UNIDAD 3: FUNCIONES DE UNA SOLA FILA

INTRODUCCION

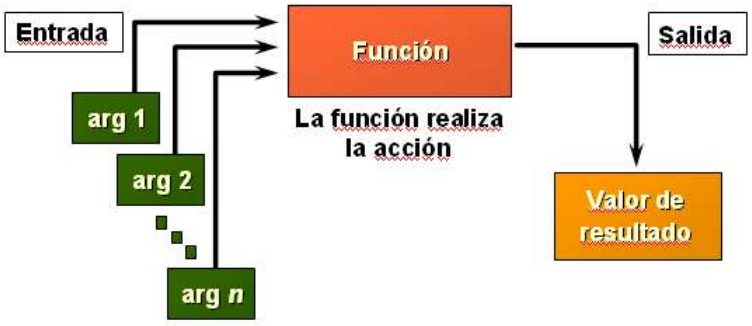
Las funciones hacen más potente el bloque de consulta básico y se utilizan para manipular valores de datos. Esta unidad se centra en funciones de caracteres, numéricas y de fecha de una sola fila, así como en otras funciones que convierten datos de un tipo en otro; por ejemplo, datos de caracteres en datos numéricos.

FUNCIONES SQL

Las funciones son unas funcionalidades potentes de SQL y se pueden utilizar para lo siguiente:

- Realizar cálculos sobre datos
- Modificar elementos de datos individuales
- Manipular el resultado para grupos de filas
- Formatear fechas y números para su visualización
- Convertir tipos de dato de columna

Las funciones SQL a veces toman argumentos y siempre devuelven un valor. La mayor parte de las funciones descritas en esta unidad son específicas a la versión Oracle de SQL.



DOS TIPOS DE FUNCIONES SQL

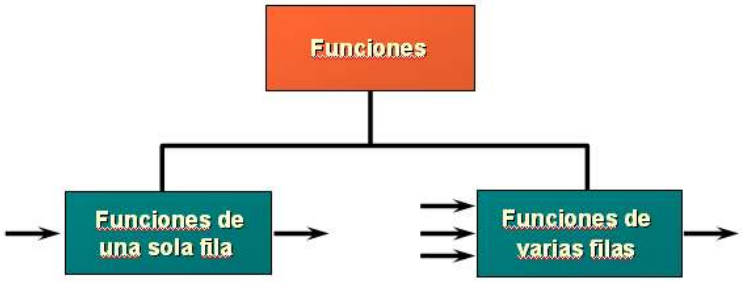
Existen dos tipos distintos de funciones:

- Funciones de una sola fila
- Funciones de varias filas

Funciones de una Sola Fila: estas funciones solamente operan en una fila y devuelven un resultado por fila. Existen distintos tipos de funciones de una sola fila. En esta unidad se cubren las siguientes:

- Carácter
- Número
- Fecha
- Conversión

Funciones de Varias Filas: estas funciones pueden manipular grupos de filas para proporcionar un resultado por cada uno de ellos. Se conocen como funciones de grupo y se tratan en una unidad posterior.



FUNCIONES DE UNA SOLA FILA

Las funciones de una sola fila se utilizan para manipular elementos de datos. Aceptan uno o varios argumentos y devuelven un valor para cada fila devuelta por la consulta. El argumento puede ser uno de los siguientes:

- Constante proporcionada por el usuario.
- Valor de variable.
- Nombre de columna.
- Expresión.

Las funcionalidades de las funciones de una sola fila incluyen:

- Actuar sobre cada fila devuelta en la consulta
- Devolver un resultado por fila
- Devolver posiblemente un valor de datos de un tipo diferente al de referencia
- Esperar posiblemente uno o varios argumentos
- Se pueden utilizar en cláusulas SELECT, WHERE y ORDER BY; se pueden anidar

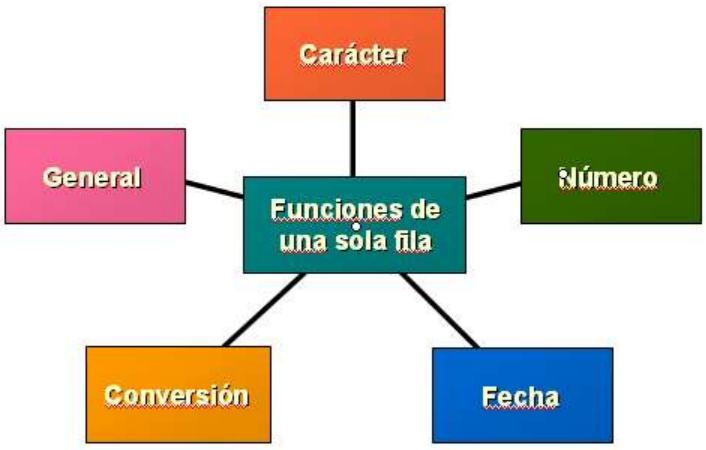
En la sintaxis:

```
function_name [(arg1, arg2,...)]
```

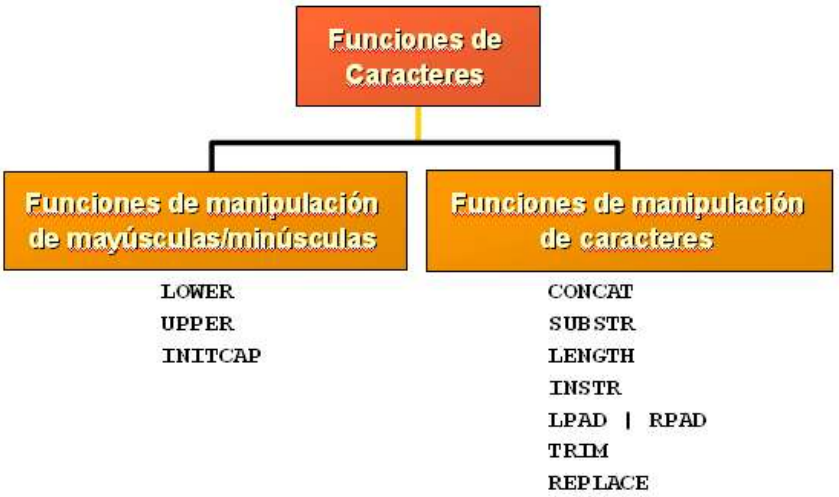
function_name	es el nombre de la función.
arg1, arg2	es cualquier argumento que debe utilizar la función. Puede venir representado por un nombre de columna o una expresión.

