Oracle 11g
PL/SQL

Curso práctico de formación

Antolín Muñoz Chaparro

Oracle 11g PL/SQL. Curso práctico de formación
Antolín Muñoz Chaparro

ISBN: 978-84-939450-1-5

EAN: 9788493945015

RC Libros
Calle Mar Mediterráneo, 2. Nave 6
28830 SAN FERNANDO DE HENARES, Madrid
Teléfono: +34 91 677 57 22
Fax: +34 91 677 57 22
Correo electrónico: info@rclibros.es
Internet: www.rclibros.es

Diseño de colección, cubierta y pre-impresión: Grupo RC
Impresión y encuadernación: Service Point
Depósito Legal: M-5313-2012
Impreso en España

16 15 14 13 12 (1 2 3 4 5 6 7 8 9 10 11 12)

PRÓLOGO

Hace más de dos décadas tuve el honor de enseñarle por primera vez a Antolín Muñoz la base de datos Oracle. En ese momento no era consciente de la relación tan fructífera que ambos iban a mantener a lo largo de estos años.

Estas dos décadas prodigiosas para los que hemos tenido la suerte de trabajar en el mundo de la informática, o como se dice ahora: Tecnologías de la Información y de las Comunicaciones, han representado una evolución vertiginosa de todos los aspectos relacionados con esta ciencia. Destacamos en el mundo del hardware y sobre todo en los mini y los mainframe; cómo han ido desapareciendo los fabricantes en unos casos, y en otros cómo han evolucionado hacia la fabricación de dispositivos generalmente periféricos y ordenadores personales. Otro aspecto a destacar es la mejora constante de las comunicaciones en la topología de estrella; desde el ordenador central, a las redes en bus Ethernet y distribuidas mediante Router y Switch, la apertura hacia el mundo exterior; desde las líneas punto a punto hasta las redes que facilitaron el acceso al correo e Internet y la mejora de las velocidades y los anchos de banda. Por último, resulta casi imposible hablar de la microinformática porque pasar de los terminales VT orientados al carácter a los actuales ordenadores personales multimedia, se convertiría en una interminable relación fuera de contexto. Igual nos pasaría con los múltiples desarrollos que han popularizado los PC (Personal Computers), en el campo profesional, de ocio y entretenimiento.

Mientras tanto: ¿Qué hacían Antolín y el gigante Oracle? Oracle lo conocimos en su versión 6 de base de datos, que no solo disponía de un SGBD (Sistema Gestor de Base de Datos), sino que como es lógico, estaba acompañado de algunas herramientas y de un lenguaje procedimental conocido como PL/SQL.

En 1992 apareció la versión 7 de Oracle donde lo más significativo respecto a la versión anterior era el almacenamiento y ejecución de programas escritos en PL/SQL dentro del SGBD, así como el soporte de la integridad referencial.

Internet empezaba a dar los primeros indicios de su existencia y a convertirse en la excelente realidad actual y en 1999 sale a la luz la versión 8i de Oracle. La "i" es un claro indicativo de que cumple los requerimientos de Internet, permitiendo el almacenamiento y ejecución de contenidos multimedia, y el SGBD incorpora la ejecución y almacenamiento de código Java, al incorporar la máquina virtual de dicho lenguaje. No hace falta decir que se habían acabado los desarrollos orientados a carácter y comenzaban los desarrollos orientados a objeto.

A partir de esta versión pasamos a utilizar una herramienta que nos permitió trabajar en cliente/servidor: Oracle Developer, que es un entorno gráfico para el diseño de aplicaciones, y que nos facilita mucho la creación de formularios, su compilación y ejecución. Es una herramienta bastante intuitiva, aunque presenta como principal desventaja el que el código fuente compilado hay que tenerlo en una carpeta compartida a los demás usuarios, y esto puede provocar la pérdida de las distintas versiones si no se es muy cuidadoso. Pero no debemos olvidarnos de que hablamos de una herramienta de diseño de formularios, y el desarrollo siempre tiene que venir acompañado de un lenguaje, que desde el comienzo siempre ha sido PL/SQL, aunque actualmente existe una herramienta paralela: Oracle JDeveloper que admite el desarrollo en lenguaje Java.

