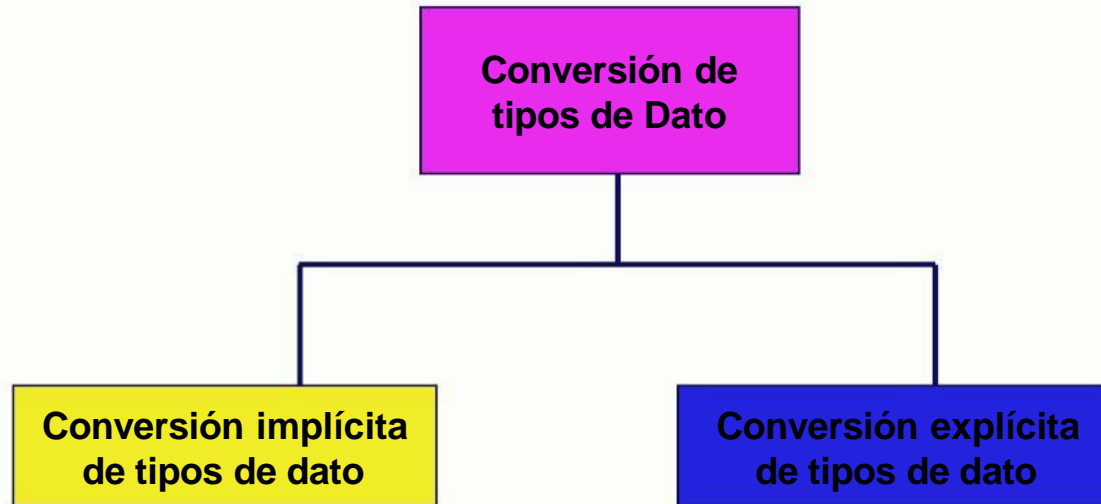


# 4

## Uso de Funciones de Conversiones y expresiones condicionales

**ORACLE®**  
DATABASE **11g**

# Funciones de Conversión



# Conversión Implícita de Tipos de Dato

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

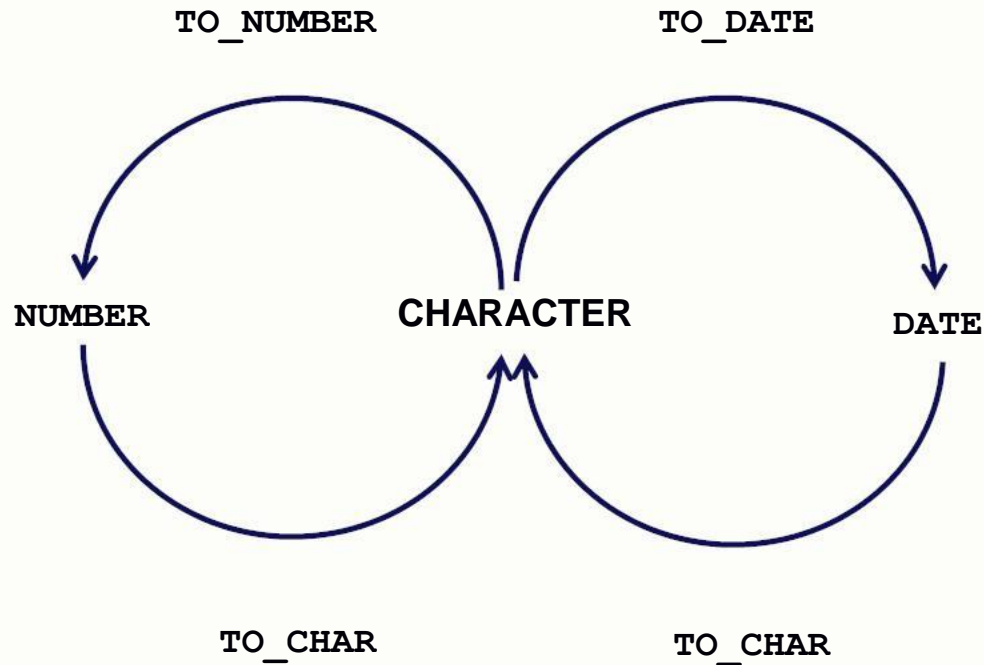
De	A
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
NUMBER	VARCNER2
DATE	VARCHAR2

# Conversión Implícita de Tipos de Dato

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 Datos



# Uso de la función TO\_CHAR con fechas

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

## El modelo de formato:

- Se debe escribir entre comillas simples y es sensible a mayúsculas/minúsculas.
- Puede incluir cualquier elemento de formato de fecha válido.
- Tiene un elemento `fm` para eliminar espacios rellenos o suprimir ceros a la izquierda.
- Se separa del valor de fecha con una coma.