Esta unidad cubre las siguientes funciones de una sola fila:

- Funciones de caracteres: Aceptan entradas de caracteres y pueden devolver valores de caracteres y numéricos.
- Funciones numéricas: Aceptan entradas numéricas y devuelven valores numéricos.
- Funciones de fecha: Operan sobre valores del tipo de dato DATE. (Todas las funciones de fecha devuelven un valor del tipo de dato DATE excepto la función MONTHS_BETWEEN, que devuelve un número).
- Funciones de conversión: Convierten un valor de un tipo de dato en otro.
- Funciones generales:
  - NVL
  - NVL2
  - NULLIF
  - COALSECE
  - CASE
  - DECODE



FUNCIONES DE CARACTERES



FUNCIONES DE MANIPULACION DE MAYUSCULAS/MINUSCULAS

LOWER, UPPER e INITCAP son las tres funciones de conversión de mayúsculas/minúsculas.

- LOWER: Convierte cadenas de caracteres en mayúsculas o mezclados a minúsculas.
- UPPER: Convierte cadenas de caracteres en minúsculas o mezclados a mayúsculas.
- INITCAP: Convierte la primera letra de cada palabra a mayúsculas y las restantes letras a minúsculas.

Function	Result
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course

El siguiente ejemplo muestra el número de empleado, el nombre y el número de departamento del empleado Higgins.

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  last_name = 'higgins';
no rows selected
```

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  LOWER(last_name) = 'higgins';
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
205	Higgins	110

La cláusula WHERE de la primera sentencia SQL especifica el nombre de empleado como higgins. Como todos los datos de la tabla EMPLOYEES se almacenan con las mayúsculas/minúsculas correspondientes, el nombre higgins no encuentra ninguna coincidencia en la tabla y no se selecciona ninguna fila.

La cláusula WHERE de la segunda sentencia SQL especifica que el nombre del empleado de la tabla EMPLOYEES se compara con higgins, convirtiendo la columna LAST_NAME en minúsculas para la comparación. Como ahora están en minúsculas los dos nombres, se encuentra una coincidencia y se selecciona una fila. La cláusula WHERE se puede reescribir de la siguiente manera para proporcionar el mismo resultado:

WHERE last_name = 'Higgins'

El apellido del resultado aparece como se almacenó en la base de datos. Para mostrar el apellido en mayúsculas, se debe utilizar la función UPPER en la sentencia SELECT:

```
SELECT employee_id, UPPER(last_name), department_id
FROM   employees
WHERE  INITCAP(last_name) = 'Higgins';
```

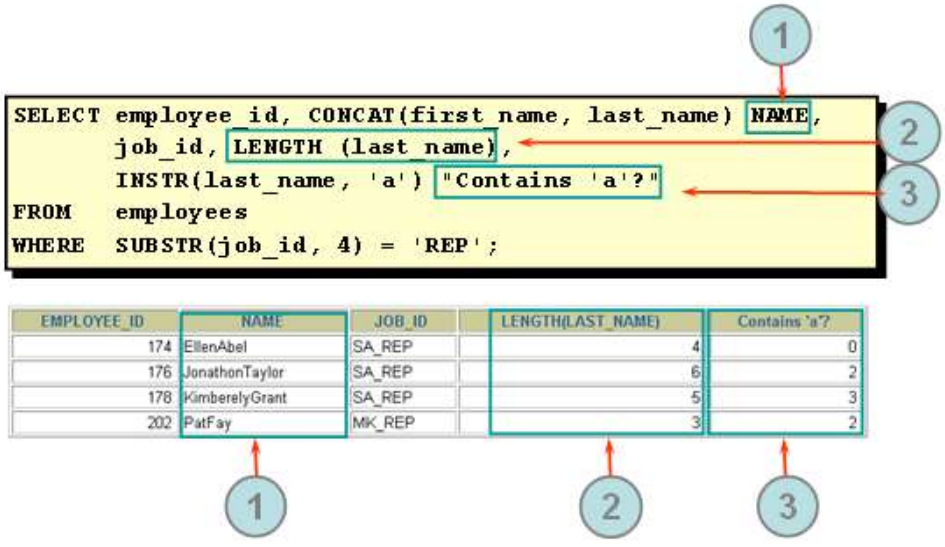
FUNCIONES DE MANIPULACION DE CARACTERES

CONCAT, SUBSTR, LENGTH, INSTR, LPAD, RPAD y TRIM son las funciones de manipulación de caracteres que se tratan en esta lección.

- CONCAT: Une valores (con está función está limitado a utilizar dos parámetros.)
- SUBSTR: Extrae una cadena de una longitud determinada.
- LENGTH: Muestra la longitud de una cadena como valor numérico.
- INSTR: Busca la posición numérica de un carácter especificado.
- LPAD: Rellena el valor de caracteres justificado a la derecha.
- RPAD: Rellena el valor de caracteres justificado a la izquierda.
- TRIM: Recorta caracteres iniciales o finales (o ambos) de una cadena de caracteres. (Si trim_character o trim_source es un literal de carácter, debe escribirlo entre comillas simples.)

Función	Resultado
CONCAT('Hello', 'World')	HelloWorld
SUBSTR('HelloWorld',1,5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', 'W')	6
LPAD(salary,10,'*')	*****24000
RPAD(salary, 10, '*')	24000*****
TRIM('H' FROM 'HelloWorld')	elloWorld

El siguiente ejemplo muestra los nombres y apellidos de empleados unidos, la longitud del apellido del empleado y la posición numérica de la letra *a* en el apellido del empleado para todos los empleados que tienen la cadena REP en el identificador de trabajo comenzando por la cuarta posición del identificador de trabajo:



Modificar la sentencia SQL del ejemplo anterior para mostrar los datos para los empleados cuyos apellidos terminan en *n*.

