1-3 Creating PL/SQL Blocks

Agustín Alejandro Mota Hinojosa

October 18, 2023

Contents

1 Vocabulary

1. Unnamed blocks of code not stored in the database and do not exist after they are executed

   Anonymous Blocks

2. A program that computes and returns a single value

   Subprogram declared as a function

3. Named PL/SQL blocks that are stored in the database and can be declared as procedures or functions

   Subprogram

4. Software that checks and translates programs written in high-level programming languages into binary code to execute

   Compiler

5. A program that performs an action, but does not have to return a value

   Subprogram declared as a procedure

2 Try / Solve it

1. Complete the following chart defining the syntactical requirements for a PL/SQL block:

	Optional or Mandatory?	Describe
`DECLARE`	optional	Contains declarations of all variables, constants, cursors, and user-defined exceptions that are referenced in the executable and exception sections.
`BEGIN`	mandatory	Contains SQL statements to retrieve data from the database and PL/SQL statements to manipulate data in the block.
`EXCEPTION`	optional	Specifies the actions to perform when errors and abnormal conditions arise in the executable section.
`END;`	mandatory	Ends a `BEGIN` statement

2. Which of the following PL/SQL blocks executes successfully? For the blocks that fail, explain why they fail
   Option D executes correctly without any problems

   ```
   DECLARE
       amount NUMBER(10);
   BEGIN
      DBMS_OUTPUT.PUT_LINE(amount);
   END;
   ```

3. Fill in the blanks:

   (a) PL/SQL blocks that have no names are called **anonymous**

   (b) `FUNCTION` and `PROCEDURE` are named blocks and are stored in the database.

4. In Application Express, create and execute a simple anonymous block that outputs "Hello World."

   ```
   BEGIN
       DBMS_OUTPUT.PUT_LINE('Hello World');
   END;
   ```

5. Create and execute a simple anonymous block that does the following:

   (a) Declares a variable of datatype DATE and populates it with the date that is six months from today

   ```
   DECLARE
       six_months_from_today date;
   BEGIN
       six_months_from_today := add_months(sysdate,6);
   END;
   ```

   (b) Outputs "In six months, the date will be: <insert date>."

   ```
   BEGIN
       DBMS_OUTPUT.PUT_LINE('In six months, the date will be ' ||
                                 six_months_from_today);
   END;
   ```