Lab 1
Procedural Language/Structured Query Language

CSE 4508

Relational Database Management System Lab

September 18, 2024

Contents

1 PL/SQL

PL/SQL is a block-structured language where we can write code organized into blocks similar to Java/C/C++, although the coding style differs here due to not having any curly braces to define blocks, rather we define blocks using the BEGIN and END keywords. You can define PROCEDUREs (which execute some code without returning anything) and FUNCTIONs (which execute code and return some variable/record). And you can even define unnamed blocks in PL/SQL, otherwise known as anonymous blocks that can be called immediately upon defining using the / at the end.

1.1 Structure of PL/SQL Block

PL/SQL extends SQL by adding constructs found in procedural languages, resulting in a structural language that is more powerful than SQL. The basic unit in PL/SQL is a block. All PL/SQL programs are made up of blocks, which can be nested within each other.

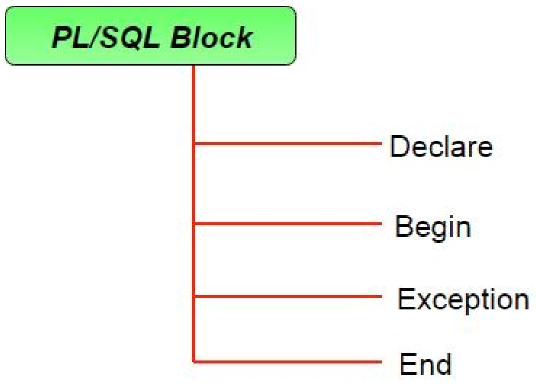


Figure 1: Structure of PL/SQL Block

1.2 Syntax of PL/SQL Block

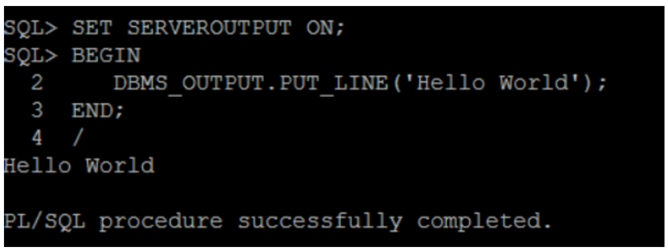
```
DECLARE
    declaration statements;

BEGIN
    executable statements;

EXCEPTION
    exception handling statements;

END;
/
```

Note: Remember to SET SERVEROUTPUT ON to see the results of the blocks.



```
SQL> SET SERVEROUTPUT ON;
SQL> BEGIN
  2      DBMS_OUTPUT.PUT_LINE('Hello World');
  3   END;
  4   /
Hello World

PL/SQL procedure successfully completed.
```

Figure 2: PL/SQL Syntax to show output

1.3 Example of PL/SQL Block

```
DECLARE
   v_salary NUMBER;
   v_bonus  NUMBER := 1000;

BEGIN
   SELECT salary INTO v_salary FROM employees WHERE employee_id = 101;

   IF v_salary > 5000 THEN
      v_salary := v_salary + v_bonus;
   END IF;

   DBMS_OUTPUT.PUT_LINE('Updated salary: ' || v_salary);

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Employee not found.');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred.');

END;
/
```

2 Function

A function in PL/SQL is a subprogram designed to perform a specific task and return a single value. Functions are commonly used when a value needs to be computed and returned, such as the result of a calculation, string manipulation, or database query. Unlike procedures, a function must always return a value.

2.1 Example of a Function

Define a function that, given the name of a department, returns the count of the number of instructors in that department.

```
CREATE FUNCTION dept_count (dept_name VARCHAR(20))
RETURNS INTEGER
BEGIN
    DECLARE d_count INTEGER;
    SELECT COUNT(*)
    INTO d_count
    FROM instructor
    WHERE instructor.dept_name = dept_name;
    RETURN d_count;
END;
/
```

Running the function:

```
SELECT dept_name, budget
FROM department
WHERE dept_count(dept_name) > 12;
```

2.2 Table Functions

```
CREATE FUNCTION instructor_of (dept_name CHAR(20))
RETURNS TABLE (
    ID VARCHAR(5),
    name VARCHAR(20),
    dept_name VARCHAR(20),
    salary NUMERIC(8,2))
RETURN TABLE
    (SELECT ID, name, dept_name, salary
     FROM instructor
     WHERE instructor.dept_name = instructor_of.dept_name);
/
```

Running the function:

```
SELECT *
FROM TABLE (instructor_of('Music'));
```

3 Procedure

A procedure in PL/SQL is also a subprogram. It can accept input and output parameters. It uses parameters to return results, but it does not use the RETURN statement. It is used to perform an action (eg. modifying data in the database).

