremos que se ejecute un código PL/SQL en SQL*Plus habrá de introducirse el mismo y al terminar indicar el símbolo `/`. Esto permite al intérprete de comandos que evalúe la expresión, la compile para ver si es correcta y la ejecute.

Dentro de iSQL*Plus y SQL*Worksheet habrá de pulsarse sobre el botón ejecutar para que lo ejecute y no es necesario que detrás del "`END;`" se introduzca el símbolo `/`.

Por ejemplo:

```
SQL>  BEGIN
2  >  NULL;
3  >  END;
4  >  /
```

Definición de variables globales

La sintaxis para la definición de una variable global es la siguiente:

```
VARIABLE nombre tipo;
```

Una variable global es accesible por cualquier código SQL o PL/SQL que se lance en la misma sesión donde se ha declarado. Una vez que se cierra dicha sesión la variable global desaparece.

Por ejemplo:

```
SQL>  VARIABLE v_num NUMBER;
```

Uso de variables globales

¿Cómo podemos visualizar el valor de una variable global? La sintaxis es:

```
PRINT nombre_variable_global;
```

¿Cómo se puede utilizar una variable global dentro de un bloque si nominar o nominado? La sintaxis es:

```
:nombre
```

Por ejemplo:

```
SQL>  VARIABLE v_pruebas NUMBER;
SQL>  SET SERVEROUTPUT ON;

SQL>  BEGIN
2  >  :v_pruebas := 20;
3  >  dbms_output.put_line(:v_pruebas);
4  >  end;
5  >  /

SQL> PRINT v_pruebas;
```

Como se puede llamar a un procedimiento almacenado

La sintaxis para invocarlo directamente en la línea de comandos es:

```
EXEC[UTE] nombre_procedimiento [ (parámetro1,…,parámetroX) ];
```

Por ejemplo para invocar un procedimiento que no tiene parámetros:

```
SQL>  EXEC pruebas;
```

Por ejemplo para invocar un procedimiento que solo tiene parámetros de entrada:

```
SQL>  EXECUTE pruebas(1);
```

Por ejemplo para invocar un procedimiento que tiene parámetros de entrada/salida:

```
SQL>  VARIABLE :v_cuenta NUMBER;

SQL>  BEGIN
2   >  :v_cuenta := 1;
3   >  END;
4   >  /
SQL>  EXEC pruebas(:v_cuenta);

SQL>  PRINT :v_cuenta;
```

Como se puede llamar a una función almacenada

La sintaxis para invocarlo dentro de una instrucción SELECT en la línea de comandos es la siguiente:

```
SELECT nombre_funcion [ (parámetros) ] FROM DUAL;
```

La sintaxis para invocarlo dentro de un bloque sin nominar en la línea de comandos es la siguiente:

```
BEGIN
  :nombre_variable_global := nombre_funcion [ (parámetros) ];
END;
```

Por ejemplo para invocarlo desde una consulta:

```
SQL>  SELECT FACTORIAL(3) FROM DUAL;
```

Por ejemplo para invocarlo desde un bloque sin nominar:

```
SQL>  VARIABLE v_cuenta NUMBER;
SQL>  BEGIN
2  >  :v_cuenta := FACTORIAL(3);
3  >  END;
4  >  /

SQL>   PRINT v_cuenta;
```

Envío de resultados a un archivo

En muchas ocasiones el resultado de la ejecución de los comandos SQL que lanzamos en el intérprete de comandos correspondiente lo queremos almacenar en un fichero para su revisión. Esta opción es posible utilizando la siguiente sintaxis:

```
SPOOL ruta_y_nombre_archivo;
```

Después del comando SPOOL debemos indicar un nombre de archivo donde se enviará el resultado de todo lo que se ejecute desde el momento de lanzar el comando, teniendo en cuenta que el archivo debe contener una ruta completa accesible desde el intérprete de comandos.

Cuando se quiera terminar el envío de los resultados de ejecutar los comandos SQL y PL/SQL al archivo, habrá de indicarse la siguiente sintaxis que cierra el archivo:

```
SPOOL OFF;
```

Por ejemplo:

```
SQL>  SPOOL C:\TMP\resultado.txt;
SQL>  SELECT SYSDATE FROM DUAL;
SQL>  SPOOL OFF;
```

Después de ejecutar el ejemplo anterior, si abriésemos el fichero RESULTADO.TXT que se ha almacenado en la ruta C:\TMP, el resultado que obtendríamos sería el siguiente:

```
SQL> SELECT SYSDATE FROM DUAL;
```

SYSDATE
20/09/11

SQL> SPOOL OFF;

Ejecución de Scripts (archivos de comandos)

Todos los intérpretes de comandos SQL y PL/SQL admite la ejecución de un script (archivo) de comandos que se haya construido en formato de texto plano.

