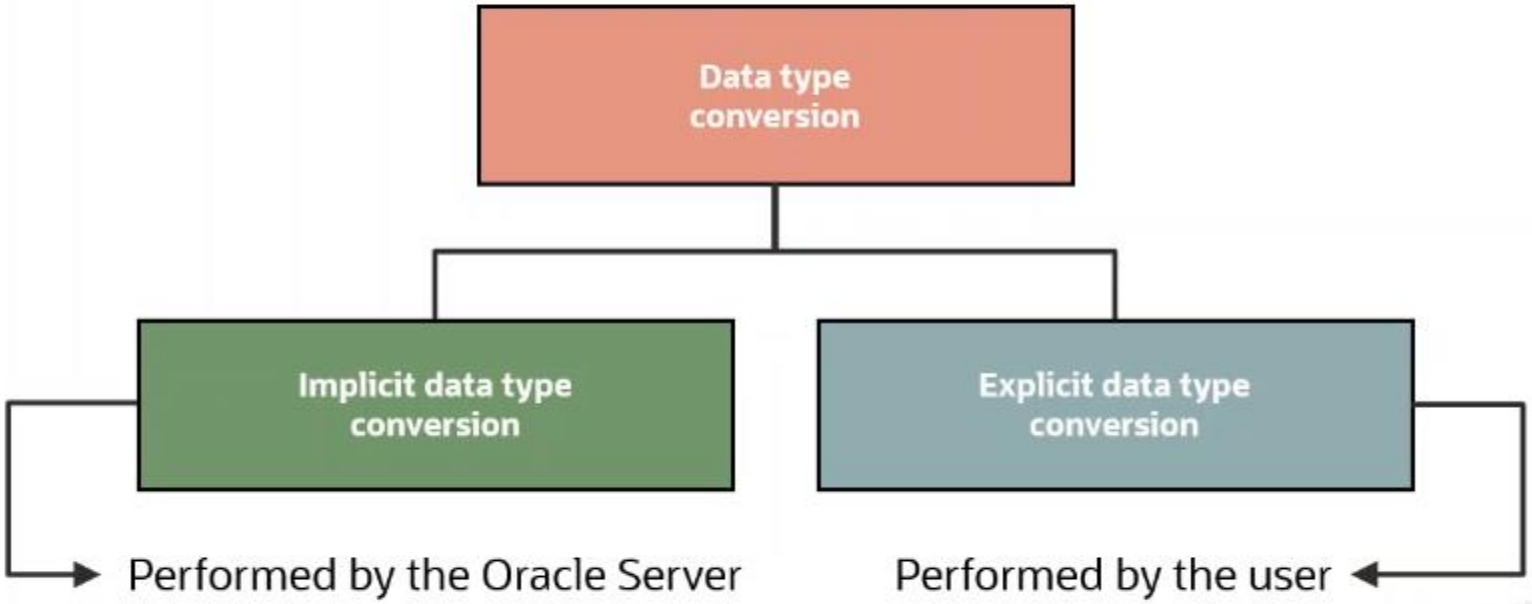
# Using Conversion Functions and Conditional Expressions

# Objectives

After completing this lesson, you should be able to do the following:

- Describe the various types of conversion functions that are available in SQL
- Use the `TO_CHAR`, `TO_NUMBER`, and `TO_DATE` conversion functions
- Apply conditional expressions in a `SELECT` statement

