TIPOS DE DATOS:

PL/SQL dispone de los mismos tipos de datos que SQL, además de otros propios.

Se pueden clasificar en las siguientes categorías:

- Escalares: almacenan valores simples. Pueden ser carácter, numérico, booleano, fecha/hora, …
- Compuestos: tipos que se forman a partir de los simples. Pueden ser tablas, varrays, objetos,… (los veremos en el próximo tema)
- Referencias: Punteros,…

TIPOS ESCALARES:

CARÁCTER

Almacena cadenas de caracteres cuya longitud máxima es 32 KBytes. Al especificar las longitudes debemos tener en cuenta que, por defecto, cada carácter ocupa 1 byte, aunque hay juegos de caracteres que ocupan más (por ejemplo, el asiático).

CHAR[(L)]

NCHAR[(L)]

Almacena cadenas de caracteres de **longitud fija**. Opcionalmente se puede especificar, entre paréntesis, la longitud máxima; en caso contrario, el valor por defecto es 1 y no se pone el paréntesis. Las posiciones no utilizadas **se rellenan con blancos**.
Ejemplo:

```
Nombre CHAR(15);
Letra_nif CHAR;
Situacion CHAR();--ERROR
```

La longitud de **CHAR** se expresa por **defecto en bytes** pero se puede expresar en caracteres del juego de caracteres nacional utilizando la opción CHAR:

```
Apellido CHAR(25 CHAR);
```

La longitud en **NCHAR** se expresa siempre en caracteres del juego de **caracteres** nacional. Equivale a usar la opción CHAR comentada arriba:

```
Apellido NCHAR(25); -- equivalente a la anterior
```

VARCHAR2(L)

NVARCHAR2(L)

VARCHAR(L)

Almacena cadenas de caracteres de **longitud variable** cuyo límite máximo se debe especificar.
Ejemplos:

```
Apellidos VARCHAR2(25);
Control VARCHAR2; --ERR falta la longitud
Nombre VARCHAR2(15 CHAR);
Nombre NVARCHAR2(25);
```

También está disponible el tipo VARCHAR por compatibilidad con el estándar SQL, aunque Oracle desaconseja su utilización recomendando utilizar en su lugar VARCHAR2 y NVARCHAR2 para beneficiarse de la evolución de estos tipos.

LONG[(L)]

Almacena cadenas de longitud variable. Es similar a VARCHAR2 pero no permite la opción CHAR.
Ejemplos:

```
v_Observaciones LONG(5000);
Resumen LONG;
```

Debemos tener en cuenta que el límite para este tipo en variables PL/SQL es de 32 KB, pero las columnas de este tipo admiten hasta 2GB.

NUMÉRICOS

NUMBER(P, E)

Donde *P* es la precisión (número total de dígitos) y *E* es la escala (número de decimales).
La precisión y la escala son opcionales, pero si se especifica la escala hay que indicar la precisión.
Ejemplo:

```
Importe NUMBER(5,2); -- admite hasta 999.99
Centimos NUMBER(,2); -- ERROR lo correcto es
Centimos NUMBER(2,2); -- admite hasta .99
```

Si al asignar un valor se excede la escala, se producirá *truncamiento*. Si se excede la precisión, se producirá un *error*.
Ejemplo:

```
Importe:= 1000; -- ERROR excede la escala
Importe:= 234.344; -- guarda 234.34
```

PL/SQL dispone de subtipos de NUMBER que se utilizan por compatibilidad y/o para establecer restricciones. Son DECIMAL, NUMERIC, INTEGER, REAL, SMALLINT, etcétera.

BINARY_INTEGER

Es un tipo numérico entero que se almacena en memoria en formato binario para facilitar los cálculos. Puede contener valores comprendidos entre –2147483647 y + 2147483647.
Se utiliza en contadores, índices, etcétera.
Por ejemplo:

```
Contador BINARY_INTEGER;
```

Subtipos: NATURAL y POSITIVE.

PLS_INTEGER

Similar a BINARY_INTEGER y con el mismo rango de valores, pero tiene dos ventajas respecto al anterior:

- Es más rápido.
- Si se da desbordamiento en el cálculo se produce un error y se levanta la excepción correspondiente, lo cual no ocurre con BINARY_INTEGER.

Por ejemplo:

```
indice PLS_INTEGER;
```

BINARY_DOUBLE
BINARY_FLOAT

Disponibles desde la versión 10gR1, su utilización se circunscribe al ámbito científico para realizar cálculos muy precisos y complejos conforme a la norma IEEE 754. Oracle no los soporta directamente como columnas en la base de datos. No los utilizaremos en este libro. Remitimos al lector a la documentación de Oracle para su eventual utilización.