```
SELECT employee_id, CONCAT(first_name, last_name) NAME,
        LENGTH (last_name), INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(last_name, -1, 1) = 'n';
```

FUNCIONES NUMERICAS

Las funciones numéricas aceptan entradas numéricas y devuelven valores numéricos. Esta unidad describe algunas de las funciones numéricas.

- **ROUND:** Redondea el valor a los decimales especificados.

  `ROUND(45.926, 2)` ⟶ `45.93`

- **TRUNC:** Trunca el valor a los decimales especificados.

  `TRUNC(45.926, 2)` ⟶ `45.92`

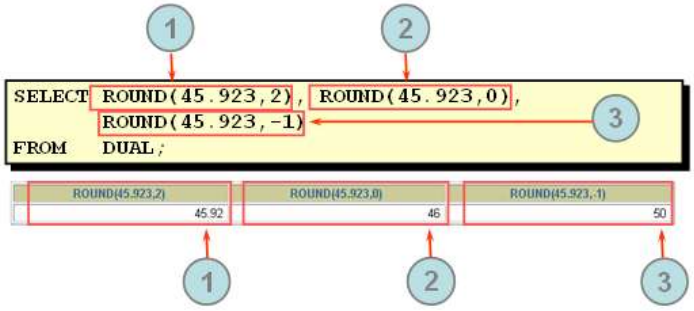
- **MOD:** Devuelve el resto de la división.

  `MOD(1600, 300)` ⟶ `100`

USO DE LA FUNCION ROUND

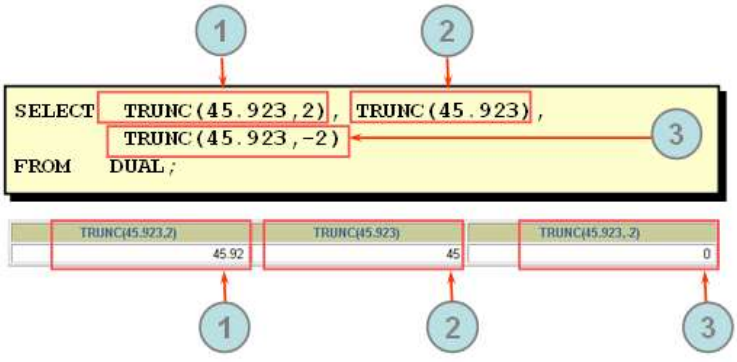
La función ROUND redondea la columna, la expresión o el valor en n posiciones decimales. Si el segundo argumento falta o es 0, el valor se redondea a cero posiciones decimales. Si es 2, el valor se redondea a dos posiciones decimales. A la inversa, si el segundo argumento es -2, el valor se redondea a dos posiciones decimales hacia la izquierda. La función ROUND también se puede utilizar con funciones de fecha.

Tabla DUAL: esta tabla es propiedad del usuario SYS y todos los usuarios pueden acceder a ella. Contiene una columna, DUMMY, y una fila con el valor X. Esta tabla resulta útil si se desea devolver un valor una sola vez, por ejemplo, el valor de una constante, una pseudocolumna o una expresión que no esté derivada de una tabla con datos de usuario. La tabla DUAL se utiliza generalmente para completar la sintaxis de la cláusula SELECT, ya que las cláusulas SELECT y FROM son las dos obligatorias y hay algunos cálculos que no necesitan seleccionar desde tablas reales.



USO DE LA FUNCION TRUNC

La función TRUNC trunca la columna, la expresión o el valor en *n* posiciones decimales. Esta función trabaja con argumentos similares a los de la función ROUND. Si el segundo argumento falta o es 0, el valor se trunca a cero posiciones decimales. Si es 2, el valor se trunca a dos posiciones decimales. A la inversa, si el segundo argumento es -2, el valor se trunca a dos posiciones decimales a la izquierda. Al igual que la función ROUND, la función TRUNC se puede utilizar con funciones de fecha.



USO DE LA FUNCION MOD

La función MOD busca el resto de valor1 dividido por valor2. El ejemplo siguiente calcula el resto del salario después de dividirlo por 5.000 para todos los empleados cuyos identificadores de cargo son SA_REP. Esta función MOD se utiliza a menudo para determinar si un valor es par o impar.

```
SELECT last_name, salary, MOD(salary, 5000)
FROM   employees
WHERE  job_id = 'SA_REP';
```

LAST_NAME	SALARY	MOD(SALARY,5000)
Abel	11000	1000
Taylor	8600	3600
Grant	7000	2000

TRATAMIENTO DE LAS FECHAS

La base de datos Oracle almacena fechas en un formato numérico interno, representando el siglo, el año, el mes, el día, las horas, los minutos y los segundos.

El formato de entrada y visualización por defecto de cualquier fecha es DD-MON-RR. Las fechas válidas para Oracle están comprendidas entre el 1 de enero de 4712 a.C. y el 31 de diciembre de 9999 d.C.

En el ejemplo siguiente el HIRE_DATE del empleado Gietz se muestra en el formato por defecto DD-MON-RR. Sin embargo, las fechas no se almacenan en la base de datos con este formato. Se almacenan todos los componentes de fecha y hora. Por ello, aunque una HIRE_DATE como 07-JUN-94 se muestre como día, mes y año, también hay una información de *hora* y *siglo* asociada a ella. Los datos completos serían 7 de junio de 1994 5:10:43 p.m.

```
SELECT last_name, hire_date
FROM   employees
WHERE  last_name like 'G%';
```

LAST_NAME	HIRE_DATE
Gietz	07-JUN-94
Grant	24-MAY-99

Los datos se almacenan internamente como se indica a continuación:

CENTURY	YEAR	MONTH	DAY	HOUR	MINUTE	SECOND
1994	06	07		5	10	43

Siglos y el Año 2000: Oracle Server cumple con el año 2000. Al insertar en una tabla un registro con una columna de fecha, la información de *siglo* se recoge de la función SYSDATE. Sin embargo, cuando se muestra la columna de fecha en pantalla, el componente siglo no aparece por defecto.

El tipo de dato DATE siempre almacena la información de año internamente como número de cuatro dígitos: dos dígitos para el siglo y otros dos para el año. Por ejemplo, la base de datos Oracle almacena el año como 1996 ó 2001 y no simplemente como 96 ó 01.

LA FUNCION SYSDATE

SYSDATE es una función de fecha que devuelve la fecha y la hora actuales del servidor de base de datos. Puede utilizar SYSDATE de la misma forma que utilizaría cualquier otro nombre de columna. Por ejemplo, puede mostrar la fecha actual seleccionando SYSDATE desde una tabla. Lo habitual es seleccionar SYSDATE desde una tabla ficticia llamada DUAL.

Ejemplo: visualizar la fecha actual utilizando la tabla DUAL.

