FUNCIONES

B. Funciones de grupos de valores

Hasta ahora nos hemos ocupado de funciones que operan con valores simples; no obstante, hay otras funciones estadísticas, como SUM, AVG y COUNT, que actúan sobre un grupo de filas para obtener un valor. Estas funciones permiten obtener la edad media de un grupo de alumnos, el alumno más joven, el más viejo, el número total de miembros de un grupo, etcétera. Los valores nulos son ignorados por las funciones de grupos de valores y los cálculos se realizan sin contar con ellos. Estas funciones se muestran en la Tabla 4.2.

Función	Propósito
AVG(n)	Calcula el valor medio de 'n' ignorando los valores nulos.
COUNT (* | expresión)	Cuenta el número de veces que la expresión evalúa algún dato con valor no nulo. La opción '*' cuenta todas las filas seleccionadas.
MAX(expresión)	Calcula el máximo valor de la 'expresión'.
MIN(expresión)	Calcula el mínimo valor de la 'expresión'.
SUM(expresión)	Obtiene la suma de valores de la 'expresión' distintos de nulos.
VARIANCE (expresión)	Obtiene la varianza de los valores de 'expresión' distintos de nulos.

Tabla 4.2. *Funciones de grupos de valores.*

Caso práctico

2 AVG(n). Cálculo del salario medio de los empleados del departamento 10 de la tabla EMPLE:

```
SQL> SELECT AVG(SALARIO) FROM EMPLE WHERE DEPT_NO=10;

     AVG(SALARIO)
     ------------
       2891,66667
```

COUNT (* | expresión). Cálculo del número de filas de la tabla EMPLE:

```
SQL> SELECT COUNT(*) FROM EMPLE;

          COUNT(*)
          --------
                14
```

Cálculo del número de filas de la tabla EMPLE donde la COMISION no es nula:

```
SQL> SELECT COUNT(COMISION) FROM EMPLE;

     COUNT(COMISION)
     ---------------
                   4
```

MAX(expresión). Cálculo del máximo salario de la tabla EMPLE:

```
SQL> SELECT MAX(SALARIO) FROM EMPLE;
```

(Continúa)

(Continuación)

```
MAX(SALARIO)
------------
        4100
```

Obtén el apellido máximo (alfabéticamente) de la tabla EMPLE:

```
SQL> SELECT MAX(APELLIDO) FROM EMPLE;

MAX(APELLI
----------
TOVAR
```

Obtén el apellido del empleado que tiene mayor salario:

```
SQL> SELECT APELLIDO, SALARIO FROM EMPLE WHERE SALARIO=(SELECT MAX(SALARIO) FROM EMPLE);

APELLIDO           SALARIO
----------      ----------
REY                   4100
```

Se necesita una subconsulta para calcular el máximo salario y compararlo después con el salario de cada uno de los empleados de la tabla EMPLE.

MIN(expresión). Obtén el mínimo salario de la tabla EMPLE:

```
SQL> SELECT MIN(SALARIO) FROM EMPLE;

MIN(SALARIO)
------------
        1040
```

Obtén el apellido del empleado que tiene mínimo salario:

```
SQL> SELECT APELLIDO, SALARIO FROM EMPLE WHERE SALARIO=(SELECT MIN(SALARIO) FROM EMPLE);

APELLIDO           SALARIO
----------      ----------
SANCHEZ               1040
```

En primer lugar, se selecciona el salario mínimo y, a continuación, se compara con los salarios de la tabla EMPLE para obtener el apellido que tiene ese salario mínimo.

SUM(expresión). Consigue la suma de todos los salarios de la tabla EMPLE:

```
SQL> SELECT SUM(SALARIO) FROM EMPLE;

SUM(SALARIO)
------------
       30460
```

VARIANCE(expresión). Obtén la varianza de todos los salarios de la tabla EMPLE:

```
SQL> SELECT VARIANCE(SALARIO) FROM EMPLE;

VARIANCE(SALARIO)
-----------------
       872226,374
```

DISTINCT en funciones de grupo

En todas las funciones de grupo, al indicar los argumentos se pueden emplear las **cláusulas DISTINCT** y **ALL**, aunque no se suelen utilizar en las funciones AVG, SUM, MAX ni MIN, pero sí es más normal su uso en COUNT.

Recordemos que DISTINCT realiza una selección de filas cuyos valores en la columna especificada no están duplicados. La cláusula ALL recoge todas las filas aunque sus valores estén duplicados.

El formato de COUNT incluyendo DISTINCT y ALL es éste:

```
COUNT ( * | [DISTINCT | ALL]  expresión)
```

Si COUNT recibe como argumento una expresión o columna, ésta podrá ir precedida de las cláusulas ALL o DISTINCT.

Caso práctico

3 **Calcula el número de oficios que hay en la tabla EMPLE:**

```
SQL> SELECT  COUNT(OFICIO) "OFICIOS"  FROM EMPLE;

        OFICIOS
        --------
              14
```

Esta consulta cuenta todos los oficios de la tabla EMPLE que no sean nulos, estén repetidos o no. Si queremos contar los distintos oficios que hay en la tabla EMPLE, tendríamos que incluir DISTINCT en la función de grupo: `SQL> SELECT COUNT(DISTINCT OFICIO) "OFICIOS"  FROM EMPLE;`

```
        OFICIOS
        --------
               5
```

DISTINCT obliga a COUNT a contar sólo el número de oficios distintos.

Actividades propuestas

2 **A partir de la tabla EMPLE, visualiza cuántos apellidos empiezan por la letra 'A'.**

Obtén el apellido o apellidos de empleados que empiecen por la letra 'A' y que tengan máximo salario (de los que empiezan por la letra 'A').

C. Funciones de listas

Las **funciones de listas** trabajan sobre un grupo de columnas dentro de una misma fila. Comparan los valores de cada una de las columnas en el interior de una fila para obtener el mayor o el menor valor de la lista. Las funciones de listas se muestran en la Tabla 4.3.