# Conversion Functions



# Implicit Data Type Conversion of Strings to Numbers

In expressions, the database can automatically convert strings to numbers. In this example, two strings are concatenated and then implicitly converted to a number for comparison with the numeric department ID in the `WHERE` clause.

```
SELECT first_name, last_name, department_id
FROM employees WHERE department_id < CONCAT('9', '0');
```

| | FIRST_NAME | LAST_NAME | DEPARTMENT_ID |
|---|---|---|---|
| 1 | Ellen | Abel | 80 |
| 2 | Curtis | Davies | 50 |
| 3 | Bruce | Ernst | 60 |
| 4 | Pat | Fay | 20 |
| 5 | Michael | Hartstein | 20 |
| 6 | Alexander | Hunold | 60 |
| 7 | Diana | Lorentz | 60 |
| 8 | Randall | Matos | 50 |
| 9 | Kevin | Mourgos | 50 |
| 10 | Trenna | Rajs | 50 |
| 11 | Jonathon | Taylor | 80 |
| 12 | Peter | Vargas | 50 |
| 13 | Jennifer | Whalen | 10 |
| 14 | Eleni | Zlotkey | 80 |

# Implicit Data Type Conversion of Numbers to Strings

In expressions, the database can automatically convert numbers to strings. In this example, the salary column is converted to a string to determine if it contains a character.

```
SELECT first_name, last_name, salary
FROM employees
WHERE INSTR(salary, '5') > 0;
```

| | FIRST_NAME | LAST_NAME | SALARY |
|---|---|---|---|
| 1 | Kevin | Mourgos | 5800 |
| 2 | Trenna | Rajs | 3500 |
| 3 | Peter | Vargas | 2500 |
| 4 | Eleni | Zlotkey | 10500 |

| # | first_name | last_name | salary |
|---|---|---|---|
| 1 | Kevin | Mourgos | 5800.00 |
| 2 | Trenna | Rajs | 3500.00 |
| 3 | Peter | Vargas | 2500.00 |
| 4 | Eleni | Zlotkey | 10500.00 |

# Using the TO_CHAR Function with Dates

Example:

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

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

| | EMPLOYEE_ID | MONTH_HIRED |
|---|---|---|
| 1 | 205 | 06/10 |

# Elements of the Date Format Model

| Element | Result |
|---|---|
| YYYY | Full year in numbers |
| YEAR | Year spelled out (in English) |
| MM | Two-digit value for the month |
| MONTH | Full name of the month |
| MON | Three-letter abbreviation of the month |
| DY | Three-letter abbreviation of the day of the week |
| DAY | Full name of the day of the week |
| DD | Numeric day of the month |

# Elements of the Date Format Model

Time elements help you format the time portion of the date:

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

Add character strings by enclosing them within double quotation marks:

| | |
|---|---|
| DD "of" MONTH | 12 of OCTOBER |

Number suffixes help in spelling out numbers:

| | |
|---|---|
| ddspth | fourteenth |

# Using the TO_CHAR Function with Dates

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

| | LAST_NAME | HIREDATE |
|---|---|---|
| 1 | King | 17 June 2011 |
| 2 | Kochhar | 21 September 2009 |
| 3 | De Haan | 13 January 2009 |
| 4 | Hunold | 3 January 2014 |
| 5 | Ernst | 21 May 2015 |
| 6 | Lorentz | 7 February 2015 |
| 7 | Mourgos | 16 November 2015 |
| 8 | Rajs | 17 October 2011 |

...

# Using the TO_CHAR Function with Numbers

These are some of the format elements that you can use with the TO_CHAR function to display a number value as a character:

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

| Element | Result |
|---|---|
| 9 | Represents a number |
| 0 | Forces a zero to be displayed |
| $ | Places a floating dollar sign |
| L | Uses the floating local currency symbol |
| . | Prints a decimal point |
| , | Prints a comma as a thousands indicator |

# Using the TO_CHAR Function with Numbers

Let us look at an example:

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

| | SALARY |
|---|---|
| 1 | $6,000.00 |

# Using the TO_NUMBER and TO_DATE Functions

- Convert a character string to a number format using the TO_NUMBER function:

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

- Convert a character string to a date format using the TO_DATE function:

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

# Using TO_CHAR and TO_DATE Functions with the RR Date Format

To find employees hired before 2010, use the RR date format, which produces the correct result if the command is run now or before the year 2049:

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM  employees
WHERE hire_date < TO_DATE('01 Jan, 10','DD Mon,RR');
```

| | LAST_NAME | TO_CHAR(HIRE_DATE,'DD-MON-YYYY') |
|---|---|---|
| 1 | Kochhar | 21-Sep-2009 |
| 2 | De Haan | 13-Jan-2009 |

# Using the CAST() function in Oracle

CAST lets you convert one data type to another.