Todas estas versiones de la B.D. Oracle implicaron una actualización de conocimientos para llevar a la práctica con gran éxito: migraciones de las distintas versiones, migraciones de hardware, migraciones de aplicaciones orientadas a carácter a aplicaciones orientadas a objeto, el "efecto 2000", la llegada del euro, etc. Todas ellas realizadas por un grupo reducido de gente en las que Antolín fue en todas y cada una de ellas el verdadero gestor ejecutivo, que junto con una cuidada planificación permitió que tan complicados objetivos se realizaran en tiempo récord.

A la vez, Antolín encontraba tiempo para hacer incursiones en el campo de la docencia relacionada con los distintos entornos de estas tecnologías, y adquirir unos conocimientos pedagógicos que le han permitido escribir estos manuales formativos.

El lector tiene ante sí un libro en el que se aúnan 20 años de experiencia con el lenguaje PL/SQL con las últimas adaptaciones a la versión 11g de Oracle, la experiencia pedagógica en la formación de productos de esta empresa, la experiencia en importantes desarrollos de aplicaciones, y sobre todo la seguridad de que esta obra está planificada, desarrollada y ejecutada con toda la precisión que Antolín pone en todos sus trabajos.

Eduardo Solans

1 FUNDAMENTOS DEL LENGUAJE PL/SQL

INTRODUCCIÓN

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 10g, 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. Este 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 cómo 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 con 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 creó el lenguaje PL/SQL (Lenguaje Procedimiental / 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 solo 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 ello.

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

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.

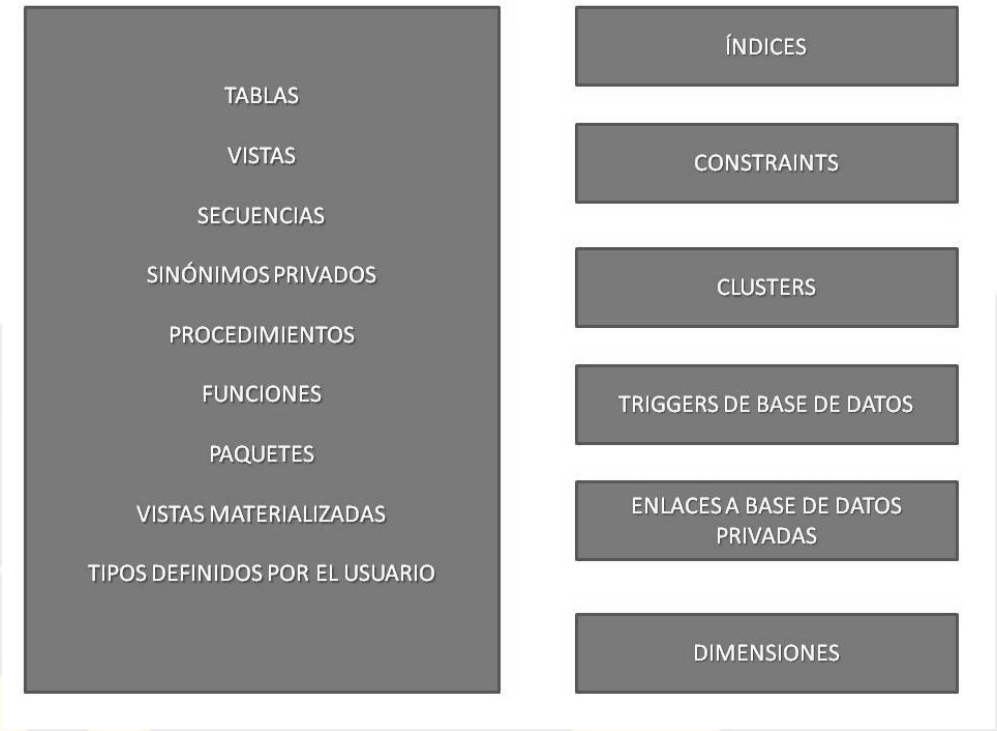

Fig. 1-1 Namespaces (Esquemas de nombre) de objetos de un esquema.

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 sí se pueden llamar igual, porque cada uno de estos objetos del esquema de base de datos posee esquemas independientes.