Función	Propósito
GREATEST (valor1, valor2, ...)	Obtiene el mayor valor de la lista.
LEAST (valor1, valor2, ...)	Obtiene el menor valor de la lista.

Tabla 4.3. *Funciones de listas.*

Caso práctico

4 Sea la tabla **NOTAS_ALUMNOS:**

```
SQL> DESC NOTAS_ALUMNOS;

       Nombre                ¿Nulo?          Tipo
       ----------------      ----------      ------------
       NOMBRE_ALUMNO         NOT NULL        VARCHAR2(25)
       NOTA1                                 NUMBER(2)
       NOTA2                                 NUMBER(2)
       NOTA3                                 NUMBER(2)
```

Obtén por cada alumno la mayor nota y la menor nota de las tres que tiene:

```
SQL>SELECT NOMBRE_ALUMNO,GREATEST(NOTA1,NOTA2,NOTA3)"MAYOR",LEAST(NOTA1,NOTA2,
NOTA3)"MENOR"  FROM NOTAS_ALUMNOS;

       NOMBRE_ALUMNO                    MAYOR          MENOR
       -------------------------      ----------     ----------
       Alcalde García, M. Luisa                5              5
       Benito Martín, Luis                     8              6
       Casas Martínez, Manuel                  7              5
       Corregidor Sánchez, Ana                 9              6
       Díaz Sánchez, Maria
```

La última fila tiene un resultado de NULL, debido a que GREATEST y LEAST no pueden comparar un valor con otro valor nulo. Estas funciones se pueden usar también con columnas de caracteres. Ejemplos: Obtén el mayor nombre alfabético de la lista:

```
SQL> SELECT GREATEST('BENITO','JORGE','ANDRES','ISABEL') FROM DUAL;

          GREAT
          -----
          JORGE
```

4.3 Funciones de cadenas de caracteres

Las **funciones de cadenas de caracteres** trabajan con datos de tipo CHAR o VARCHAR2. Estos datos incluyen cualquier carácter alfanumérico: letras, números y caracteres especiales. Los literales se deben encerrar entre comillas simples. Ejemplo de una cadena de caracteres: 'El Quijote'.

Las funciones de cadenas permiten manipular cadenas de letras u otros caracteres. Estas funciones pueden calcular el número de caracteres de una cadena, convertir una cadena a mayúsculas o a minúsculas, suprimir o añadir caracteres a la izquierda o a la derecha, etcétera.

A. Funciones que devuelven valores carácter

Estas funciones devuelven un carácter o un conjunto de caracteres; son un ejemplo las funciones que devuelven una cadena en mayúscula o una cadena en minúscula, o las que obtienen parte de una cadena. Se muestran en la Tabla 4.4.

Función	Propósito
CHR (n)	Devuelve el carácter cuyo valor en binario es equivalente a 'n'.
CONCAT (cad1, cad2)	Devuelve 'cad1' concatenada con 'cad2'. Es equivalente al operador ||.
LOWER (cad)	Devuelve la cadena 'cad' con todas sus letras convertidas a minúsculas.
UPPER (cad)	Devuelve la cadena 'cad' con todas sus letras convertidas a mayúsculas.
INITCAP (cad)	Convierte la cadena 'cad' a tipo título, la primera letra de cada palabra de 'cad' a mayúsculas y el resto, a minúsculas.
LPAD (cad1, n [, cad2])	Esta función añade caracteres a la izquierda de 'cad1' hasta que alcance una cierta longitud 'n'. Devuelve 'cad1' con longitud 'n' y ajustado a la derecha; 'cad2' es la cadena con la que se rellena por la izquierda; cad1 puede ser una columna de una tabla o cualquier literal. Si 'cad2' se suprime, asume como carácter de relleno el blanco.
RPAD (cad1, n [, cad2])	Añade caracteres a la derecha de 'cad1' hasta que alcance una cierta longitud 'n'. Devuelve 'cad1' con longitud 'n' y ajustado a la izquierda; 'cad2' es la cadena con la que se rellena por la derecha; 'cad1' puede ser una columna de una tabla o cualquier literal. Si 'cad2' se suprime, asume como carácter de relleno el blanco.
LTRIM (cad [, set])	Suprime un conjunto de caracteres a la izquierda de la cadena 'cad'; 'set' es el conjunto de caracteres a suprimir. Devuelve 'cad' con el grupo de caracteres 'set' omitidos por la izquierda. Si el segundo parámetro se omite, devuelve la misma cadena. Por defecto, si la cadena contiene blancos a la izquierda y se omite el segundo parámetro, la función devuelve la cadena sin blancos a la izquierda.
RTRIM (cad [, set])	Suprime un conjunto de caracteres a la derecha de la cadena 'cad'. Devuelve 'cad' con el grupo de caracteres 'set' omitidos por la derecha. Si se omite 'set', devuelve 'cad' tal como está. Por defecto, si la cadena contiene blancos a la derecha y se omite el segundo parámetro, la función devuelve la cadena sin blancos a la derecha.
REPLACE (cad, cadena_búsqueda [,cadena_sustitución])	Sustituye un carácter o varios caracteres de una cadena con 0 o más caracteres. Devuelve 'cad' con cada ocurrencia de 'cadena_búsqueda' sustituida por 'cadena_sustitución'.
SUBSTR (cad, m [,n])	Obtiene parte de una cadena. Devuelve la subcadena de 'cad', que abarca desde la posición indicada en 'm' hasta tantos caracteres como indique el número 'n'. Si se omite 'n', devuelve la cadena desde la posición especificada por 'm'. El valor de 'n' no puede ser inferior a 1. El valor de 'm' puede ser negativo; en ese caso, devuelve la cadena empezando por su final, y avanzando de derecha a izquierda.
TRANSLATE (cad1, cad2, cad3)	Convierte caracteres de una cadena en caracteres diferentes, según un plan de sustitución marcado por el usuario. Devuelve 'cad1' con los caracteres encontrados en 'cad2' y sustituidos por los caracteres de 'cad3'. Cualquier carácter que no esté en la cadena 'cad2' permanece como estaba.