```
CAST(input_value as destination_type)
```

Examples:

1

```
SELECT first_name, last_name, department_id
FROM employees
WHERE department_id < CAST(CONCAT('9', '0') AS
DECIMAL(2,0));
```

2

```
SELECT first_name, last_name, salary
FROM employees
WHERE INSTR(CAST(salary AS VARCHAR2(30)), '5')
> 0;
```

# General functions

- General functions:
  - NVL
  - NVL2
  - NULLIF
  - COALESCE

# General Functions

The following functions pertain to using nulls and can be used with any data type:

```
NVL (expr1, expr2)
```

```
NVL2 (expr1, expr2, expr3)
```

```
NULLIF (expr1, expr2)
```

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

# NVL Function (Oracle)

Converts a null value to an actual value:

- Data types that can be used are date, character, and number.
- Data types for both expressions must match.
- Examples:

| Oracle |
| --- |
| NVL(commission_pct,0) |
| NVL(hire_date,'01-JAN-97') |
| NVL(job_id,'No Job Yet') |

```
NVL (expr1,
expr2)
```

# Using the NVL Function in Oracle

```
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 |
|---|---|---|---|---|
| 1 | King | 24000 | 0 | 288000 |
| 2 | Kochhar | 17000 | 0 | 204000 |
| 3 | De Haan | 17000 | 0 | 204000 |
| 4 | Hunold | 9000 | 0 | 108000 |
| 5 | Ernst | 6000 | 0 | 72000 |
| 6 | Lorentz | 4200 | 0 | 50400 |
| 7 | Mourgos | 5800 | 0 | 69600 |
| 8 | Rajs | 3500 | 0 | 42000 |
| 9 | Davies | 3100 | 0 | 37200 |
| 10 | Matos | 2600 | 0 | 31200 |

...

# Using the NVL2 Function in Oracle

```
NVL2 (expr1, expr2, expr3)
```

```
SELECT last name,  salary, commission pct,
       NVL2(commission_pct,
            'SAL+COMM', 'SAL') income
FROM   employees WHERE department_id IN (50, 80);
```

| | LAST_NAME | SALARY | COMMISSION_PCT | INCOME |
|---|---|---|---|---|
| 1 | Mourgos | 5800 | (null) | SAL |
| 2 | Rajs | 3500 | (null) | SAL |
| 3 | Davies | 3100 | (null) | SAL |
| 4 | Matos | 2600 | (null) | SAL |
| 5 | Vargas | 2500 | (null) | SAL |
| 6 | Zlotkey | 10500 | 0.2 | SAL+COMM |
| 7 | Abel | 11000 | 0.3 | SAL+COMM |
| 8 | Taylor | 8600 | 0.2 | SAL+COMM |

# Using the COALESCE Function

- The advantage of the COALESCE function over the NVL or IFNULL functions is that the COALESCE function can take multiple alternative values.
- If the first expression is not null, the COALESCE function returns that expression; otherwise, it does a COALESCE of the remaining expressions.

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

- If the first expression is not null, the COALESCE function returns that expression; otherwise, it does a COALESCE of the remaining expressions.