# Uso de la función TO\_CHAR con fechas

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

EMPLOYEE_ID	MONTH_HIRED
205	06/94

# Elementos del Modelo de Formato de Fecha

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



# Elementos del Modelo de Formato de Fecha

- 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 número en letra.

Ddspth	fourteenth
--------	------------

# Uso de la Función TO\_CHAR con Fechas

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

LAST_NAME	HIREDATE
King	17 Junio 1987
Kochhar	21 Septiembre 1989
De Haan	13 Enero 1993
Hunold	3 Enero 1990
Ernst	21 Mayo 1991
Austin	25 Junio 1997
Pataballa	5 Febrero 1998
Lorentz	7 Febrero 1999
Greenberg	17 Agosto 1994
Faviet	16 Agosto 1994
Chen	28 Septiembre 1997
Sciarra	30 Septiembre 1997
Urman	7 Marzo 1998
Popp	7 Diciembre 1999
Raphaely	7 Diciembre 1994

# Uso de la Función TO\_CHAR con Fechas

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

LAST_NAME	HIREDATE
King	Seventeenth of Junio 1987 12:00:00 AM
Kochhar	Twenty-First of Septiembre 1989 12:00:00 AM
De Haan	Thirteenth of Enero 1993 12:00:00 AM
Hunold	Third of Enero 1990 12:00:00 AM
Ernst	Twenty-First of Mayo 1991 12:00:00 AM
Austin	Twenty-Fifth of Junio 1997 12:00:00 AM
Pataballa	Fifth of Febrero 1998 12:00:00 AM
Lorentz	Seventh of Febrero 1999 12:00:00 AM
Greenberg	Seventeenth of Agosto 1994 12:00:00 AM
Faviet	Sixteenth of Agosto 1994 12:00:00 AM
Chen	Twenty-Eighth of Septiembre 1997 12:00:00 AM
Sciarra	Thirtieth of Septiembre 1997 12:00:00 AM
Urman	Seventh of Marzo 1998 12:00:00 AM

# Uso de la Función TO\_CHAR con números

```
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:

9	Representa un número.
0	Obliga a mostrar un cero.
\$	Coloca un signo de dólar flotante.
L	Utiliza el símbolo de divisa local flotante.
.	Imprime una coma decimal.
,	Imprime un indicador de miles.

# Uso de la Función TO\_CHAR con números

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

SALARY
\$6,000.00

# 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'])
```

- Estas funciones tienen un modificador *fx* que especifica la coincidencia exacta para el argumento de caracteres y un modelo de formato de fecha de una función TO\_DATE.

# Ejemplo de Formato de Fecha RR

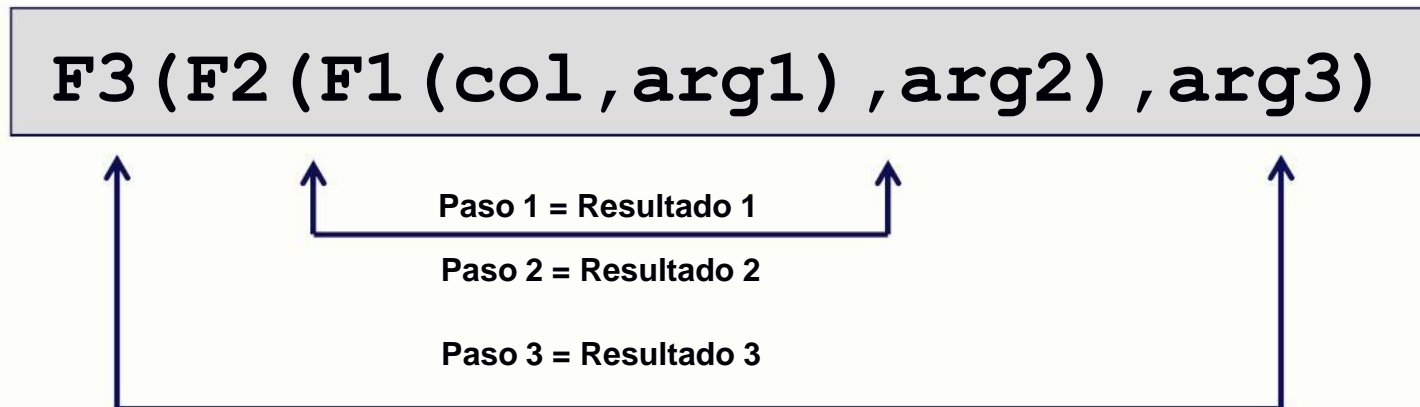
Para buscar empleados contratados antes de 1990, utilice el formato RR, que produce los mismos resultados tanto si se ejecuta el comando en 1999 o ahora:

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM employees
WHERE hire_date < TO_DATE('01-Ene-09', 'DD-Mon-RR');
```

LAST_NAME	TO_CHAR(HIRE_DATE,'DD-MON-YYYY')
King	17-jun-1987
Kochhar	21-sep-1989
Hunold	03-ene-1990
Whalen	17-sep-1987

# Anidando funciones

- Las funciones de una sola fila se pueden anidar a cualquier nivel.
- Las funciones anidadas se evalúan desde el nivel más profundo al menos profundo.





# Anidando funciones

```
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.

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

# Función NVL

**Convierte un valor nulo en un valor real.**

- **Los tipos de datos 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')`

# Uso de la Función NVL

```
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
Greenberg	12000	0	144000
Faviet	9000	0	108000
Chen	8200	0	98400
Raphaely	11000	0	132000
Fripp	8200	0	98400
Russell	14000	,4	235200
Partners	13500	,3	210600
Errazuriz	12000	,3	187200
Cambrault	11000	,3	171600

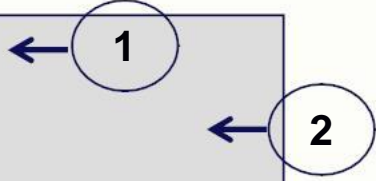
↑  
1

↑  
2


ORACLE

# Uso de la Función NVL

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



LAST_NAME	SALARY	COMMISSION_PCT	AN_SAL
King	24000		
Kochhar	17000		
De Haan	17000		
Hunold	9000		
Greenberg	12000		
Faviet	9000		
Chen	8200		
Raphaely	11000		
Fripp	8200		
Russell	14000	,4	235200
Partners	13500	,3	210600
Errazuriz	12000	,3	187200
Cambrault	11000	,3	171600
Zlotkey	10500	,2	151200



# Uso de la Función NVL2

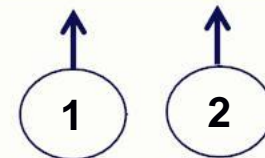
La función NVL2 examina la primera expresión:

- Si la expr1 es nula, entonces retorna la expr3.
- Si la expr1 no es nula, entonces retorna la expr2.

- NVL2 (expr1, expr2, expr3)

```
SELECT last_name, salary, commission_pct, ← 1
       NVL2(commission_pct, ← 2
            'NO NULO', 'NULO') income
FROM   employees
WHERE  department_id IN (50,80);
```

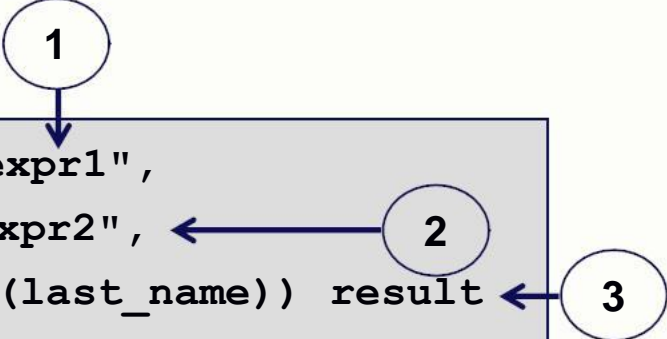
Livingston	8400	,2	NO NULO
Johnson	6200	,1	NO NULO
Taylor	3200		NULO
Fleaur	3100		NULO



# Uso de la Función NULLIF

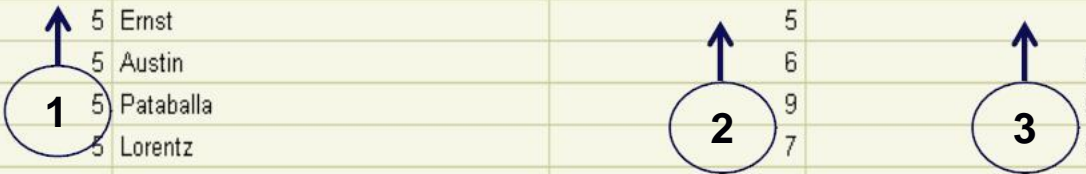
La función NULLIF compara dos expresiones:

- Si es igual la expr1 y expr2, la función retorna nulo.
- Si no son iguales la expr1 y expr2, retorna la expr1.
- Si expr1 es nula, entonces retorna la expr2.
- **NULLIF (expr1, expr2)**