```
SELECT SYSDATE
FROM   DUAL
```

ARITMETICA CON FECHAS

Como la base de datos almacena las fechas como números, puede realizar cálculos utilizando operadores aritméticos como la suma y la resta. Puede sumar y restar constantes numéricas así como fechas.

El siguiente ejemplo muestra el apellido y el número de semanas de empleo de todos los trabajadores del departamento 90. Resta la fecha en la que se contrató al empleado de la fecha actual (SYSDATE) y divide el resultado por 7 para calcular el número de semanas que lleva empleado un trabajador.

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS
FROM   employees
WHERE  department_id = 90;
```

LAST_NAME	WEEKS
King	744.245395
Kochhar	626.102538
De Haan	453.245395

SYSDATE es una función SQL que devuelve la fecha y la hora actuales. Si a una fecha se le resta otra más reciente, la diferencia es un número negativo.

FUNCIONES DE FECHA

Las funciones de fecha operan sobre fechas de Oracle. Todas las funciones de fecha devuelven un valor del tipo de dato DATE excepto MONTHS_BETWEEN, que devuelve un valor numérico.

- MONTHS_BETWEEN(*date1, date2*): Busca el número de meses entre *date1* y *date2*. El resultado puede ser positivo o negativo. Si *date1* es posterior a *date2*, el resultado es positivo; si *date1* es anterior a *date2*, el resultado es negativo. La parte no entera del resultado representa una porción del mes.
- ADD_MONTHS(*date, n*): Suma *n* meses de calendario a *date*. El valor de *n* debe ser entero y puede ser negativo.
- NEXT_DAY(*date,* '*char*'): Busca la fecha del siguiente día de la semana especificado ('*char*') posterior a *date*. El valor de *char* puede ser un número que represente un día o una cadena de caracteres.
- LAST_DAY(*date*): Busca la fecha del último día del mes en el que está *date*.
- ROUND(*date*[,'*fmt*']): Devuelve *date* redondeado a la unidad especificada por el modelo de formato *fmt.* Si el modelo de formato *fmt* está omitido, *date* se redondea al día más próximo.
- TRUNC(*date*[, '*fmt*']): Devuelve *date* con la parte de hora del día truncada a la unidad especificada por el modelo de formato *fmt.* Si el modelo de formato *fmt* está omitido, *date* se trunca al día más próximo.

Esta lista es un subjuego de las funciones de fecha disponibles. Los modelos de formato se tratan más adelante en esta lección. Ejemplos de modelos de formato son mes y año.

Función	Descripción
MONTHS_BETWEEN	Número de meses entre dos fechas
ADD_MONTHS	Suma meses de calendario a una fecha
NEXT_DAY	Siguiente día de la fecha especificada
LAST_DAY	Último día del mes
ROUND	Redondea la fecha
TRUNC	Trunca la fecha

Por ejemplo, muestre el número de empleado, la fecha de contratación, el número de meses empleado, la fecha de revisión de seis meses, el primer viernes después de la fecha de contratación y el último día del mes de contratación para todos los trabajadores que lleven empleados menos de 36 meses.

```
SELECT employee_id, hire_date, MONTHS_BETWEEN (SYSDATE, hire_date) TENURE,
       ADD_MONTHS (hire_date, 6) REVIEW, NEXT_DAY (hire_date, 'FRIDAY'), LAST_DAY(hire_date)
FROM   employees
WHERE  MONTHS_BETWEEN (SYSDATE, hire_date) < 36;
```

USO DE FUNCIONES DE FECHA

- `MONTHS_BETWEEN ('01-SEP-95','11-JAN-94')` → `19.6774194`
- `ADD_MONTHS ('11-JAN-94',6)` → `'11-JUL-94'`
- `NEXT_DAY ('01-SEP-95','FRIDAY')` → `'08-SEP-95'`
- `LAST_DAY('01-FEB-95')` → `'28-FEB-95'`

Las funciones ROUND y TRUNC se pueden utilizar para valores numéricos y de fecha. Cuando se utilizan con fechas, estas funciones redondean o truncan al modelo de formato especificado. Por lo tanto, se puede redondear fechas al año o mes más próximo.

Ejemplo

Compare las fechas de contratación para todos los empleados que comenzaron en 1997. Visualice el número de empleado, la fecha de contratación y el mes de inicio utilizando las funciones ROUND y TRUNC.

```
SELECT employee_id, hire_date,
    ROUND(hire_date, 'MONTH'), TRUNC(hire_date, 'MONTH')
FROM   employees
WHERE  hire_date LIKE '%97';
```

Asuma `SYSDATE = '25-JUL-95'`:

- `ROUND(SYSDATE,'MONTH')` → `01-AUG-95`
- `ROUND(SYSDATE ,'YEAR')` → `01-JAN-96`
- `TRUNC(SYSDATE ,'MONTH')` → `01-JUL-95`
- `TRUNC(SYSDATE ,'YEAR')` → `01-JAN-95`

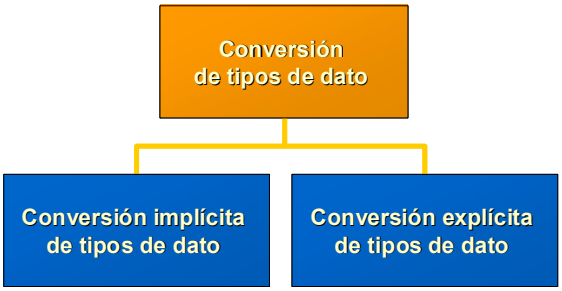
FUNCIONES DE CONVERSIÓN

Además de los tipos de dato Oracle, las columnas de las tablas de las bases de datos Oracle9*i* se pueden definir utilizando tipos de dato ANSI, DB2 y SQL/DS. Sin embargo, Oracle Server convierte internamente estos tipos de dato en tipos de dato Oracle.

En algunos casos, Oracle Server utiliza datos de un tipo donde espera datos de otro tipo distinto. Cuando esto ocurre, Oracle Server puede convertir automáticamente los datos al tipo de dato esperado. Esta conversión la puede hacer Oracle Server *implícitamente* o el usuario *explícitamente*.

