Database Programming with SQL

5-2
NULL Functions
Objectives

This lesson covers the following objectives:

• Demonstrate and explain the evaluation of a nested function
• List at least four general functions that work with any data type and relate to handling null values
• Explain the use of the COALESCE and the NVL functions
• Explain the use of general functions to deal with null values in data
• Construct and execute a SQL query that correctly applies NVL, NVL2, NULLIF, and COALESCE single-row functions
Purpose

• Besides functions that control how data is formatted or converted to another type, SQL uses a set of general functions designed specifically to deal with null values.

• You may be wondering how a value that is unavailable, unassigned, unknown, or inapplicable can deserve so much attention.

• Null may be "nothing," but it can affect how expressions are evaluated, how averages are computed, and where a value appears in a sorted list.

• This lesson is all about handling null values.
How Functions are Evaluated

• Up to now, you have applied single-row functions in simple statements.
• It is possible, however, to nest functions to any depth.
• It is important to know how nested functions are evaluated.
• "Nesting" refers to one thing being contained within another thing (like an egg contained within a nest).
• The following example is a nested function.
• The evaluation process begins from the innermost level to the outermost level.
How Functions are Evaluated

SELECT TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'FRIDAY'),
          'fmDay, Month DDth, YYYY') AS "Next Evaluation"
FROM employees
WHERE employee_id=100;

• The results are:
  – Friday, December 18th, 1987
How Functions are Evaluated

SELECT TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'FRIDAY'), 'fmDay, Month DDth, YYYY') AS "Next Evaluation"
FROM employees
WHERE employee_id=100;

• Step 1: The hire date is going to have six months added to it.

• Step 2: The first Friday following the day returned at Step 1 will be identified.

• Step 3: The default date format will be formatted to read and display the date returned by Step 2 in a format similar to: Friday, December 18th, 1987, and will appear in the output under the column name "Next Evaluation."
Functions Pertaining to Null Values

• At the beginning of the course, the term "null" was introduced.

• Null is the value that is unavailable, unassigned, unknown, or inapplicable.

• As a result, we cannot test to see if it is the same as another value, because we do not know what value it has.

• It isn't equal to anything, not even zero!

• But just because it really isn't anything doesn't mean that it is not important.
Functions Pertaining to Null Values

• Imagine this question: Is it true that $X = Y$?
• In order to answer you have to know the values of $X$ and $Y$.
• Oracle has four general functions that pertain to the use of null values.
• The four functions are:
  – NVL
  – NVL2
  – NULLIF
  – COALESCE
NVL Function

• The NVL function converts a null value to a known value of a fixed data type, either date, character, or number.

• The data types of the null value column and the new value must be the same.

• The NVL function is:

\[
\text{NVL (expression 1 value that may contain a null, expression 2 value to substitute for null)}
\]

• NVL (value or column that may contain a null, value to substitute for null)
NVL Function

- The following query uses the NVL function with character data types:

```sql
SELECT country_name, NVL(internet_extension, 'None')
    AS "Internet extn"
FROM wf_countries
WHERE location = 'Southern Africa'
ORDER BY internet_extension DESC;
```

- Null values are replaced with the text 'None'.

<table>
<thead>
<tr>
<th>COUNTRY_NAME</th>
<th>Internet extn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Europa Island</td>
<td>None</td>
</tr>
<tr>
<td>Juan de Nova Island</td>
<td>None</td>
</tr>
<tr>
<td>Republic of Zimbabwe</td>
<td>.zw</td>
</tr>
<tr>
<td>Republic of Zambia</td>
<td>.zm</td>
</tr>
<tr>
<td>Republic of South Africa</td>
<td>.za</td>
</tr>
</tbody>
</table>
NVL Function

• The data types of the null value column and the new value must be the same as shown in the following examples:

<table>
<thead>
<tr>
<th>Examples:</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT last_name, NVL(commission_pct, 0) FROM employees WHERE department_id IN(80,90);</td>
<td>Zlotkey .2</td>
</tr>
<tr>
<td></td>
<td>Abel .3</td>
</tr>
<tr>
<td></td>
<td>Taylor .2</td>
</tr>
<tr>
<td></td>
<td>King 0</td>
</tr>
<tr>
<td>SELECT NVL(date_of_independence, '01/01/9999') FROM wf_countries;</td>
<td>1-Jul-1867</td>
</tr>
<tr>
<td></td>
<td>15-Sep-1821</td>
</tr>
<tr>
<td></td>
<td>5-Jul-1975</td>
</tr>
<tr>
<td></td>
<td>01/01/9999</td>
</tr>
</tbody>
</table>
NVL Function

• You can use the NVL function to convert column values containing nulls to a number before doing calculations.

• When an arithmetic calculation is performed with null, the result is null.

• The NVL function can convert the null value to a number before arithmetic calculations are done to avoid a null result.
NVL Function

• In the example, the commission_pct column in the employees table contains null values.