Tabla 4.4. *Funciones que devuelven factores carácter.*

Caso práctico

5 **CHR(n).** Devuelve las letras cuyo valor ASCII es 75 y 65:

```
SQL> SELECT  CHR(75), CHR(65) FROM DUAL;

  C    C
  -    -
  K    A
```

CONCAT(cad1, cad2). Obtén el apellido de los empleados de la tabla EMPLE de la siguiente manera: El apellido es: APELLIDO.

Para ello necesitamos concatenar la cadena 'El apellido es: ' con la columna APELLIDO de la tabla EMPLE: `SQL> SELECT CONCAT('El apellido es: ', APELLIDO) FROM EMPLE;`

```
CONCAT('ELAPELLIDOES:',APE
--------------------------
El apellido es: SANCHEZ
El apellido es: ARROYO
El apellido es: SALA
El apellido es: JIMENEZ
El apellido es: MARTIN
El apellido es: NEGRO
El apellido es: CEREZO
El apellido es: GIL
El apellido es: REY
El apellido es: TOVAR
El apellido es: ALONSO
El apellido es: JIMENO
El apellido es: FERNANDEZ
El apellido es: MUÑOZ

14 filas seleccionadas.
```

LOWER(cad), UPPER(cad) e INITCAP(cad). Visualiza en mayúsculas, minúsculas y tipo título la cadena: 'oRACle Y sqL':

```
SQL> SELECT LOWER('oRACle Y sqL') "minuscula", UPPER('oRACle Y sqL') "MAYUSCULA",
INITCAP('oRACle Y sqL') "Tipo Titulo" FROM DUAL;

    minuscula           MAYUSCULA           Tipo Titulo
    ------------        ------------        ------------
    oracle y sql        ORACLE Y SQL        Oracle Y Sql
```

LPAD(cad1, n [, cad2]) y RPAD(cad1, n [, cad2]). Para cada fila de la tabla NOTAS_ALUMNOS, obtenemos en una columna el nombre del alumno con una longitud de 30 caracteres y rellenando por la izquierda con puntos y en otra columna lo mismo pero rellenando por la derecha: `SQL> SELECT LPAD(NOMBRE_ALUMNO,30,'.') "IZQUIERDA", RPAD(NOMBRE_ALUMNO,30,'.') "DERECHA" FROM NOTAS_ALUMNOS;`

```
IZQUIERDA                      DERECHA
-----------------------------  -----------------------------
......Alcalde García, M. Luisa Alcalde García, M. Luisa......
```

(Continúa)

(Continuación)

```
.....Benito Martín, Luis        Benito Martín, Luis.....
.....Casas Martínez, Manuel     Casas Martínez, Manuel.....
.....Corregidor Sánchez, Ana    Corregidor Sánchez, Ana.....
.....Díaz Sánchez, Maria        Díaz Sánchez, Maria.....
```

LTRIM(cad [, set]) y RTRIM(cad [, set]). Usamos las funciones LTRIM y RTRIM sin el segundo parámetro y con una cadena con blancos a la izquierda y a la derecha que, por defecto, eliminará:

```
SQL> SELECT LTRIM('       hola') || RTRIM('      adios  ')||'*' FROM DUAL;

      LTRIM('HOLA')||R
      ----------------
      hola      adios*
```

A partir de la tabla MISTEXTOS: `SQL> SELECT * FROM MISTEXTOS;`

```
TITULO                          AUTOR              EDITORIAL            PAGINA
------------------------------- -----------------  ---------------  ---------
METODOLOGÍA DE LA PROGRAMACIÓN.  ALCALDE GARCÍA     MCGRAWHILL             140
"INFORMÁTICA BÁSICA."            GARCÍA GARCERÁN    PARANINFO              130
SISTEMAS OPERATIVOS              GARCÍA ESTRUCH     OBSBORNE               300
SISTEMAS DIGITALES.              RUÍZ LOPEZ         PRENTICE HALL          190
"MANUAL DE C."                   RUÍZ LOPEZ         MCGRAWHILL             340
```

Quitamos los caracteres punto y comilla de la derecha de la columna TITULO:

```
SQL> SELECT RTRIM (TITULO, '."' ) FROM MISTEXTOS;

  RTRIM(TITULO,'."')
  --------------------------------
  METODOLOGÍA DE LA PROGRAMACIÓN
  "INFORMÁTICA BÁSICA
  SISTEMAS OPERATIVOS
  SISTEMAS DIGITALES
  "MANUAL DE C
```

Quitamos las comillas de la izquierda de la columna TITULO:

```
SQL> SELECT LTRIM (TITULO, '"' ) FROM MISTEXTOS;

  LTRIM(TITULO,'"')
  --------------------------------
  METODOLOGÍA DE LA PROGRAMACIÓN.
  INFORMÁTICA BÁSICA."
  SISTEMAS OPERATIVOS
  SISTEMAS DIGITALES.
  MANUAL DE C."
```

REPLACE(cad, cadena_búsqueda [,cadena_sustitución]). Sustituimos 'O' por 'AS' en la cadena 'BLANCO Y NEGRO':

```
SQL> SELECT  REPLACE('BLANCO Y NEGRO', 'O', 'AS')  FROM  DUAL;
```

(Continúa)

(Continuación)

```
REPLACE('BLANCOY
---------------
BLANCAS Y NEGRAS
```

Si no ponemos nada en la cadena_sustitución, se sustituiría la cadena_búsqueda por nada (NULL). En el siguiente ejemplo, sustituimos 'O' por nada:

```
SQL> SELECT  REPLACE('BLANCO Y NEGRO', 'O') FROM DUAL;

        REPLACE('BLA
        ------------
        BLANC Y NEGR
```