3.1 Example of a Procedure

```
CREATE PROCEDURE dept_count_proc (dept_name IN VARCHAR(20), d_count OUT
   INTEGER)
BEGIN
    SELECT COUNT(*) INTO d_count
    FROM instructor
    WHERE instructor.dept_name = dept_count_proc.dept_name;
END;
/
```

Running the procedure:

```
DECLARE d_count INTEGER;
CALL dept_count_proc('Physics', d_count);
```

4 Language Constructs

PL/SQL syntax for conditionals, loops, functions, and mathematical operations.

4.1 Conditionals

```
DECLARE
    num NUMBER := 7;
BEGIN
    IF MOD(num, 2) = 0 THEN
        DBMS_OUTPUT.PUT_LINE('The number is even.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('The number is odd.');
    END IF;
END;
/
```

4.2 Loops

Print numbers from 1 to 5 using a WHILE loop.

```
DECLARE
    counter NUMBER := 1;
```

```
BEGIN
    WHILE counter <= 5 LOOP
        DBMS_OUTPUT.PUT_LINE('Number: ' || counter);
        counter := counter + 1;
    END LOOP;
END;
/
```

Print numbers from 1 to 5 using a FOR loop

```
BEGIN
    FOR i IN 1..5 LOOP
        DBMS_OUTPUT.PUT_LINE('Number: ' || i);
    END LOOP;
END;
/
```

4.3 Function inside a Procedure

```
CREATE OR REPLACE FUNCTION calculate_square(n NUMBER) RETURN NUMBER IS
BEGIN
    RETURN n * n;
END calculate_square;
/

CREATE OR REPLACE PROCEDURE print_square(num IN NUMBER) IS
    result NUMBER;
BEGIN
    result := calculate_square(num);
    DBMS_OUTPUT.PUT_LINE('The square of ' || num || ' is: ' || result);
END print_square;
/

BEGIN
    print_square(5);
END;
/
```

4.4 Mathematical Functions

```
DECLARE
    num NUMBER := 16;
    result NUMBER;
BEGIN
    result := SQRT(num);
    DBMS_OUTPUT.PUT_LINE('The square root of ' || num || ' is ' || result
      );
END;
/
```

```
DECLARE
    num1 NUMBER := 10;
    num2 NUMBER := 3;
BEGIN
    DBMS_OUTPUT.PUT_LINE('The remainder when ' || num1 || ' is divided by
        ' || num2 || ' is ' || MOD(num1, num2));
END;
/
```

5 Tasks

Task 1

Write a block of PL/SQL code that checks whether the current year is the starting year of a new decade (years such as 2000, 2010, 2020) and prints either “Yes” or “no”. After this, it should print the current decade (e.g., for 2000 to 2009, print ‘The 2000s’, for 2010 to 2019, print ‘The 2010s’).

Task 2

Write a PL/SQL procedure (or function) called `prime_generator` which takes only one input: `s`. The function will keep generating prime numbers, starting from 2, until the sum of all the prime numbers generated so far is less than or equal to `s`. For example, if `s = 20`, the output will be: 2, 3, 5, 7 (since $2+3+5+7=17$. “11” is not included since that would make the sum greater than 20). Execute this function from a PL/SQL block.

Task 3

As a course teacher, you are asked to calculate the final marks of your students based on various assessments. The final marks should be calculated based on the following criteria:

1. Attendance: 10%
2. Quiz: 15%
3. Mid: 25%
4. Final: 50%

Task 3.1

Design a student table with the required columns and insert 5 dummy samples.

Task 3.2

Write a PL/SQL procedure that calculates the total marks based on the aforementioned criteria.

Task 3.3

Write another PL/SQL procedure that calculates grade based on the total marks. Use this simplified grading scheme:

$$\mathrm{A}: \geq 80\%,\ \mathrm{B}: \geq 70\%,\ \mathrm{C}: \geq 60\%,\ \mathrm{D}: \geq 40\%,\ \mathrm{F}: < 40\%$$

Submission Guidelines

You are required to submit a report that includes your code, a detailed explanation, and the corresponding output. Rename your report as `<StudentID_Lab_1.pdf>`.

- Your lab report must be generated in LaTeX. Recommended tool: Overleaf. You can use some of the available templates in Overleaf.
- Include all code/pseudo code relevant to the lab tasks in the lab report.
- Please provide citations for any claims that are not self-explanatory or commonly understood.
- It is recommended to use vector graphics (e.g., `.pdf`, `.svg`) for all visualizations.
- In cases of high similarities between two reports, both reports will be discarded.