2 Writing Executable Statements

ORACLE

Objectives

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

- Describe the significance of the executable section
- Use identifiers correctly
- Write statements in the executable section
- Describe the rules of nested blocks
- Execute and test a PL/SQL block
- Use coding conventions

ORACLE

PL/SQL Block Syntax and Guidelines

- Statements can continue over several lines.
- Lexical units can be classified as:
  - Delimiters
  - Identifiers
  - Literals
  - Comments

ORACLE

Identifiers

- Can contain up to 30 characters
- Must begin with an alphabetic character
- Can contain numerals, dollar signs, underscores, and number signs
- Cannot contain characters such as hyphens, slashes, and spaces
- Should not have the same name as a database table column name
- Should not be reserved words

ORACLE

PL/SQL Block Syntax and Guidelines

- **Literals**
  - **Character and date literals must be enclosed in single quotation marks.**

```
v_name := 'Henderson';
```

  - Numbers can be simple values or scientific notation.
- A slash (/) runs the PL/SQL block in a script file or in some tools such as *i*SQL*PLUS.

ORACLE

Commenting Code

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

  Example:

```
DECLARE
...
    v_sal NUMBER (9,2);
BEGIN
    /* Compute the annual salary based on the
       monthly salary input from the user */
    v_sal := :g_monthly_sal * 12;
END; -- This is the end of the block
```

ORACLE

SQL Functions in PL/SQL

- Available in procedural statements:
  - Single-row number
  - Single-row character
  - Data type conversion
  - Date
  - Timestamp
  - GREATEST and LEAST
  - Miscellaneous functions
- Not available in procedural statements:
  - DECODE
  - Group functions

} Same as in SQL

ORACLE

SQL Functions in PL/SQL: Examples

- **Build the mailing list for a company.**

```
v_mailing_address := v_name||CHR(10)||
                     v_address||CHR(10)||v_state||
                     CHR(10)||v_zip;
```

- Convert the employee name to lowercase.

```
v_ename := LOWER(v_ename);
```

ORACLE

Data Type Conversion

- Convert data to comparable data types.
- Mixed data types can result in an error and affect performance.
- Conversion functions:
  - TO_CHAR
  - TO_DATE
  - TO_NUMBER.

```
DECLARE
v_date DATE := TO_DATE('12-JAN-2001', 'DD-MON-YYYY');
BEGIN
```

ORACLE

Data Type Conversion

This statement produces a compilation error if the variable v_date is declared as a DATE data type.

```
v_date :=   'January 13, 2001';
```

ORACLE

Data Type Conversion

To correct the error, use the TO_DATE conversion function.

```
v_date := TO_DATE ('January 13, 2001',
                   'Month DD, YYYY');
```

ORACLE

Nested Blocks and Variable Scope

- PL/SQL blocks can be nested wherever an executable statement is allowed.
- A nested block becomes a statement.
- An exception section can contain nested blocks.
- The scope of an identifier is that region of a program unit (block, subprogram, or package) from which you can reference the identifier.

ORACLE

Nested Blocks and Variable Scope

Example:

```
...
  x  BINARY_INTEGER;
BEGIN
  ...
  DECLARE
    y  NUMBER;
  BEGIN
      y:= x;
  END;
  ...
END;
```

Scope of *x*

Scope of *y*

Identifier Scope

An identifier is visible in the regions where you can reference the identifier without having to qualify it:

- A block can look up to the enclosing block.
- A block cannot look down to enclosed blocks.

ORACLE

Qualify an Identifier

- The qualifier can be the label of an enclosing block.
- Qualify an identifier by using the block label prefix.

```
<<outer>>
   DECLARE
      birthdate DATE;
   BEGIN
         DECLARE
             birthdate DATE;
         BEGIN
            ...
            outer.birthdate :=
                              TO_DATE('03-AUG-1976',
                                      'DD-MON-YYYY');
         END;
     ....
    END;
```

ORACLE

Determining Variable Scope

Class Exercise

```
<<outer>>
DECLARE
  v_sal        NUMBER(7,2)  := 60000;
  v_comm       NUMBER(7,2)  := v_sal * 0.20;
  v_message    VARCHAR2(255) := ' eligible for commission';
BEGIN
  DECLARE
       v_sal           NUMBER(7,2)  := 50000;
       v_comm          NUMBER(7,2)  := 0;
       v_total_comp    NUMBER(7,2)  := v_sal + v_comm;
  BEGIN
       v_message := 'CLERK not'||v_message;
       outer.v_comm := v_sal * 0.30;
  END;
    v_message := 'SALESMAN'||v_message;
END;
```

1 →

2 →

Operators in PL/SQL

- Logical
- Arithmetic
- Concatenation
- Parentheses to control order of operations

- Exponential operator (**)

ORACLE

Operators in PL/SQL

Examples:

- Increment the counter for a loop.

```
v_count          := v_count + 1;
```

- Set the value of a Boolean flag.

```
v_equal          := (v_n1 = v_n2);
```

- Validate whether an employee number contains a value.

```
v_valid          := (v_empno IS NOT NULL);
```

ORACLE

Programming Guidelines

Make code maintenance easier by:

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

ORACLE

Indenting Code

For clarity, indent each level of code.

Example:

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

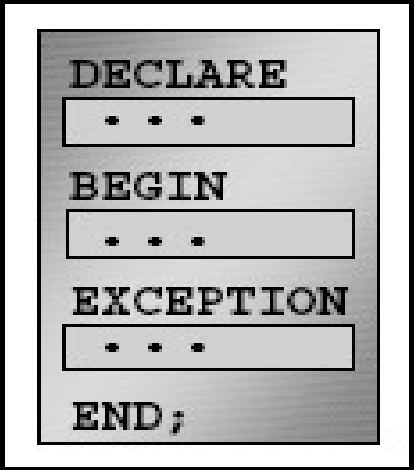
```
DECLARE
   v_deptno          NUMBER(4);
   v_location_id     NUMBER(4);
BEGIN
   SELECT   department_id,
            location_id
   INTO     v_deptno,
            v_location_id
   FROM     departments
   WHERE    department_name
            = 'Sales';
...
END;
/
```

ORACLE

Summary

In this lesson you should have learned that:

- PL/SQL block syntax and guidelines
- How to use identifiers correctly
- PL/SQL block structure: nesting blocks and scoping rules
- PL/SQL programming:
  - Functions
  - Data type conversions
  - Operators
  - Conventions and guidelines



ORACLE

Practice 2 Overview

This practice covers the following topics:

- Reviewing scoping and nesting rules
- Developing and testing PL/SQL blocks

ORACLE