BOOLEANO

BOOLEAN

Almacena valores lógicos TRUE, FALSE y NULL. Para representar estos valores **no debemos utilizar comillas.**
Ejemplo:

```
v_ocupado BOOLEAN;
v_eliminado BOOLEAN DEFAUL FALSE;
```

La base de datos no soporta este tipo para definir columnas.

FECHA / HORA

Almacenan valores de tiempo. Son idénticos a los tipos correspondientes de la base de datos.

DATE

Almacena fechas incluyendo la hora. Los formatos en que muestran la información son los establecidos por defecto, aunque se pueden especificar máscaras de formato. Por defecto, sólo se muestra la fecha, pero también se almacena la hora: `día/mes/año horas:minutos:segundos`.

TIMESTAMP [(P)]

TIMESTAMP también almacena fecha y hora pero incluyendo fracciones de segundo. La precisión de estas fracciones se puede especificar como parámetro (por defecto es 6, es decir, 999999 millonésimas de segundo, pero puede variar entre 0 y 9). Normalmente el valor se toma de la variable del sistema SYSTIMESTAMP.

TIMESTAMP [(P)] **WITH TIME ZONE**

TIMESTAMP WITH TIME ZONE incluye el valor de la **zona horaria.**

TIMESTAMP [(P)] **WITH LOCAL TIME ZONE**

TIMESTAMP WITH LOCAL TIME ZONE igual que TIMESTAMP, pero:

- Los datos **se almacenan normalizados a la zona horaria donde se encuentra la base de datos.**
- Los datos recuperados **se muestran en el formato de la zona horaria correspondiente a la sesión** de los usuarios.

El siguiente ejemplo ilustra estos conceptos:

```
DECLARE
   fechaTI TIMESTAMP;
   fechaTZ TIMESTAMP WITH TIME ZONE;
   fechaTL TIMESTAMP WITH LOCAL TIME ZONE;
BEGIN /* incluiremos dos instrucciones en la misma linea */
   fechaTI := SYSTIMESTAMP;   DBMS_OUTPUT.PUT_LINE('fechaTI:'||fechaTI);
   fechaTZ := SYSTIMESTAMP;   DBMS_OUTPUT.PUT_LINE('fechaTZ:'||fechaTZ);
   fechaTL := SYSTIMESTAMP;   DBMS_OUTPUT.PUT_LINE('fechaTL:'||fechaTL);
END;
```

El resultado de la ejecución será:

```
fechaTI:02/04/06 10:07:40,196000
fechaTZ:02/04/06 10:07:40,196000 +02:00
fechaTL:02/04/06 10:07:40,196000
```

En este caso, FechaTL tiene el mismo valor que FechaTI porque la zona horaria de almacenamiento y recuperación es la misma.

INTERVAL YEAR [(PY)] **TO MONTH**

INTERVAL DAY [(PD)] **TO SECOND** [(PS)]

Los subtipos de INTERVAL representan intervalos de tiempo entre dos fechas. La diferencia entre ellos es la manera de expresar el intervalo:

INTERVAL YEAR TO MONTH se expresa en años/meses. Opcionalmente se puede incluir la precisión para el número de años, entre 0 y 9 dígitos, por defecto es 2.

```
v_anios_meses2 INTERVAL YEAR TO MONTH := '1-06';
```

- INTERVAL DAY TO SECOND se expresa en días/segundos. Opcionalmente se puede incluir la precisión para el número de días (entre 0 y 9 dígitos, por defecto es 2), y para el número de segundos (entre 0 y 9 dígitos, por defecto es 6).

```
v_dias_segundos INTERVAL DAY TO SECOND DEFAULT '02 14:0:0.0';
```

Estos tipos se utilizan cuando se requiere manejar diferencias de tiempo con precisiones expresadas en fracciones de segundo. En la mayoría de los casos recurriremos a los tipos tradicionales junto con las funciones disponibles MONTHS_BETWEEN, etcétera.

OTROS TIPOS ESCALARES

RAW(L)

Almacena datos binarios en longitud fija. Se utiliza para almacenar cadenas de caracteres evitando problemas por las conversiones entre conjuntos de caracteres que realiza Oracle.

LONG RAW

Almacena datos binarios en longitud variable evitando conversiones entre conjuntos de caracteres.

ROWID

Almacenan identificadores de direcciones de fila. Son idénticos a los tipos correspondientes de la base de datos.

UROWID[(L)]

UROWID permite especificar la longitud de almacenamiento en número de bytes (4000 bytes es el máximo y el valor por defecto).