• The NVL function is used to change the null to zero before arithmetic calculations.

```sql
SELECT last_name, NVL(commission_pct, 0)*250 AS "Commission"
FROM employees
WHERE department_id IN(80,90);
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>Commission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zlotkey</td>
<td>50</td>
</tr>
<tr>
<td>Abel</td>
<td>75</td>
</tr>
<tr>
<td>Taylor</td>
<td>50</td>
</tr>
<tr>
<td>King</td>
<td>0</td>
</tr>
<tr>
<td>Kochhar</td>
<td>0</td>
</tr>
<tr>
<td>De Haan</td>
<td>0</td>
</tr>
</tbody>
</table>
NVL2 Function

- The NVL2 function evaluates an expression with three values.
- If the first value is not null, then the NVL2 function returns the second expression.
- If the first value is null, then the third expression is returned.
- The values in expression 1 can have any data type.
- Expression 2 and expression 3 can have any data type except LONG.
- The data type of the returned value is always the same as the data type of expression 2, unless expression 2 is character data, in which case the returned type is VARCHAR2.
NVL2 Function

• The NVL2 function is:

NVL2 (expression 1 value that may contain a null, expression 2 value to return if expression 1 is not null, expression 3 value to replace if expression 1 is null)

• An easy way to remember NVL2 is to think, "if expression 1 has a value, substitute expression 2; if expression 1 is null, substitute expression 3."
NVL2 Function

• The NVL2 function shown uses number data types for expressions 1, 2 and 3.

```sql
SELECT last_name, salary,
    NVL2(commission_pct, salary + (salary * commission_pct), salary)
AS income
FROM employees
WHERE department_id IN(80,90);
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>SALARY</th>
<th>INCOME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zlotkey</td>
<td>10500</td>
<td>12600</td>
</tr>
<tr>
<td>Abel</td>
<td>11000</td>
<td>14300</td>
</tr>
<tr>
<td>Taylor</td>
<td>8600</td>
<td>10320</td>
</tr>
<tr>
<td>King</td>
<td>24000</td>
<td>24000</td>
</tr>
<tr>
<td>Kochhar</td>
<td>17000</td>
<td>17000</td>
</tr>
<tr>
<td>De Haan</td>
<td>17000</td>
<td>17000</td>
</tr>
</tbody>
</table>
NULLIF Function

• The NULLIF function compares two expressions.
• If they are equal, the function returns null.
• If they are not equal, the function returns the first expression.
• The NULLIF function is:

  NULLIF(expression 1, expression 2)
NULLIF Function

• In this example, NULLIF compares the length of employees first and last names.

• If the length of both names are the same, NULLIF returns NULL (as in row 2 Curtis Davies), otherwise expression 1 LENGTH of first_name is returned.

```sql
SELECT first_name, LENGTH(first_name) AS "Length FN", last_name, LENGTH(last_name) AS "Length LN", NULLIF(LENGTH(first_name), LENGTH(last_name)) AS "Compare Them"
FROM employees;
```

<table>
<thead>
<tr>
<th>FIRST_NAME</th>
<th>Length FN</th>
<th>LAST_NAME</th>
<th>Length LN</th>
<th>Compare Them</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ellen</td>
<td>5</td>
<td>Abel</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Curtis</td>
<td>6</td>
<td>Davies</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>Lex</td>
<td>3</td>
<td>De Haan</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>
COALESCE Function

• The COALESCE function is an extension of the NVL function, except COALESCE can take multiple values.

• The word "coalesce" literally means "to come together" and that is what happens.

• If the first expression is null, the function continues down the line until a not null expression is found.

• Of course, if the first expression has a value, the function returns the first expression and the function stops.

• The COALESCE function is:

  COALESCE (expression 1, expression 2, ...expression n)
COALESCE Function

• Examine the SELECT statement from the employees table shown at right.

• If an employee has a value (not NULL) for commission_pct, this is returned, otherwise if salary has a value, return salary.

• If an employees commission_pct and salary are NULL, return the number 10.

SELECT last_name,
       COALESCE(commission_pct, salary, 10) 
AS "Comm"
FROM employees
ORDER BY commission_pct;

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>Comm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grant</td>
<td>.15</td>
</tr>
<tr>
<td>Zlotkey</td>
<td>.2</td>
</tr>
<tr>
<td>Taylor</td>
<td>.2</td>
</tr>
<tr>
<td>Abel</td>
<td>.3</td>
</tr>
<tr>
<td>Higgins</td>
<td>12000</td>
</tr>
<tr>
<td>Gietz</td>
<td>8300</td>
</tr>
</tbody>
</table>
Terminology

Key terms used in this lesson included:

• NVL
• NVL2
• NULLIF
• COALESCE
Summary

In this lesson, you should have learned how to:

• Demonstrate and explain the evaluation of a nested function
• List at least four general functions that work with any data type and relate to handling null values
• Explain the use of the COALESCE and the NVL functions
• Explain the use of general functions to deal with null values in data
• Construct and execute a SQL query that correctly applies NVL, NVL2, NULLIF, and COALESCE single-row functions