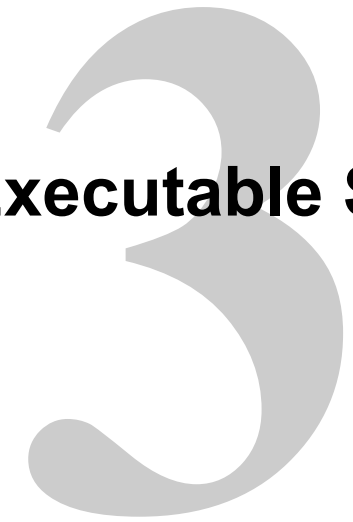


Writing Executable Statements

Objectives

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

- Identify lexical units in a PL/SQL block
- Use built-in SQL functions in PL/SQL
- Describe when implicit conversions take place and when explicit conversions have to be dealt with
- Write nested blocks and qualify variables with labels
- Write readable code with appropriate indentation
- Use sequences in PL/SQL expressions

Lexical Units in a PL/SQL Block

Lexical units:

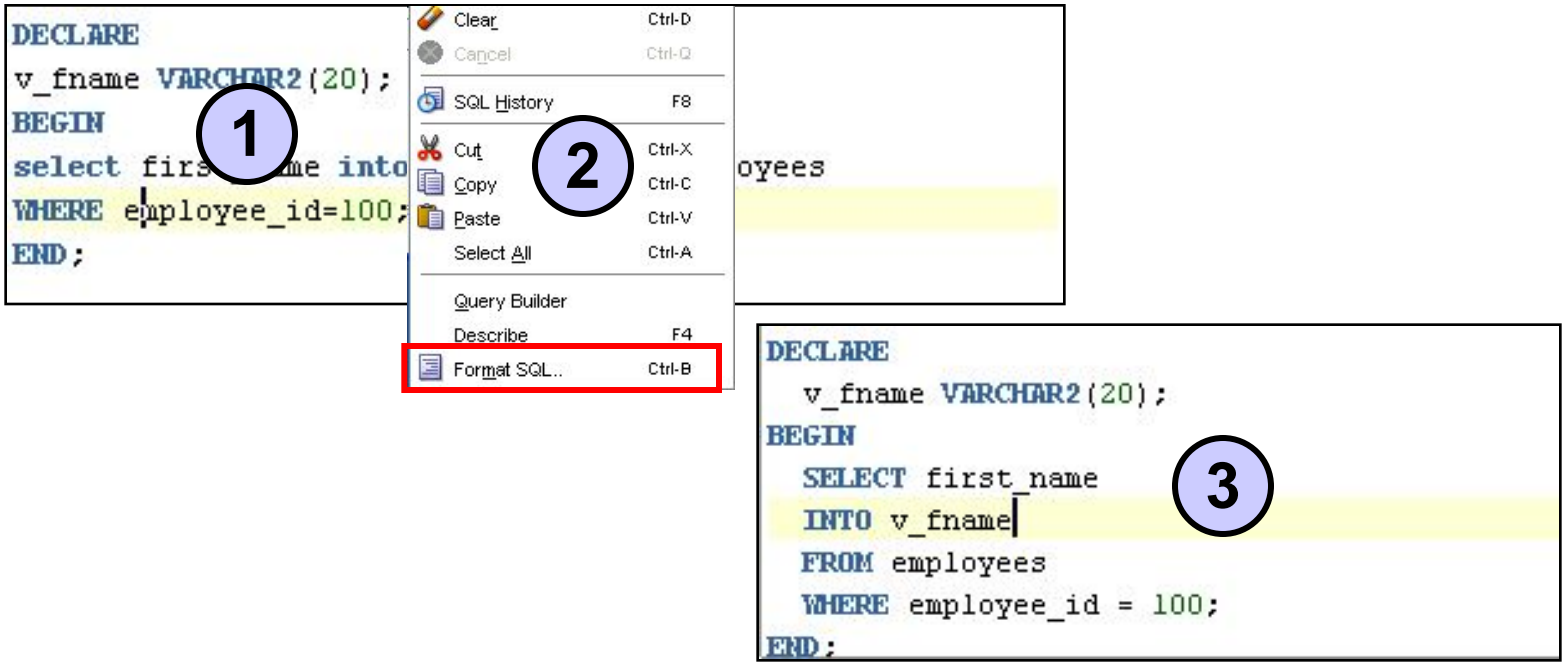
- Are building blocks of any PL/SQL block
- Are sequences of characters including letters, numerals, tabs, spaces, returns, and symbols
- Can be classified as:
  - Identifiers: `v_fname, c_percent`
  - Delimiters: `; , +, -`
  - Literals: `John, 428, True`
  - Comments: `--, /* */`

PL/SQL Block Syntax and Guidelines

- Literals
  - Character and date literals must be enclosed in single quotation marks.
  - Numbers can be simple values or in scientific notation.

```
name := 'Henderson';
```

- Statements can span several lines.