SUBSTR(cad, m [,n]). Partiendo de la cadena 'ABCDEF', obtenemos en una columna dos caracteres a partir de la tercera posición, en otra columna otros dos caracteres a partir de la tercera posición empezando por el final de la cadena y en una última columna la cadena a partir de su cuarta posición:

```
SQL>  SELECT  SUBSTR('ABCDEFG',3,2)  "s1",  SUBSTR  ('ABCDEFG',-3,2)  "s2",
SUBSTR('ABCDEFG',4) "s3" FROM DUAL;

        s1        s2        s3
        --        --        ----
        CD        EF        DEFG
```

Visualiza el apellido y su primera letra para los empleados del departamento 10 de la tabla EMPLE:

```
SQL> SELECT APELLIDO, SUBSTR(APELLIDO,1,1) FROM EMPLE WHERE DEPT_NO =10;

  APELLIDO         S
  ----------       -
  CEREZO           C
  REY              R
  MUÑOZ            M
```

TRANSLATE(cad1, cad2 , cad3). A partir de la cadena 'LOS PILARES DE LA TIERRA', sustituye la 'A' por 'a', la 'E' por 'e', la 'I' por 'i', la 'O' por 'o', y la 'U' por 'u':

```
SQL> SELECT TRANSLATE('LOS PILARES DE LA TIERRA','AEIOU','aeiou') FROM DUAL;

  TRANSLATE('LOSPILARESDEL
  ------------------------
  LoS PiLaReS De La TieRRa
```

En este ejemplo TRANSLATE sustituye la 'L' por 'l':

```
SQL> SELECT TRANSLATE('LOS PILARES DE LA TIERRA','LAEIOU','1') FROM DUAL;

  TRANSLATE('LOSP
  ---------------
  1S P1RS D 1 TRR
```

Ahora la cadena "cad2" está formada por 'LAEIOU', pero en la cadena "cad3" sólo existe 'l', lo que hace que desaparezcan las vocales en la cadena resultante; ya que sólo la 'L' se reemplaza por 'l'; el resto, 'AEIOU', se reemplaza por nada.

B. Funciones que devuelven valores numéricos

Estas funciones devuelven valores numéricos, como el número de caracteres que tiene una cadena o la posición en la que se encuentra un determinado carácter en una cadena. Se trata de las siguientes mostradas en la Tabla 4.5.

Función	Propósito
ASCII(cad)	Devuelve el valor ASCII de la primera letra de la cadena 'cad'.
INSTR(cad1,cad2 [,comienzo [,m]])	Esta función busca un conjunto de caracteres en una cadena. Devuelve la posición de la 'm_ésima' ocurrencia de 'cad2' en 'cad1', empezando la búsqueda en la posición 'comienzo'. Por omisión, empieza buscando en la posición 1.
LENGTH(cad)	Devuelve el número de caracteres de 'cad'.

Tabla 4.5. *Funciones que devuelven valores numéricos.*

Caso práctico

6 **ASCII(cad).** Obtén el valor ASCII de 'A': `SQL> SELECT ASCII ('A') FROM DUAL;`

```
ASCII('A')
----------
        65
```

Si ponemos como argumento una cadena de caracteres, visualiza sólo el valor ASCII del primer carácter de la cadena.

INSTR(cad1, cad2 [,comienzo [,m]]). A partir de la cadena 'II VUELTA CICLISTA A TALAVERA' encuentra la segunda ocurrencia 'TA' desde la posición 3:

```
SQL> SELECT INSTR('II VUELTA CICLISTA A TALAVERA', 'TA', 3, 2) "EJEMPLO" FROM DUAL;

   EJEMPLO
----------
        17
```

Cuando «comienzo» es negativo (-1), se comienza la búsqueda en la posición final y se va de derecha a izquierda en la cadena. Devuelve la posición contando desde la izquierda, es decir, la primera A que encuentra desde la primera posición empezando por el final:

```
SQL> SELECT INSTR('II VUELTA CICLISTA A TALAVERA', 'A', -1) "EJEMPLO" FROM DUAL;

   EJEMPLO
----------
        29
```

(Continúa)

(Continuación)

Si en el ejemplo anterior empleamos un –2, haría que la función comenzase desde la segunda posición empezando por el final; un –3 haría que se iniciase desde la tercera posición y, así, sucesivamente.

A partir de la tabla MISTEXTOS, encuentra la posición de la segunda ocurrencia de la letra 'A' en la columna AUTOR a partir del comienzo:

```
SQL> SELECT AUTOR, INSTR(AUTOR,'A',1,2) FROM MISTEXTOS;

AUTOR                INSTR(AUTOR,'A',1,2)
-------------------- --------------------
ALCALDE GARCÍA                          4
GARCÍA GARCERÁN                         6
GARCÍA STRUCH                           6
RUÍZ LOPEZ                              0
RUÍZ LOPEZ                              0
```

Si en la función INSTR, "cad2" es un conjunto de caracteres, entonces la función devuelve la posición donde comienza el primer carácter de ese conjunto.

LENGTH(cad). Calcula el número de caracteres de las columnas TITULO y AUTOR para todas las filas de la tabla MISTEXTOS:

```
SQL> SELECT TITULO, LENGTH(TITULO), AUTOR, LENGTH(AUTOR) FROM MISTEXTOS;

TITULO                         LENGTH(TITULO) AUTOR           LENGTH(AUTOR)
------------------------------ -------------- --------------- -------------
METODOLOGÍA DE LA PROGRAMACIÓN.             31 ALCALDE GARCÍA               14
"INFORMÁTICA BÁSICA."                       21 GARCÍA GARCERÁN              15
SISTEMAS OPERATIVOS                         19 GARCÍA STRUCH                13
SISTEMAS DIGITALES.                         19 RUÍZ LOPEZ                   10
"MANUAL DE C."                              14 RUÍZ LOPEZ                   10
```

4.4 Funciones para el manejo de fechas

Oracle puede almacenar datos de tipo fecha (DATE) y posee una interesante utilidad para formatear las fechas de cualquier manera que se pueda concebir. Tiene un formato por omisión: 'DD/MM/YY', pero con la función TO_CHAR es posible mostrar las fechas de cualquier modo. Los literales de fecha deben encerrarse entre comillas simples.