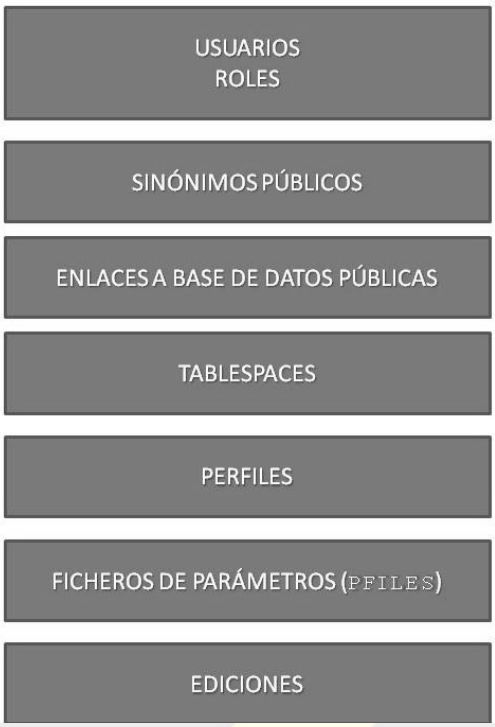

Fig. 1-2 Namespaces de objetos que no pertenecen a un esquema.

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 estas 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 ellos.

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 2.000 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 solo 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 período de tiempo en años y meses.
INTERVAL DAY TO SECOND	Tipo de dato que almacenará un período en días horas, minutos y segundos.

TIPOS RAW

Tipo	Descripción
RAW	Tipo de datos binario con un tamaño máximo de 2.000 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 explicarán, 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 estos 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.

DECLARACIÓN DE 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

Aparte de los tipos estándar definidos en ORACLE, y que se han definido con anterioridad, podemos utilizar dos más:

- %TYPE.
- %ROWTYPE.

%TYPE.

Permite declarar una variable que asume el tamaño y tipo de la columna de la tabla referenciada.

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.

Permite declarar una variable que asume el conjunto de columnas (en el mismo orden) y los tipos de la tabla referenciada.

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;
```

SUPUESTO PRÁCTICO 0: ***Resolución en el Anexo I de este manual.***

Si usted no ha realizado previamente el "Curso de SQL para Oracle 10g" de esta misma editorial, tendrá que cargar la estructura de base de datos de un HOSPITAL que se utilizará como ejemplo durante este curso. Para ello, deberá realizar las siguientes operaciones:

- Ejecutar en SQL*PLUS el script *script_bdhospital.sql* incluido en el Anexo III de este libro. También puede encontrarlo en la web del autor o de la editorial.

- Una vez ejecutado este script se habrá cargado toda la configuración del esquema HOSPITAL (tablas, secuencias, datos…) en el usuario PEPERF con contraseña PEPITO.

.

- Conéctese con el usuario PEPERF y consulte la información de los objetos creados para este usuario dentro de la tabla USER_OBJECTS, cuyo nombre no empiece por 'BIN'.

SUPUESTO PRÁCTICO 1: ***Resolución en el Anexo I de este manual.***

Crear un bloque sin nominar en SQL*PLUS que realice las siguientes operaciones:

- Solicitar el siguiente literal por pantalla: *nombre_de_enfermo,* y almacenarlo en una variable del mismo tipo y tamaño que el campo que almacena el nombre en la tabla enfermo.
- Solicitar el siguiente literal por pantalla: *apellidos_del_mismo,* y almacenarlo en una variable del mismo tipo y tamaño que el campo que almacena el apellido en la tabla enfermo.
- Solicitar el siguiente literal por pantalla: *dirección_donde_reside,* y almacenarlo en una variable del mismo tipo y tamaño que el campo que almacena la dirección en la tabla enfermo.
- Una vez introducidos todos estos datos, se deberá ejecutar el siguiente código:

```
DBMS_OUTPUT.PUT_LINE('DATOS     DEL     ENFERMO');
DBMS_OUTPUT.PUT_LINE ('-----------------'||
                      CHR(10));
```

- Mostrar en una primera línea el nombre del enfermo introducido por pantalla seguido de una coma (,) y el apellido.
- Mostrar en una segunda línea la dirección del enfermo introducida por pantalla.

```
/* Activación en SQL*PLUS de la variable de
entorno que posibilita ver resultados por
pantalla mediante el paquete DBMS_OUTPUT de
PL/SQL */

SET SERVEROUTPUT ON
```