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

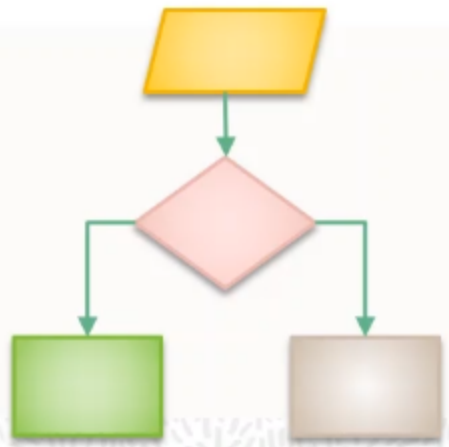
# Using the COALESCE Function

```
SELECT last name, salary, commission pct,
COALESCE((salary+(commission pct*salary)), salary+2000)
AS New_Salary
FROM   employees;
```

| | LAST_NAME | SALARY | COMMISSION_PCT | NEW_SALARY |
|---|---|---|---|---|
| 1 | King | 24000 | (null) | 26000 |
| 2 | Kochhar | 17000 | (null) | 19000 |
| 3 | De Haan | 17000 | (null) | 19000 |
| 4 | Hunold | 9000 | (null) | 11000 |
| 5 | Ernst | 6000 | (null) | 8000 |
| 6 | Lorentz | 4200 | (null) | 6200 |
| 7 | Mourgos | 5800 | (null) | 7800 |
| 8 | Rajs | 3500 | (null) | 5500 |
| 9 | Davies | 3100 | (null) | 5100 |
| 10 | Matos | 2600 | (null) | 4600 |
| 11 | Vargas | 2500 | (null) | 4500 |
| 12 | Zlotkey | 10500 | 0.2 | 12600 |
| 13 | Abel | 11000 | 0.3 | 14300 |
| 14 | Taylor | 8600 | 0.2 | 10320 |
| 15 | Grant | 7000 | 0.15 | 8050 |
| 16 | Whalen | 4400 | (null) | 6400 |
| 17 | Hartstein | 13000 | (null) | 15000 |
| 18 | Fay | 6000 | (null) | 8000 |
| 19 | Higgins | 12008 | (null) | 14008 |
| 20 | Gietz | 8300 | (null) | 10300 |

# Conditional Expressions

- Help provide the use of `IF-THEN-ELSE` logic within a SQL statement
- You can use the following methods:
  - `CASE` expression
  - Searched `CASE` expression
  - `DECODE` function



# CASE Expression

Facilitates conditional inquiries by doing the work of an IF-THEN-ELSE statement:

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

# Using the CASE Expression

```
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     AS REVISED_SALARY
FROM   employees;
```

| | LAST_NAME | JOB_ID | SALARY | REVISED_SALARY |
|---|---|---|---|---|
| 1 | King | AD_PRES | 24000 | 24000 |
| ... | | | | |
| 4 | Hunold | IT_PROG | 9000 | 9900 |
| 5 | Ernst | IT_PROG | 6000 | 6600 |
| 6 | Lorentz | IT_PROG | 4200 | 4620 |
| 7 | Mourgos | ST_MAN | 5800 | 5800 |
| 8 | Rajs | ST_CLERK | 3500 | 4025 |
| 9 | Davies | ST_CLERK | 3100 | 3565 |
| 10 | Matos | ST_CLERK | 2600 | 2990 |
| 11 | Vargas | ST_CLERK | 2500 | 2875 |
| ... | | | | |
| 13 | Abel | SA_REP | 11000 | 13200 |
| 14 | Taylor | SA_REP | 8600 | 10320 |
| 15 | Grant | SA_REP | 7000 | 8400 |

# Searched CASE Expression

```
CASE
     WHEN condition1 THEN use_expression1
     WHEN condition2 THEN use_expression2
     WHEN condition3 THEN use_expression3
     ELSE default_use_expression
END
```

```
SELECT last_name,salary,
(CASE WHEN salary<5000 THEN 'Low'
      WHEN salary<10000 THEN 'Medium'
      WHEN salary<20000 THEN 'Good'
      ELSE 'Excellent'
END) AS qualified_salary
FROM employees;
```