Database Programming with PL/SQL

2-7

Good Programming Practices
Objectives

This lesson covers the following objectives:

• List examples of good programming practices
• Accurately insert comments into PL/SQL code
• Create PL/SQL code that follows formatting guidelines to produce readable code
Purpose

• Good programming practices are techniques that you can follow to create the best code possible.

• Programming practices cover everything from making code more readable to creating code with faster performance.

• Software engineering teams often follow a style guide so that everyone on the team uses the same techniques.

• This makes it easier to read and modify code written by others.
Good Programming Practices

Several examples of good programming practices have already been demonstrated and/or discussed in this course:

• Use explicit data type conversions because implicit data type conversions can be slower and the rules can change in later software releases.

• Use meaningful identifiers when declaring variables, constants, and parameters.

• Declare one variable or constant identifier per line for better readability and code maintenance.
Good Programming Practices

Other good programming practices demonstrated and/or discussed:

• Avoid ambiguity when choosing identifiers.
• Use the %TYPE attribute to declare a variable according to another previously declared variable or database column.
• Use the NOT NULL constraint when declaring a variable that must hold a value.
Programming Guidelines

Other programming guidelines include:

• Documenting code with comments
• Developing a case convention for the code
• Developing naming conventions for identifiers and other objects
• Enhancing readability by indenting
Commenting Code Example

- Prefix single-line comments with two dashes (--).
- Place multiple-line comments between the symbols " /* " and " */ ".

```sql
DECLARE
  -- Created by Clara Oswald
  ...
  v_annual_sal NUMBER (9,2);
BEGIN   -- Start of executable section

/* Compute the annual salary based on the monthly salary input from the user */

  v_annual_sal := v_monthly_sal * 12;
  ...
END;    -- End of executable section
```
Variable Scope

• Case Conventions are shown below.
• The following table provides guidelines for writing code in uppercase and lowercase to help you distinguish keywords from named objects.

<table>
<thead>
<tr>
<th>Category</th>
<th>Case Convention</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL keywords</td>
<td>Uppercase</td>
<td>SELECT, INSERT</td>
</tr>
<tr>
<td>PL/SQL keywords</td>
<td>Uppercase</td>
<td>DECLARE, BEGIN, IF</td>
</tr>
<tr>
<td>Data types</td>
<td>Uppercase</td>
<td>VARCHAR2, BOOLEAN</td>
</tr>
<tr>
<td>Identifiers</td>
<td>Lowercase</td>
<td>v_salary, emp_cursor, c_tax_rate, p_empno</td>
</tr>
<tr>
<td>Tables and columns</td>
<td>Lowercase</td>
<td>employees, dept_id, salary, hire_date</td>
</tr>
</tbody>
</table>
Naming Conventions

• The naming of identifiers should be clear, consistent, and unambiguous.

• One commonly-used convention is to name:
  – Variables starting with \texttt{v_}
  – Constants starting with \texttt{c_}
  – Parameters starting with \texttt{p_} (for passing to procedures and functions)
Naming Conventions

Examples:

• `v_date_of_birth`
• `v_last_name`
• `c_tax_rate`
• `c_commission_rate`
• `p_employee_id`
• `p_salary`
Indenting Code

For clarity, indent each level of code. Examples:

```
BEGIN
  IF x = 0 THEN
    y := 1;
  END IF;
END;
```

```
DECLARE
  v_deptno NUMBER(4);
  v_location_id NUMBER(4);
BEGIN
  SELECT department_id, location_id
  INTO v_deptno, v_location_id
  FROM departments
  WHERE department_name = 'Sales';
  DBMS_OUTPUT.PUTLINE(...
END;
```
Summary

In this lesson, you should have learned how to:

• List examples of good programming practices
• Accurately insert comments into PL/SQL code
• Create PL/SQL code that follows formatting guidelines to produce readable code