Las conversiones explícitas de tipos de dato se realizan utilizando las funciones de conversión, que convierten un valor de un tipo de dato en otro. Generalmente, la forma de los nombres de función sigue la convención *data type* TO *data type*. El primer tipo de dato es el de entrada y el último, el de salida.

Nota: Aunque la conversión implícita de tipos de dato está disponible, se recomienda que haga conversiones explícitas de tipos de dato para asegurar la fiabilidad de las sentencias SQL.



CONVERSIÓN IMPLÍCITA DE TIPOS DE DATO

La asignación es correcta si Oracle Server puede convertir el tipo de dato del valor utilizado en la asignación al del destino de la asignación.

Para las asignaciones, Oracle Server puede convertir automáticamente lo siguiente:

De	A
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

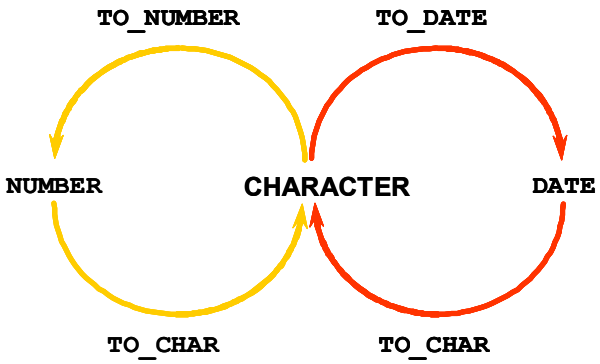
En general, Oracle Server utiliza la regla para expresiones cuando es necesaria una conversión de tipos de dato en lugares no cubiertos por una regla para conversiones de asignación.

Las conversiones CHAR a NUMBER sólo son correctas si la cadena de caracteres representa un número válido. Para la evaluación de la expresión, Oracle Server puede convertir automáticamente lo siguiente:

De	A
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE

CONVERSIÓN EXPLÍCITA DE TIPOS DE DATO

SQL proporciona tres funciones para convertir un valor de un tipo de dato en otro:



VISUALIZACIÓN DE UNA FECHA EN UN FORMATO ESPECÍFICO

Anteriormente, todos los valores de fecha de Oracle se mostraban en formato DD-MON-YY. Puede utilizar la función TO_CHAR para convertir una fecha de este formato por defecto al que especifique.

Instrucciones

- El modelo de formato se debe escribir entre comillas sencillas y es sensible a mayúsculas/minúsculas.
- El modelo de formato puede incluir cualquier elemento de formato de fecha válido. Asegúrese de separar el valor de fecha del modelo de formato mediante una coma.
- Los nombres de los días y los meses en la salida se rellenan automáticamente con espacios en blanco.
- Para eliminar los espacios rellenados o suprimir los ceros a la izquierda, utilice el elemento *fm* de modo de relleno.
- Puede formatear el campo de caracteres resultante con el comando COLUMN de *i*SQL*Plus que se cubre en una lección posterior.

```
SELECT employee_id, TO_CHAR(hire_date, 'MM/YY') Month_Hired
FROM   employees
WHERE  last_name = 'Higgins';
```

```
TO_CHAR(date, 'format_model')
```

ELEMENTOS DEL MODELO DE FORMATO DE FECHA

YYYY	Año completo en números
YEAR	Años en letra
MM	Valor de dos dígitos para el mes
MONTH	Nombre completo del mes
MON	Abreviatura de tres letras del mes
DY	Abreviatura de tres letras del día de la semana
DAY	Nombre completo del día de la semana
DD	Día del mes en número

ELEMENTOS DE FORMATO DE FECHA: FORMATOS DE HORA

Utilice los formatos listados en las siguientes tablas para mostrar información de hora y literales y para cambiar numerales a números en letra.

- Los elementos de hora formatean la porción de hora de la fecha.

HH24:MI:SS AM	15:45:32 PM

- Agregue cadenas de caracteres escribiéndolas entre comillas dobles.

DD "of" MONTH	12 of OCTOBER

- Los sufijos numéricos escriben los números en letra.

ddspth	fourteenth

LA FUNCIÓN TO_CHAR CON FECHAS

La siguiente sentencia SQL muestra los apellidos y las fechas de contratación de todos los empleados. La fecha de contratación aparece como 17 June 1987.

```
SELECT last_name,
       TO_CHAR(hire_date, 'fmDD Month YYYY')
       AS HIREDATE
FROM   employees;
```

LAST_NAME	HIREDATE
King	17 June 1987
Kochhar	21 September 1989
De Haan	13 January 1993
Hunold	3 January 1990
Ernst	21 May 1991
Lorentz	7 February 1999
Mourgos	16 November 1999

...

20 rows selected.

Modificar el ejemplo anterior para mostrar las fechas en un formato que aparezca como Seventh of June 1994 12:00:00 AM.

```
SELECT  last_name, TO_CHAR(hire_date,
            'fmDdspth "of" Month YYYY fmHH:MI:SS AM') HIREDATE
FROM    employees;
```

LA FUNCIÓN TO_CHAR CON NÚMEROS

Al trabajar con valores numéricos como, por ejemplo, cadenas de caracteres, debe convertir estos números al tipo de dato de caracteres mediante la función TO_CHAR, que traduce un valor del tipo de dato NUMBER al tipo de dato VARCHAR2. Esta técnica resulta especialmente útil con la concatenación.

```
TO_CHAR(number, 'format_model')
```

Estos son algunos de los elementos de formato que puede utilizar con la función TO_CHAR para mostrar un valor numérico como carácter:

Elemento	Descripción	Ejemplo	Resultado
9	Posición numérica (el número de nueves determina el ancho de la visualización).	999999	1234
0	Muestra ceros a la izquierda.	999999	001234
$	Signo de dólar flotante	$999999	$1234
L	Símbolo de divisa local flotante	L999999	FF1234
.	Coma decimal en la posición especificada.	999999.99	1234.00
,	Coma en la posición especificada	999,999	1,234
MI	Signos menos a la derecha (valores negativos)	999999MI	1234-
PR	Números negativos entre paréntesis.	999999PR	<1234>
EEEE	Notación científica (el formato debe especificarse con cuatro **E**.	99.999EEEE	1234E+03
V	Multiplicar por 10 **n** veces (**n** = número de nueves después de **V**.	9999V99	123400
B	Muestra valores cero como espacios en blanco y no como 0 (cero).	B9999.99	1234.00

- Oracle Server muestra una cadena de signos numeral (#) en lugar de un número entero cuyos dígitos excedan el número de dígitos que se proporciona en el modelo de formato.
- Oracle Server redondea el valor decimal almacenado al número de espacios decimales proporcionados en el modelo de formato.