Ejemplo: '18/11/05'.

Recordemos que el tipo de datos DATE se almacena en un formato especial que incluye Siglo/Año/Mes/Día/Hora/Minutos/Segundos. Las funciones para el manejo de fechas se exponen en la Tabla 4.6.

Función	Propósito
SYSDATE	Devuelve la fecha del sistema.
ADD_MONTHS(fecha, n)	Devuelve la fecha 'fecha' incrementada en 'n' meses.
LAST_DAY(fecha)	Devuelve la fecha del último día del mes que contiene 'fecha'.
MONTHS_BETWEEN (fecha1, fecha2)	Devuelve la diferencia en meses entre las fechas 'fecha1' y 'fecha2'.
NEXT_DAY(fecha, cad)	Devuelve la fecha del primer día de la semana indicado por 'cad' después de la fecha indicada por 'fecha'. El día de la semana en 'cad' se indica con su nombre, es decir, lunes (*monday*), martes (*tuesday*), miércoles (*wednesday*), jueves (*thursday*), viernes (*friday*), sábado (*saturday*) o domingo (*sunday*).

Tabla 4.6. *Funciones para el manejo de fechas.*

Caso práctico

7 **SYSDATE.** Esta función devuelve la fecha del sistema. Por ejemplo:

```
SQL> SELECT SYSDATE FROM DUAL;

 SYSDATE
--------
03/08/05
```

ADD_MONTHS(fecha, n). A partir de la tabla EMPLE, suma doce meses a la fecha de alta para los empleados del departamento 10:

```
SQL> SELECT FECHA_ALT, ADD_MONTHS(FECHA_ALT, 12)  FROM EMPLE WHERE DEPT_NO=10;

FECHA_AL            ADD_MONT
--------            --------
09/06/91            09/06/92
17/11/91            17/11/92
23/01/92            23/01/93
```

LAST_DAY(fecha). Obtén de la tabla EMPLE el último día del mes para cada una de las fechas de alta de los empleados del departamento 10:

```
SQL> SELECT FECHA_ALT, LAST_DAY(FECHA_ALT) FROM EMPLE WHERE DEPT_NO=10;

FECHA_AL            LAST_DAY
--------            --------
09/06/91            30/06/91
17/11/91            30/11/91
23/01/92            31/01/92
```

(Continúa)

(Continuación)

MONTHS_BETWEEN(fecha1, fecha2). Cálculo de la edad: necesitamos la función "SYSDATE", que devuelve la fecha actual (fecha del sistema) y calculamos los meses transcurridos entre la fecha de hoy y la fecha de nacimiento. Dividimos entre 12 ese resultado y aplicamos la función TRUNC para suprimir decimales:

```
SQL> SELECT TRUNC (MONTHS_BETWEEN (SYSDATE,'18/11/1964') / 12) "Edad actual" FROM DUAL;

    Edad actual
    -----------
             40
```

NEXT_DAY(fecha, cad). Si hoy es jueves 19 de octubre de 2006 (fecha del sistema «sysdate»), ¿qué fecha será el próximo jueves?

```
SQL> SELECT NEXT_DAY(SYSDATE, 'JUEVES')  "Sig. Jueves"  FROM  DUAL;

       Sig. Jue
       --------
       26/10/06
```

4.5 Funciones de conversión

La mayoría de las funciones que hemos visto hasta ahora son funciones de transformación, esto es, cambian los objetos. Hay otras funciones que cambian los objetos de una manera especial, pues transforman un tipo de dato en otro. Las *funciones de conversión* elementales se muestran en la Tabla 4.7. La Tabla 4.8 muestra las máscaras de formato para las fechas y la Tabla 4.9 muestra las máscaras de formatos numéricos.

Función	Propósito
TO_CHAR (fecha, 'formato')	Convierte una _fecha_ (de tipo DATE) a tipo _VARCHAR2_ en el _formato_ especificado. El _formato_ es un cadena de caracteres que puede incluir las máscaras de formato definidas en la *Tabla de control de formato de fechas* (Tabla 4.8), y donde es posible incluir literales definidos por el usuario encerrados entre comillas dobles.
TO_CHAR (numero, 'formato')	Esta función convierte un 'número' (de tipo NUMBER) a tipo VARCHAR2 en el 'formato' especificado. Los formatos numéricos se muestran en la Tabla 4.9.
TO_DATE (cad, 'formato')	Convierte 'cad', de tipo VARCHAR2 o CHAR, a un valor de tipo DATE según el 'formato' especificado.
TO_NUMBER (cadena ['formato'])	Convierte la 'cadena' a tipo NUMBER según el 'formato' especificado. La cadena ha de contener números, el carácter decimal o el signo menos a la izquierda. No puede haber espacios entre los números, ni otros caracteres.