```
SELECT first_name, LENGTH(first_name) "expr1",  
       last_name,   LENGTH(last_name) "expr2",  
       NULLIF(LENGTH(first_name), LENGTH(last_name)) result  
FROM   employees;
```

FIRST_NAME	EXPR1	LAST_NAME	EXPR2	RESULT
Steven	6	King	4	6
Neena	5	Kochhar	7	5
Lex	3	De Haan	7	3
Alexander	9	Hunold	6	9
Bruce	5	Ernst	5	5
David	5	Austin	6	5
Valli	5	Pataballa	9	5
Diana	5	Lorentz	7	5
Nancy	5	Greenberg	9	5



# Uso de la Función COALESCE

- La ventaja de la función COALESCE sobre la función NVL es que puede tomar varios valores alternativos.
- Si la primera expr1 no es nula, devuelve dicha expresión, expr1.
- Si la expr1 es nula, retorna la expr2.
- Si la expr1 y expr2 son nulas, retorna la expr3

**COALESCE (expr1, expr2, expr3)**



# Uso de la Función COALESCE

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

LAST_NAME	COMMISSION_PCT	MANAGER_ID	COMM
Partners	,3	100	,3
Errazuriz	,3	100	,3
Doran	,3	146	,3
Abel	,3	149	,3
Smith	,3	146	,3
Cambrault	,3	100	,3
Tucker	,3	145	,3
King	,35	146	,35
McEwen	,35	146	,35
Sully	,35	146	,35
Russell	,4	100	,4
King			10
Kochhar		100	100
De Haan		100	100

# Expresiones Condicionales

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

# La Expresión CASE

**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 retrun_exprn
            ELSE else_expr]
END
```

# Uso de la Expresión CASE

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

```
SELECT last_name, job_id, salary,  
       CASE job_id WHEN 'IT_PROG'      THEN 1.10 * salary  
                   WHEN 'FI_ACCOUNT'  THEN 1.15 * salary  
                   WHEN 'AD_VP'       THEN 1.20 * salary  
       ELSE salary END    "REVISED SALARY"  
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED SALARY
King	AD_PRES	24000	24000
Kochhar	AD_VP	17000	20400
De Haan	AD_VP	17000	20400
Hunold	IT_PROG	9000	9900
Ernst	IT_PROG	6000	6600
Austin	IT_PROG	4800	5280
Pataballa	IT_PROG	4800	5280
Lorentz	IT_PROG	4200	4620
Greenberg	FI_MGR	12000	12000
Faviet	FI_ACCOUNT	9000	10350
Chen	FI_ACCOUNT	8200	9430
Sciarra	FI_ACCOUNT	7700	8855
Urman	FI_ACCOUNT	7800	8970

# La Función `DECODE`

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

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

# Uso de la Función DECODE

```
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;
```

# Uso de la Función DECODE

**Muestre el tipo impositivo aplicable para cada empleado del departamento 80.**

```
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;
```

# Resumen

**En esta lección, debería haber aprendido a:**

- **Realizar cálculos sobre datos utilizando funciones**
- **Modificar elementos de datos individuales utilizando funciones**
- **Manipular la salida para grupos de filas utilizando funciones.**
- **Alterar formatos de fecha para su visualización utilizando funciones.**
- **Convertir tipos de dato de columna utilizando funciones**
- **Utilizar funciones NVL**
- **Utilizar la lógica IF-THEN-ELSE**



# Práctica 4. Visión General

**Esta práctica cubre los siguientes temas:**

- **Creación de consultas que requieran el uso de funciones numéricas, de caracteres y de fecha**
- **Uso de la concatenación con funciones**
- **Escritura de consultas sensibles a mayúsculas/minúsculas para probar la utilidad de las funciones de caracteres**
- **Realización de cálculos de meses y años de servicio para un empleado**
- **Determinación de la fecha de revisión para un empleado**