El siguiente ejemplo muestra el uso de la función CHAR con números.

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY
FROM   employees
WHERE  last_name = 'Ernst';
```

SALARY
$6,000.00

FUNCIONES TO_NUMBER Y TO_DATE

Se puede convertir una cadena de caracteres en un número o en una fecha. Para ello, utilice las funciones TO_NUMBER o TO_DATE. El modelo de formato que elija se basa en los elementos de formato mostrados previamente.

El modificador "x" especifica una coincidencia exacta para el argumento de caracteres y el modelo de formato de fecha de una función TO_DATE:

- La puntuación y el texto entrecomillado del argumento de caracteres deben coincidir exactamente (excepto en las mayúsculas/minúsculas) con las partes correspondientes del modelo de formato.
- El argumento de caracteres no puede tener espacios en blanco adicionales. Sin fx, Oracle ignora los espacios en blanco adicionales.
- Los datos numéricos del argumento de caracteres deben tener el mismo número de dígitos que el elemento correspondiente del modelo de formato. Sin fx, los números del argumento de caracteres pueden omitir los ceros a la izquierda.

USO DE LAS FUNCIONES TO_NUMBER Y TO_DATE

- Convierta una cadena de caracteres en formato numérico utilizando la función TO_NUMBER:

```
TO_NUMBER(char[, 'format_model'])
```

- Convierta una cadena de caracteres en formato de fecha utilizando la función TO_DATE:

```
TO_DATE(char[, 'format_model'])
```

EL ELEMENTO DE FORMATO DE FECHA RR

El formato de fecha RR es similar al elemento YY, pero se puede utilizar para especificar siglos distintos. Puede utilizar el elemento de formato de fecha RR en lugar del YY, de forma que el siglo del valor de retorno varía en función del año especificado de dos dígitos y los dos últimos dígitos del año actual. La tabla de la transparencia resume el comportamiento del elemento RR.

Año Actual	Fecha Especificada	Formato RR	Formato YY
1995	27-OCT-95	1995	1995
1995	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2001	27-OCT-95	1995	2095

		Si el año especificado de dos dígitos es:	
		0–49	50–99
Si los dos dígitos del año actual son:	0–49	La fecha que se devuelve está en el siglo actual.	La fecha que se devuelve está en el siglo anterior al actual.
	50–99	La fecha que se devuelve está en el siglo siguiente al actual.	La fecha que se devuelve está en el siglo actual.

En el siguiente ejemplo se buscan empleados contratados antes de 1990, se puede utilizar el formato RR. Como el año es posterior a 1999, el formato RR interpreta la parte de año de la fecha desde 1950 a 1999.

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM  employees
WHERE hire_date < TO_DATE('01-Jan-90', 'DD-Mon-RR');
```

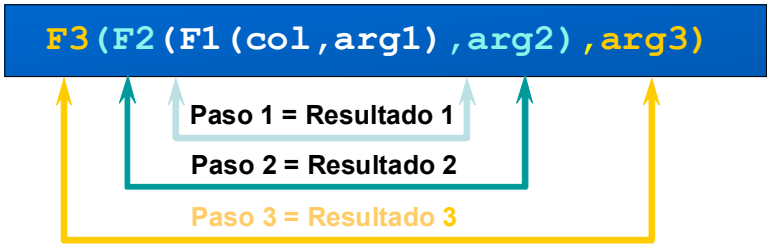
LAST_NAME	TO_CHAR(HIR
King	17-Jun-1987
Kochhar	21-Sep-1989
Whalen	17-Sep-1987

Por otra parte, con el siguiente comando no se selecciona ninguna fila porque el formato YY interpreta la parte de año de la fecha en el siglo actual (2090).

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-yyyy')
FROM   employees
WHERE  TO_DATE(hire_date, 'DD-Mon-yy') < '01-Jan-1990';
```

FUNCIONES DE ANIDAMIENTO

Las funciones de una sola fila se pueden anidar a cualquier profundidad. Las funciones anidadas se evalúan desde el nivel más interno al más externo. A continuación, se indican algunos ejemplos que muestran la flexibilidad de estas funciones.



El ejemplo de la transparencia muestra la cabeza de la compañía, que no tiene director. La evaluación de la sentencia SQL implica dos pasos:

1. Evaluar la función interna que convierte un valor numérico en una cadena de caracteres.
   - Result1 = TO_CHAR(manager_id)
2. Evaluar la función externa que sustituye el valor nulo por una cadena de texto.
   - NVL(Result1, 'No Manager')

La expresión completa se convierte en la cabecera de columna al no haberse proporcionado alias de columna.

Ejemplo

Visualizar la fecha del primer viernes transcurridos seis meses desde la fecha de contratación. La fecha resultante debe aparecer como Friday, August 13th, 1999. Ordene los resultados por fecha de contratación.

```
SELECT   TO_CHAR(NEXT_DAY(ADD_MONTHS (hire_date, 6), 'FRIDAY'),
         'fmDay, Month DDth, YYYY') "Next 6 Month Review"
FROM     employees
ORDER BY hire_date;
```

```
SELECT last_name,
       NVL(TO_CHAR(manager_id), 'No Manager')
FROM   employees
WHERE  manager_id IS NULL;
```

LAST_NAME	NVL(TO_CHAR(MANAGER_ID),'NOMANAGER')
King	No Manager

FUNCIONES GENERALES

Estas funciones trabajan con cualquier tipo de dato y están relacionadas con el uso de valores nulos en la lista de expresiones.

- `NVL (expr1, expr2)`
- `NVL2 (expr1, expr2, expr3)`
- `NULLIF (expr1, expr2)`
- `COALESCE (expr1, expr2, ..., exprn)`

FUNCIÓN NVL

Para convertir un valor nulo en uno real, utilice la función NVL. Se debe tener en cuenta lo siguiente:

- Los tipos de dato que se pueden utilizar son fechas, caracteres y numéricos.

- Los tipos de dato deben coincidir:
  - `NVL(commission_pct,0)`
  - `NVL(hire_date,'01-JAN-97')`
  - `NVL(job_id,'No Job Yet')`

Sintaxis

NVL (*expr1*, *expr2*)

En la sintaxis:

expr1 es el valor o la expresión de origen que puede contener un valor nulo.

expr2 es el valor de destino para convertir el valor nulo.

Puede utilizar la función NVL para convertir cualquier tipo de dato, pero el valor de retorno siempre es del mismo tipo que *expr1*.

Tipo de dato	Ejemplo de conversión
NUMBER	NVL(number_column, 9)
DATE	NVL(date_column, '01-JAN-95')
CHAR or VARCHAR2	NVL(character_column, 'Unavailable')

En el siguiente ejemplo se calcula la compensación anual de todos los empleados, se debe multiplicar el salario mensual por 12 y sumarle el porcentaje de comisión.