Tabla 4.7. *Funciones de conversión.*

Máscaras de formato numéricas	
cc o scc	Valor del siglo
y, yyy ó sy,yyy	Año con coma, con o sin signo
yyyy	Año sin signo
yyy	Últimos tres dígitos del año
yy	Últimos dos dígitos del año
y	Último dígito del año
q	Número del trimestre
ww	Número de semana del año
w	Número de semana del mes
mm	Número de mes
ddd	Número de día del año
dd	Número de día del mes
d	Número de día de la semana
hh ó hh12	Hora (1-12)
hh24	Hora (1-24)
mi	Minutos
ss	Segundos
sssss	Segundos transcurridos desde medianoche
j	Juliano
Máscaras de formato de caracteres	
syear ó year	Año en inglés (ej: *nineteen-eighty-two*)
month	Nombre del mes (ENERO)
mon	Abreviatura de tres letras del nombre del mes (ENE)
day	Nombre del día de la semana (LUNES)
dy	Abreviatura de tres letras del nombre del día (LUN)
a.m. o p.m.	Muestra a.m. ó p.m. dependiendo del momento del día
b.c. o a.d.	Indicador para el año (antes de Cristo o después de Cristo)

Tabla 4.8. *Máscaras de control de formatos de fechas.*

Caso práctico

8 **A partir de la tabla EMPLE, obtén la fecha de alta (columna FECHA_ALT) formateada, de manera que aparezca el nombre del mes con todas sus letras (month), el número de día de mes (dd) y el año (yyyy), para aquellos empleados del departamento 10:**

```
SQL> SELECT TO_CHAR (FECHA_ALT, 'month DD, YYYY') "NUEVA FECHA" FROM EMPLE WHERE
DEPT_NO=10;

                              NUEVA      FECHA
                              ------------------
                              junio      09,  1991
                              noviembre  17,  1991
                              enero      23,  1992
```

Hay que hacer una observación: el nombre del mes aparece en minúscula porque en el formato se definió *month* en minúscula, pero también puede aparecer en mayúscula o con la primera letra mayúscula y las siguientes en minúsculas. Las opciones para *month* son las siguientes: si el nombre de mes es Enero, Month produce Enero, month produce enero, MONTH produce ENERO. Lo mismo ocurre con day y dy.

(Continúa)

(Continuación)

Ahora se desea obtener la fecha de alta de forma que aparezca el nombre del mes con tres letras (mon), el número de día del año (ddd), el último dígito del año (y) y los tres últimos dígitos del año (yyy):

```
SQL> SELECT TO_CHAR (FECHA_ALT, 'mon ddd y yyy') "FECHA" FROM EMPLE WHERE DEPT_NO=10;

  FECHA
  -------------
  jun 160 1 991
  nov 321 1 991
  ene 023 2 992
```

Por defecto, el formato para la fecha viene definido por el parámetro **NLS_TERRITORY**, que especifica el idioma para el formato de la fecha, los separadores de miles, el signo decimal y el símbolo de la moneda. Este parámetro se inicializa al arrancar Oracle. Para el idioma español, el valor de este parámetro es: NLS_TERRITORY=SPAIN.

Podemos cambiar el valor por omisión para la fecha con el parámetro **NLS_DATE_FORMAT**, usando la orden **ALTER SESSION**. Por ejemplo, para cambiar el formato de la fecha y que aparezca de la siguiente manera: día/nombre mes/ año hora:minutos:segundos utilizaremos la siguiente sentencia:

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT='DD/month/YYYY
HH24:MI:SS';
Sesión modificada.
SQL> SELECT SYSDATE FROM DUAL;

SYSDATE
---------------------------
03/agosto    /2005 17:20:48
```

También podemos cambiar el lenguaje utilizado para nombrar los meses y los días con el parámetro **NLS_DATE_LANGUAGE**.

Caso práctico

9 **Por defecto, al iniciar nuestra sesión el idioma definido para la fecha es el español. Podemos definir otro idioma, por ejemplo, el francés, de la siguiente manera:** `SQL> ALTER SESSION SET NLS_DATE_LANGUAGE=French;`

Si queremos visualizar el nombre del mes y el día de la semana, éstos aparecerán en francés, con lo que la fecha de hoy nos quedará así: `SQL> SELECT TO_CHAR(sysdate,'"Hoy es " day "," dd " de " month " de " yyyy') "Fecha" FROM DUAL;`