Commenting Code

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

Example:

```
DECLARE
...
v_annual_sal NUMBER (9,2);
BEGIN
/* Compute the annual salary based on the
   monthly salary input from the user */
v_annual_sal := monthly_sal * 12;
--The following line displays the annual salary
DBMS_OUTPUT.PUT_LINE(v_annual_sal);
END;
/
```

SQL Functions in PL/SQL

- Available in procedural statements:
  - Single-row functions
- Not available in procedural statements:
  - `DECODE`
  - Group functions

SQL Functions in PL/SQL: Examples

- Get the length of a string:

```
v_desc_size INTEGER(5);
v_prod_description VARCHAR2(70):='You can use this
product with your radios for higher frequency';

-- get the length of the string in prod_description
v_desc_size:= LENGTH(v_prod_description);
```

- Get the number of months an employee has worked:

```
v_tenure:= MONTHS_BETWEEN (CURRENT_DATE, v_hiredate);
```

Using Sequences in PL/SQL Expressions

Starting in 11*g*:

```
DECLARE
  v_new_id NUMBER;
BEGIN
  v_new_id := my_seq.NEXTVAL;
END;
/
```

Before 11*g*:

```
DECLARE
  v_new_id NUMBER;
BEGIN
  SELECT my_seq.NEXTVAL INTO v_new_id FROM Dual;
END;
/
```

Data Type Conversion

- Converts data to comparable data types
- Is of two types:
  - Implicit conversion
  - Explicit conversion
- Functions:
  - `TO_CHAR`
  - `TO_DATE`
  - `TO_NUMBER`
  - `TO_TIMESTAMP`

Data Type Conversion

1
```
date_of_joining DATE:= '02-Feb-2000';
```

2
```
date_of_joining DATE:= 'February 02,2000';
```

3
```
date_of_joining DATE:= TO_DATE('February
02,2000','Month DD, YYYY');
```

Nested Blocks

PL/SQL blocks can be nested.

- An executable section (`BEGIN ... END`) can contain nested blocks.
- An exception section can contain nested blocks.

Nested Blocks: Example

```
DECLARE
 v_outer_variable VARCHAR2(20):='GLOBAL VARIABLE';
BEGIN
  DECLARE
   v_inner_variable VARCHAR2(20):='LOCAL VARIABLE';
  BEGIN
   DBMS_OUTPUT.PUT_LINE(v_inner_variable);
   DBMS_OUTPUT.PUT_LINE(v_outer_variable);
  END;
 DBMS_OUTPUT.PUT_LINE(v_outer_variable);
END;
```

```
anonymous block completed
LOCAL VARIABLE
GLOBAL VARIABLE
GLOBAL VARIABLE
```

Variable Scope and Visibility

```
DECLARE
 v_father_name VARCHAR2(20):='Patrick';
 v_date_of_birth DATE:='20-Apr-1972';
BEGIN
  DECLARE
   v_child_name VARCHAR2(20):='Mike';
   v_date_of_birth DATE:='12-Dec-2002';
  BEGIN
   DBMS_OUTPUT.PUT_LINE('Father''s Name: '||v_father_name);
   DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
   DBMS_OUTPUT.PUT_LINE('Child''s Name: '||v_child_name);
  END;
 DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
END;
/
```

1

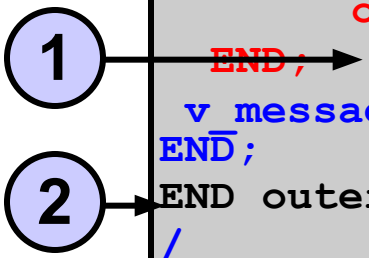
2

Qualify an Identifier

```
BEGIN <<outer>>
DECLARE
 v_father_name VARCHAR2(20):='Patrick';
 v_date_of_birth DATE:='20-Apr-1972';
BEGIN
  DECLARE
   v_child_name VARCHAR2(20):='Mike';
   v_date_of_birth DATE:='12-Dec-2002';
  BEGIN
   DBMS_OUTPUT.PUT_LINE('Father''s Name: '||v_father_name);
   DBMS_OUTPUT.PUT_LINE('Date of Birth: '
                        ||outer.v_date_of_birth);
   DBMS_OUTPUT.PUT_LINE('Child''s Name: '||v_child_name);
   DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
  END;
END;
END outer;
```

Determining Variable Scope: Example

```
BEGIN <<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;
END outer;
/
```



Operators in PL/SQL

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

Same as in SQL

- Exponential operator (**)

Operators in PL/SQL: Examples

- Increment the counter for a loop.

```
loop_count := loop_count + 1;
```

- Set the value of a Boolean flag.

```
good_sal := sal BETWEEN 50000 AND 150000;
```

- Validate whether an employee number contains a value.

```
valid  := (empno IS NOT NULL);
```

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

Indenting Code

For clarity, indent each level of code.

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

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

Quiz

You can use most SQL single-row functions such as number, character, conversion, and date single-row functions in PL/SQL expressions.

1. True
2. False

Summary

In this lesson, you should have learned how to:

- Identify lexical units in a PL/SQL block
- Use built-in SQL functions in PL/SQL
- Write nested blocks to break logically related functionalities
- Decide when to perform explicit conversions
- Qualify variables in nested blocks
- Use sequences in PL/SQL expressions

Practice 3: Overview

This practice covers the following topics:

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