```
SELECT last_name, salary, NVL(commission_pct, 0),
   (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL
FROM employees;
```

LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
King	24000	0	288000
Kochhar	17000	0	204000
De Haan	17000	0	204000
Hunold	9000	0	108000
Ernst	6000	0	72000
Lorentz	4200	0	50400
Mourgos	5800	0	69600
Rajs	3500	0	42000

...

20 rows selected.

FUNCIÓN NVL2

La función NVL2 examina la primera expresión. Si no es nula, devuelve la segunda expresión. Si es nula, devuelve la tercera.

Sintaxis

NVL(*expr1*, *expr2*, *expr3*)

En la sintaxis:

expr1 es el valor o la expresión de origen que puede contener un valor nulo.

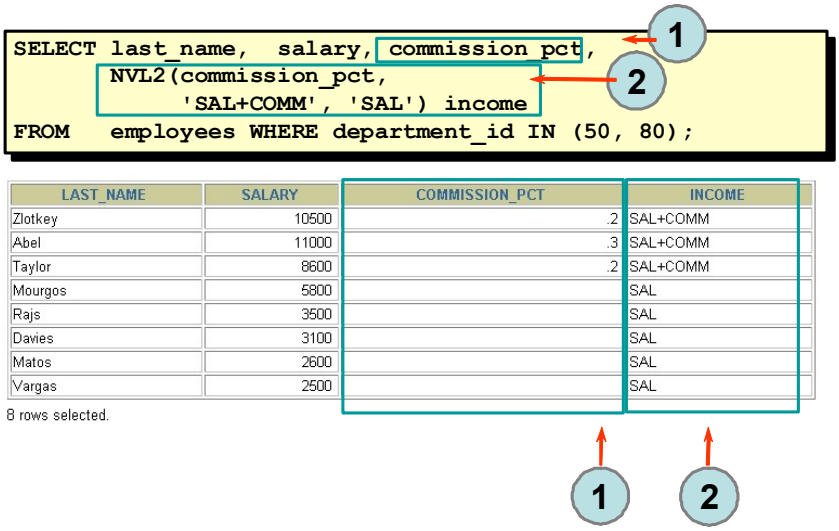
expr2 es el valor devuelto si *expr1* no es nulo.

expr3 es el valor devuelto si *expr1* es nulo.

En el ejemplo siguiente, se examina la columna COMMISSION_PCT. Si se detecta un valor, se devuelve la segunda expresión, SAL+COMM. Si la columna COMMISSION_PCT contiene un valor nulo, se devuelve la tercera expresión, SAL.

El argumento *expr1* puede tener cualquier tipo de dato. Los argumentos *expr2* y *expr3* pueden tener cualquier tipo de dato excepto LONG. Si los tipos de dato de *expr2* y *expr3* son diferentes, el servidor Oracle convierte *expr3* en el tipo de dato de *expr2* antes de compararlos a menos que *expr3* sea una constante nula, en cuyo caso no es necesaria una conversión del tipo de dato.

El tipo de dato del valor de retorno es siempre el mismo que el de *expr2*, a menos que *expr2* sea un dato de caracteres, en cuyo caso el tipo de dato del valor de retorno es VARCHAR2.



LA FUNCIÓN NULLIF

La función NULLIF compara dos expresiones: si son iguales, la función devuelve nulo; si no lo son devuelve la primera expresión. No se puede especificar el literal NULL para la primera expresión.

Sintaxis

NULLIF (*expr1*, *expr2*)

En la sintaxis:

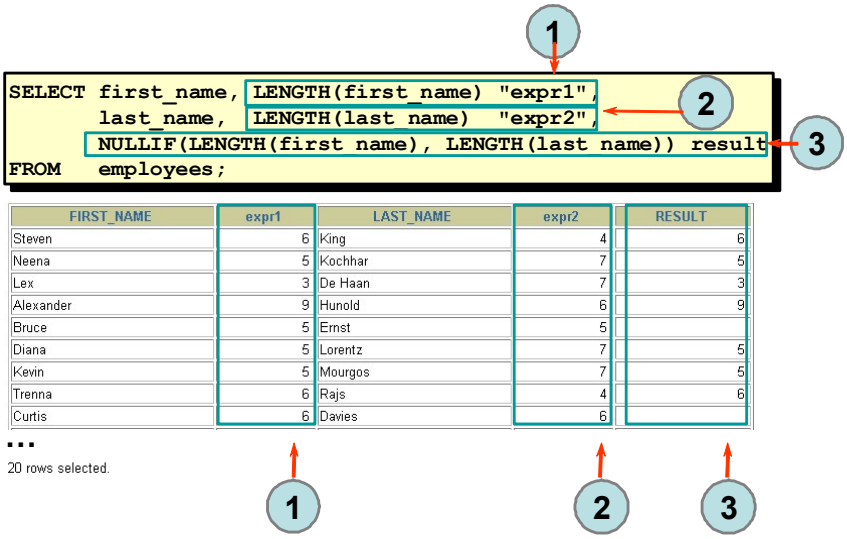
expr1 es el valor de origen que se compara con *expr2.*

expr2 es el valor de origen que se compara con *expr1* (si no es igual a *expr1*, se devuelve *expr1*).

En el ejemplo mostrado, se compara el identificador de cargo de la tabla EMPLOYEES con el de la tabla JOB_HISTORY para cualquier empleado que esté en ambas tablas. La salida muestra el cargo actual de cada empleado. Si el empleado aparece más de una vez, significa que ha tenido anteriormente al menos dos cargos.

Nota: La función NULLIF es el equivalente lógico de la siguiente expresión CASE, que se trata más adelante:

CASE WHEN expr1 = expr 2 THEN NULL ELSE expr1 END



LA FUNCIÓN COALESCE

La función COALESCE devuelve la primera expresión no nula de la lista.

Sintaxis

COALESCE (*expr1*, *expr2*, ... *exprn*)

En la sintaxis:

expr1 devuelve esta expresión si no es nula.

expr2 devuelve esta expresión si la primera es nula y ésta no lo es.

exprn devuelve esta expresión si las expresiones precedentes son nulas.

La ventaja de la función COALESCE sobre la función NVL es que puede tomar varios valores alternativos. Si la primera expresión no es nula, devuelve dicha expresión; en caso contrario, realiza una fusión (COALESCE) de las expresiones restantes.

En el siguiente ejemplo, si el valor de COMMISSION_PCT no es nulo, se muestra. Si es nulo, se muestra SALARY. Si ambos valores son nulos, se muestra el valor 10.