```
Fecha
-----------------------------------------------
Hoy es  mercredi , 03   de   août       de  2005
```

Elemento	Ejemplo	Descripción

9	999	Devuelve el valor con el número especificado de dígitos. Si es positivo, deja un espacio.
		Devuelve el valor con el número especificado de dígitos con el signo menos si es negativo.
		Si el valor tiene ceros a la izquierda, los deja en blanco, excepto si el valor es 0.
		`SQL> SELECT TO_CHAR(1,'999'), TO_CHAR(-1,'999'), TO_CHAR(01,'999') , TO_CHAR(0,'999') FROM DUAL;`
		`TO_C TO_C TO_C TO_C`
		`---- ---- ---- ----`
		`1 -1 1 0`

0	9990
	999
		9990 - Muestra un 0 si el valor es 0.
		0999 - Devuelve el valor dejando ceros al principio.
		`SQL> SELECT TO_CHAR(10,'0999'), TO_CHAR(10,'9990') ,TO_CHAR(10,'990090') FROM DUAL;`
		`TO_CH TO_CH TO_CHAR`
		`----- ----- -------`
		`0010 10 0010`

$	$9999	Devuelve el valor con el signo dólar a la izquierda.
		`SQL> SELECT TO_CHAR(10,'$9999'),TO_CHAR(10,'$009'), TO_CHAR(10,'99$') FROM DUAL;`
		`TO_CHA TO_CH TO_C`
		`------ ----- ----`
		`$10 $010 $10`

B	B999	Muestra un espacio en blanco si el valor es 0. Es el formato por omisión.
		`SQL> SELECT TO_CHAR(0,'B999'), TO_CHAR(5,'B999') FROM DUAL;`
		`TO_C TO_C`
		`---- ----`
		`5`

MI	999MI	Si el número es negativo, el signo menos sigue al número. Por omisión, el signo se pone a la izquierda.
		`SQL> SELECT TO_CHAR(-55,'999MI'), TO_CHAR (55, '999MI') FROM DUAL;`
		`TO_C TO_C`
		`---- ----`
		`55- 55`

S	S999
	
	999S
		'S' representa el signo. Devuelve el valor con el signo '+' si el valor es positivo o con el signo '_' si es negativo.
		`SQL> SELECT TO_CHAR(-55,'999S'), TO_CHAR(-55, 'S999'), TO_CHAR(55,'S999'), TO_CHAR(55,'999S') FROM DUAL;`
		`TO_C TO_C TO_C TO_C`
		`---- ---- ---- ----`
		`55- -55 +55 55+`

PR	9999PR	Los números negativos se muestran entre estos símbolos: < >.
		`SQL> SELECT TO_CHAR(-55,'9999PR'), TO_CHAR(55, '9999PR') FROM DUAL;`
		`TO_CHA TO_CHA`
		`------ ------`
		`<55> 55`

D	99D99	Devuelve el carácter decimal en la posición especificada.
		`SQL> SELECT TO_CHAR(34.55,'99D99') FROM DUAL;`
		`TO_CHA`
		`------`
		`34,55`

Tabla 4.9. *Tabla de formatos numéricos.*

Elemento	Ejemplo	Descripción
G	9G999	Devuelve el carácter de grupo (carácter de los miles) en la posición especificada. SQL> SELECT TO_CHAR(1234,'9G999') FROM DUAL; TO_CHA ------ 1.234 SQL> SELECT TO_CHAR(123456.98, '999G999D99') FROM DUAL; TO_CHAR(123 ---------- 123.456,98
C	C999	Devuelve el símbolo ISO del territorio en la posición especificada. SQL> SELECT TO_CHAR(123,'C999') "ISO" FROM DUAL; ISO ------ EUR123
L	L999 999L	Devuelve el símbolo de la moneda local en la posición indicada. SQL> SELECT TO_CHAR(123,'L999') "MONEDA" FROM DUAL; MONEDA -------- € 123
, (coma)	9,999	Devuelve la coma en la posición especificada (carácter de los miles). SQL> SELECT TO_CHAR(1234,'9,999') FROM DUAL; TO_CHA ------ 1,234
. (punto)	99.99	Devuelve el punto decimal en la posición especificada. SQL> SELECT TO_CHAR(12.34,'99.99') FROM DUAL; TO_CHA ------ 12.34 SQL> SELECT TO_CHAR(12345.67,'99,999.99') FROM DUAL; TO_CHAR(12 --------- 12,345.67
V	999V99	Devuelve el valor multiplicado por 10n, donde 'n' es el número de nueves después 'V'. SQL> SELECT TO_CHAR(123.45,'999V99'), TO_CHAR(123,'999V99') FROM DUAL; TO_CHA TO_CHA ------ ------ 12345 12300
EEEE	9.9EEEE	Devuelve el valor usando notación científica. SQL> SELECT TO_CHAR(12345,'9.9EEEE') FROM DUAL; TO_CHAR(1 ------- 1.2E+04
RN rn	RN	Devuelve el valor en números romanos. 'RN' devuelve el valor en mayúsculas y 'rn', en minúsculas. SQL> SELECT TO_CHAR(12,'RN') , TO_CHAR(12,'rn') FROM DUAL; TO_CHAR(12,'RN' TO_CHAR(12,'RN' --------------- --------------- XII xii
FM	FM90.9	Devuelve el valor alineado a la izquierda. SQL>SELECT TO_CHAR(12.8,'FM90.9'),TO_CHAR(12,'FM99'), TO_CHAR(-12,'FM99') FROM DUAL; TO_CH TO_ TO_ ----- --- --- 12.8 12 -12

Tabla 4.9 *(Continuación). Tabla de formatos numéricos.*

Los caracteres devueltos en algunos de estos formatos se especifican inicializando una serie de parámetros. Estos parámetros se muestran en la Tabla 4.10.

Elemento	Elemento	Descripción
NLS_NUMERIC_ CHARACTERS	D, G	Define los caracteres decimal ('D') y separador de los miles ('G'). Formato: NLS_NUMERIC_CHARACTERS= 'DG' ____D: carácter decimal. ____G: separador de miles. Ejemplo:____NLS_NUMERIC_CHARACTERS= ',.' Carácter decimal, la coma, y separador de miles, el punto.
NLS_ISO_CURRENCY	C	Especifica el símbolo del territorio. Para Espana el símbolo es 'ESP'.
NLS_CURRENCY	L	Especifica el símbolo de la moneda local. Para España es '€'.

Tabla 4.10. *Parámetros NLS.*

Para cambiar el valor de estos parámetros se utiliza la orden ALTER SESSION. Suponga-mos, por ejemplo, que queremos definir como carácter de los miles el asterisco (*) y como carácter decimal la barra (/). Hemos de usar el parámetro NLS_NUMERIC_CHARACTERS con ALTER SESSION:

```
SQL> ALTER SESSION SET NLS_NUMERIC_CHARACTERS='/*';
Sesión modificada.
SQL>  SELECT TO_CHAR(12345.67,'999G999D999') FROM DUAL;

TO_CHAR(1234
------------
  12*345/670
```

Los valores para el carácter decimal y de los miles permanecerán hasta que el usuario finalice la sesión o hasta que el usuario aplique de nuevo la orden ALTER SESSION para cambiar estos caracteres:

```
SQL> ALTER SESSION SET NLS_NUMERIC_CHARACTERS=',.';
```

Si queremos cambiar el símbolo de la moneda, de manera que, en lugar de '€' aparezca 'Pesetas', usamos el parámetro NLS_CURRENCY. Ejemplo:

```
SQL> ALTER SESSION SET NLS_CURRENCY='PESETAS';
Sesión modificada.
SQL> SELECT TO_CHAR(123,'L999') FROM DUAL;

TO_CHAR(123,'L
--------------
    PESETAS123
```

Caso práctico

10 **TO_NUMBER(cadena [,'formato']).** Suponemos que el carácter decimal es la coma y el carácter separador de los miles, el punto.