La sintaxis para la ejecución del script es:

```
START ruta_y_nombre_archivo;
```

También se puede utilizar la siguiente sintaxis en sustitución de la anterior:

```
@ruta_y_nombre_archivo;
```

Por ejemplo:

```
SQL>  START C:\TMP\archivocomandos.sql;
SQL>  @C:\TMP\archivocomandos.sql;
```

Mostrar errores de compilación

En este libro hemos aprendido a visualizar los errores que se producen en la compilación de bloques tanto nominados (procedimientos, funciones, paquetes y

triggers), como sin nominar, utilizando la vista del sistema USER_ERRORS, pero también existe otra opción equivalente, que resulta más rápida para capturarlos:

```
SHOW ERRORS;
```

Por ejemplo:

```
SQL> CREATE PROCEDURE PRUEBAS2 IS
1  > BEGIN
2  > NULL
3  > END;
4  >  /

Warning: Procedure created with compilation errors.

SQL> SHOW ERRORS;

LINE/COL ERROR
-------- -----------------------------------------------------
----------------
3/5        PLS-00103: Encountered the symbol "NULL" when
expecting one of
         the following:
         constant exception <an identifier>
         <a   double-quoted   delimited-identifier>   table
LONG_ double
```

En este ejemplo concreto hemos forzado que se produzca un error de compilación al haber omitido el símbolo ; después de la instrucción NULL.

HERRAMIENTAS DE DISEÑO

Con SQL*Plus y en general cualquier intérprete de comandos SQL y PL/SQL podemos diseñar estructuras de programa que nos permitan realizar las acciones que necesitamos, pero resulta evidente que no podemos diseñar un programa entero que posea pantallas, botones, campos de texto, listas, etc...

Para ello necesitamos herramientas de diseño que nos permitan trabajar con código SQL, PL/SQL o Java contra Oracle, y que dispongan de un interfaz visual moldeable por el usuario a través de formularios y pantallas.

Estas herramientas han ido renovándose en el portal de Oracle con el paso de los años. Puede encontrar la relación de herramientas disponibles para el desarrollo y diseño contra bases de datos Oracle en este enlace:

https://www.oracle.com/application-development/technologies/

En este apartado vamos a repasar algunas de las que históricamente han tenido más éxito entre los profesionales de las Tecnologías de la Información:

- Oracle JDeveloper
- Oracle Forms
- Oracle Reports
- Oracle BI Publisher
- Oracle Developer Tools for Visual Studio

Oracle JDeveloper

Es una herramienta orientado a la programación en arquitecturas SOA y Java. Permite el diseño completo de aplicaciones para dichos entornos programando en Java.

Puede encontrar más información y descarga del producto en este enlace:

https://www.oracle.com/application-development/technologies/jdeveloper.html

Oracle Forms

Es un una herramienta que permite la creación de aplicaciones en entornos gráficos programando directamente en SQL y PL/SQL. La filosofía de trabajo de la herramienta es el uso de formularios cuya funcionalidad se obtiene a través de eventos que ocurren sobre los elementos del mismo.

Posee asistentes para la creación de los elementos más importantes de un proyecto: bloques, lista de valores, elementos de texto, pantallas, etc...

Posee un potente depurador para poder controlar y visualizar paso a paso y mediante interrupciones el código que se ejecuta en el proyecto.

Permite la creación de aplicaciones para su publicación en Web.

Es un programa abierto dado que interactúa con otras aplicaciones y herramientas gracias a que tiene habilitado recursos de Java Beans, controles Active X, OLE (Object linking and embedding) y DDE (Dynamic Data Exchange).

Permite además el desarrollo de aplicaciones para acceso a otras bases de datos como RDB, SQL SERVER, INFORMIX, SYBASE, DB2 y ACCESS.

Puede encontrar más información y descarga del producto en el siguiente enlace:

https://www.oracle.com/application-development/technologies/forms/forms.html

Oracle Reports

Es un una herramienta que permite la creación de informes o listados. No se recomienda su uso, dado que Oracle dejará de mantenerla a partir de 2023

Como en caso anterior utiliza asistentes para la creación de los elementos más importantes del informe. Y también permite la generación de listados / informes para la Web.

Es capaz de generar listados/informes en los siguientes formatos: HTML, PDF, XML, RTF y texto.

Puede encontrar más información y descarga del producto en el siguiente enlace:

https://www.oracle.com/middleware/technologies/reports.html

Oracle BI Publisher

Es la solución de informes para crear, administrar y entregar informes y documentos de manera más fácil y rápida que con las herramientas de informes tradicionales (Oracle Reports). Es la herramienta que recomienda Oracle para sustituir a Oracle Reports.

Posee una versión de escritorio y de servidor.

Puede encontrar más información y descarga del producto en el siguiente enlace:

https://www.oracle.com/es/middleware/technologies/bi-publisher.html

Oracle Developer Tools for Visual Studio

Es una herramienta para añadir a Microsoft Visual Studio y que se integra perfectamente con Visual Studio hasta las versiones 2017 y 2019.

Las características que aporta esta herramienta a Microsoft Visual Studio son las siguientes:

- Generación automática de código .NET.
- Facilidad para el desarrollo de aplicaciones ASP.NET para la Web.
- Herramientas para la mejora del rendimiento de las aplicaciones (Tuning).
- Generación de scripts SQL para creación de objetos del esquema de Oracle con integración en el panel de control del código fuente.
- Editor PL/SQL y debugger para corrección de errores.
- Desarrollo de procedimientos almacenados NET.
- Sistema integrado de ayuda online.
- Manejo de usuarios, roles y privilegios.
- Diseñadores avanzados de consultas.
- Definición y creación de tipos de usuarios.

- Asistentes para la importación de tablas.
- Edición de datos, testeo de procedimientos almacenados y ejecución de código SQL.

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¶