```
SELECT   last_name,
         COALESCE(commission_pct, salary, 10) comm
FROM     employees
ORDER BY commission_pct;
```

LAST_NAME	COMM
Grant	.15
Zlotkey	.2
Taylor	.2
Abel	.3
King	24000
Kochhar	17000
De Haan	17000
Hunold	9000

...

20 rows selected.

EXPRESIONES CONDICIONALES

Los dos métodos utilizados para implementar el procesamiento condicional (lógica IF-THEN-ELSE) dentro de una sentencia SQL son la expresión CASE y la función DECODE.

Nota: La expresión CASE es nueva en la versión Oracle9*i* Server y cumple con ANSI SQL; DECODE es específica de la sintaxis de Oracle.

- Proporcionan el uso de la lógica IF-THEN-ELSE dentro de una sentencia SQL.
- Utilizan dos métodos:
  - Expresión CASE
  - Función DECODE

LA EXPRESIÓN CASE

Las expresiones CASE le permiten utilizar la lógica IF-THEN-ELSE en sentencias SQL sin tener que llamar a procedimientos.

En una expresión CASE sencilla, Oracle busca el primer par WHEN ... THEN para el que expr es igual a comparison_expr y devuelve return_expr. Si ninguno de los pares WHEN ... THEN cumplen esta condición y existe una cláusula ELSE, Oracle devuelve else_expr. En caso contrario, Oracle devuelve un valor nulo. No puede especificar el literal NULL para todas las expresiones return_expr ni para else_expr.

Todas las expresiones (expr, comparison_expr y return_expr) deben tener el mismo tipo de dato, que puede ser CHAR, VARCHAR2, NCHAR o NVARCHAR2.

Facilita las consultas condicionales realizando el trabajo de una sentencia IF-THEN-ELSE:

```
CASE expr WHEN comparison_expr1 THEN return_expr1
          [WHEN comparison_expr2 THEN return_expr2
           WHEN comparison_exprn THEN return_exprn
           ELSE else_expr]
END
```

USO DE LA EXPRESIÓN CASE

En la sentencia SQL precedente, se descodifica el valor de JOB_ID. Si JOB_ID es IT_PROG, el incremento salarial es del 10 %; si es ST_CLERK, el incremento es del 15 %; si es SA_REP, el incremento es del 20 %. Para todos los demás cargos, no hay incremento salarial.

La misma sentencia se puede escribir con la función DECODE.

```
SELECT last_name, job_id, salary,
       CASE job_id WHEN 'IT_PROG'  THEN  1.10*salary
                   WHEN 'ST_CLERK' THEN  1.15*salary
                   WHEN 'SA_REP'   THEN  1.20*salary
       ELSE       salary END     "REVISED_SALARY"
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

LA FUNCIÓN DECODE

Esta instrucción facilita las consultas condicionales realizando el trabajo de una sentencia IF-THEN-ELSE:

La función DECODE descodifica una expresión de forma similar a la lógica IF-THEN-ELSE utilizada en varios idiomas. Esta función descodifica *expression* después de compararla con cada valor *search*. Si la expresión es la misma que *search*, se devuelve *result*.

Si el valor por defecto está omitido, se devuelve un valor nulo cuando un valor de búsqueda no coincide con ninguno de los valores resultantes.

```
DECODE(col|expression, search1, result1
                      [, search2, result2,...,]
                      [, default])
```

USO DE LA FUNCIÓN DECODE

En el siguiente ejemplo, se prueba el valor de JOB_ID. Si JOB_ID es IT_PROG, el incremento salarial es del 10 %; si es ST_CLERK, el incremento es del 15 %; si es SA_REP, el incremento es del 20 %. Para todos los demás cargos, no hay incremento salarial.

```
SELECT last_name, job_id, salary,
       DECODE(job_id, 'IT_PROG',  1.10*salary,
                      'ST_CLERK', 1.15*salary,
                      'SA_REP',   1.20*salary,
              salary)
       REVISED_SALARY
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

La misma sentencia se puede expresar en pseudocódigo como sentencia IF-THEN-ELSE:

IF job_id = 'IT_PROG' THEN salary = salary*1.10

IF job_id = 'ST_CLERK' THEN salary = salary*1.15

IF job_id = 'SA_REP' THEN salary = salary*1.20

ELSE salary = salary

En el siguiente ejemplo, se determina el tipo impositivo de cada empleado del departamento 80 en función del salario mensual. Los tipos impositivos son según los valores mencionados a continuación.

Rango de Salario Mensual	***Tipo***
$0,00 - 1999,99	00%
$2.000,00 - 3.999,99	09%
$4.000,00 - 5.999,99	20%
$6.000,00 - 7.999,99	30%
$8.000,00 - 9.999,99	40%
$10.000,00 - 11.999,99	42%
$12.200,00 - 13.999,99	44%
$14.000,00 o superior	45 %

```
SELECT last_name, salary,
       DECODE (TRUNC(salary/2000, 0),
                         0, 0.00,
                         1, 0.09,
                         2, 0.20,
                         3, 0.30,
                         4, 0.40,
                         5, 0.42,
                         6, 0.44,
                            0.45) TAX_RATE
FROM   employees
WHERE  department_id = 80;
```