```
SQL> SELECT TO_NUMBER('-123456') "NUMERO1", TO_NUMBER('123,99','999D99') "NUMERO2"
FROM DUAL;

          NUMERO1         NUMERO2
        ---------       ---------
          -123456           123,99

SQL> SELECT TO_NUMBER('123.456','999G999') "NO CONVIERTE" FROM DUAL;

     NO CONVIERTE
     ------------
           123456
```

Este ejemplo no convierte porque la cadena '123.456' contiene el carácter que define el separador de los miles (en este ejemplo, el punto), y una cadena válida ha de contener el carácter decimal (en este caso la coma).

TO_DATE(cad , 'formato'). Cambia el formato de la fecha para que aparezca el año con cuatro dígitos:

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT='DD/MM/YYYY';
```

Convertir una cadena a tipo DATE: `SQL> SELECT TO_DATE('01012006') FROM DUAL;`

```
TO_DATE('0
----------
01/01/2006
```

Cuando en la orden TO_DATE no se indica el formato, una cadena de caracteres será convertida a fecha sólo si está en el formato que tenga la fecha del sistema. En el siguiente ejemplo no se convierte la cadena a tipo fecha porque no está en el formato 'DDMMYYYY' definido en la sesión para la fecha: `SQL> SELECT TO_DATE('010106') FROM DUAL;`

```
SELECT TO_DATE('010106') FROM DUAL
       *
ERROR en línea 1:
ORA-01861: el literal no coincide con la cadena de formato
```

Lo correcto sería: `SQL> SELECT TO_DATE('01012006') FROM DUAL;`

Obtén el nombre del mes a partir de la cadena '01012007' (antes hay que convertir la cadena a tipo fecha): `SQL> SELECT TO_CHAR(TO_DATE ('01012007', 'ddmmyyyy'), 'Month') "MES" FROM DUAL;`

```
MES
----------
Enero
```

TO_DATE y TO_CHAR son similares; la diferencia que los separa estriba en que TO_DATE convierte una cadena de caracteres en una fecha y TO_CHAR convierte una fecha en una cadena de caracteres. Ambas pueden utilizar las máscaras de formato de fechas.

4.6 Otras funciones

A continuación, se exponen otras funciones que, por sus características, no se han incluido en los grupos anteriores, aunque no por ello dejan de ser útiles.

- **DECODE (var, val1, cod1, val2, cod2..., valor_por_defecto)**

Esta función sustituye un valor por otro. Si «var» es igual a cualquier valor de la lista («val1», «val2»...), devuelve el correspondiente código («cod1», «cod2»...). En caso contrario se obtiene el valor señalado como valor por defecto («valor_por_defecto»); «val» debe ser un dato del mismo tipo de «var». Ésta es una función IF - THEN - ELSE.

- **VSIZE (expresión)**

Devuelve el número de bytes que ocupa expresión. Si expresión es nulo, la función devuelve nulo.

- **DUMP (cadena[, formato [, comienzo[, longitud]]])**

Esta función visualiza el valor de «cadena», que puede ser un literal o una expresión, en formato de datos interno, en ASCII, octal, decimal, hexadecimal o en formato de carácter. Por defecto, el formato es ASCII o EBCDIC, lo que depende de la máquina. El argumento «formato» puede tener los siguientes valores:

8 devuelve el resultado en octal.
10 devuelve el resultado en decimal.
16 devuelve el resultado en hexadecimal.
17 devuelve el resultado en formato carácter (ASCII o EBCDIC).

«Comienzo» es la posición de inicio de la cadena y «longitud» es el número de caracteres que se van a visualizar.

- **USER**

Esta función devuelve el nombre del usuario actual.

- **UID**

Devuelve el identificador del usuario actual. Al crear un usuario, Oracle le asigna un número. Este número identifica a cada usuario y es único en la base de datos.

Aunque USER, UID y SYSDATE se han incluido entre las funciones, realmente son *pseudocolumnas*. Una **pseudocolumna** es una 'columna' que devuelve un valor al seleccionarla, pero que no es una columna actual de una tabla.

Caso práctico

11 **Sea la tabla EMPLE. Seleccionar todas las filas y codificar el OFICIO. Si el oficio es PRESIDENTE, codificar con un 1; si es EMPLEADO, con un 2; en cualquier otro caso, codificar con un 5:**

```
SQL> SELECT APELLIDO, OFICIO, DECODE(UPPER(OFICIO), 'PRESIDENTE', 1, 'EMPLEADO',
2, 5) "Codigo" FROM EMPLE;

APELLIDO       OFICIO            Codigo
----------     ----------     ----------
SANCHEZ        EMPLEADO               2
ARROYO         VENDEDOR               5
SALA           VENDEDOR               5
JIMENEZ        DIRECTOR               5
MARTIN         VENDEDOR               5
NEGRO          DIRECTOR               5
CEREZO         DIRECTOR               5
GIL            ANALISTA               5
REY            PRESIDENTE             1
TOVAR          VENDEDOR               5
ALONSO         EMPLEADO               2
JIMENO         EMPLEADO               2
FERNANDEZ      ANALISTA               5
MUÑOZ          EMPLEADO               2

14 filas seleccionadas.
```

Calculamos el número de bytes que tiene la columna APELLIDO de la tabla EMPLE para aquellos empleados del departamento 10:

```
SQL> SELECT APELLIDO, VSIZE(APELLIDO) BYTES FROM EMPLE WHERE DEPT_NO=10;

APELLIDO         BYTES
---------       -------
CEREZO                6
REY                   3
MUÑOZ                 5
```

Representamos en hexadecimal los caracteres 1 al 4 del APELLIDO 'SALA' de la tabla EMPLE:

```
SQL> SELECT APELLIDO, DUMP(APELLIDO,16,1,4) FROM EMPLE WHERE APELLIDO LIKE 'SALA';

APELLIDO       DUMP(APELLIDO,16,1,4)
----------     -------------------------
SALA           Typ=1 Len=4: 53,41,4c,41
```

Visualiza el usuario que está conectado y su identificador:

```
SQL> SELECT USER, UID FROM DUAL;

USER                UID
-------           ------
SCOTT                 57
```

La orden SHOW USER muestra el nombre de usuario que está conectado:

```
SQL> SHOW USER
```