Islamic University of Technology

RELATIONAL DATABASE MANAGEMENT SYSTEM LAB

CSE 4508

Lab 2: Procedural Language/Structured Query Language

Author:
Ahmed M. S. Albreem (210041258)

September 24, 2024

Contents

1 Introduction

This report details the solutions to the tasks assigned in the CSE 4508 Relational Database Management System Lab 2. The lab focuses on procedural language and structured query language (PL/SQL) concepts in Oracle. Key topics include PL/SQL blocks, functions, procedures, and control flow constructs. The report outlines the code, explanations, and outputs for each task.

2 Task 1: Password Permutations

2.1 Code

```
DECLARE
    max_length NUMBER;
    total_permutations NUMBER;
BEGIN
    SELECT MAX(Password_Length)
    INTO max_length
    FROM your_table_name;

    total_permutations := 1;
    FOR i IN 1..max_length LOOP
        total_permutations := total_permutations * (52 - i +
   1);
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Total Permutations: ' ||
   total_permutations);
END;
/
```

2.2 Explanation

This PL/SQL block calculates the total number of possible permutations for passwords based on their length. Here's how it works:

1. Retrieve the maximum password length from the specified table and store it in the *max_length* variable.

2. Initialize *total_permutations* to 1 and loop through all password lengths to calculate the total number of possible permutations.
3. Output the result.

2.3 Output

For example, if the longest password length is 6, the output will be:
Total Permutations: 20,358,255,520,000.

3 Task 2: String Manipulation

3.1 Code

```sql
CREATE OR REPLACE PROCEDURE check_palindrome(input_string IN
    VARCHAR2)
IS
    spaced_string VARCHAR2(255);
    reversed_string VARCHAR2(255);
BEGIN
    spaced_string := '';
    FOR i IN 1..LENGTH(input_string) LOOP
        spaced_string := spaced_string || SUBSTR(input_string
    , i, 1) || ' ';
    END LOOP;

    reversed_string := REVERSE(input_string);

    IF input_string = reversed_string THEN
        DBMS_OUTPUT.PUT_LINE('Yes');
    ELSE
        DBMS_OUTPUT.PUT_LINE('No');
    END IF;

    DBMS_OUTPUT.PUT_LINE('Spaced String: ' || spaced_string);
END;
/

BEGIN
    check_palindrome('racecar');
END;
/
```

3.2 Explanation

This PL/SQL procedure performs the following tasks:

- Adds spaces between each character of the input string.
- Checks if the string is a palindrome (reads the same forwards and backwards).
- Outputs whether the string is a palindrome and displays the spaced version of the string.

3.3 Output

For the input ”racecar”, the output will be:
Yes Spaced String: r a c e c a r

4 Task 3: Library System

4.1 Code

```sql
CREATE TABLE Library_Borrowing (
    B_ID NUMBER PRIMARY KEY,
    Name VARCHAR2(255),
    Book_Title VARCHAR2(255),
    Borrow_Date DATE,
    Due_Date DATE,
    Returned_Date DATE
);

INSERT INTO Library_Borrowing (B_ID, Name, Book_Title,
   Borrow_Date, Due_Date, Returned_Date) VALUES
    (1, 'John Doe', 'The Great Gatsby', '2024-08-01', '
   2024-08-15', '2024-08-20'),
    (2, 'Jane Smith', '1984', '2024-08-10', '2024-08-24',
   NULL),
    (3, 'Alice Johnson', 'To Kill a Mockingbird', '2024-09-01
   ', '2024-09-15', '2024-09-16'),
    (4, 'Bob Brown', 'Moby Dick', '2024-08-20', '2024-09-03',
    '2024-09-02'),
    (5, 'Charlie Adams', 'The Catcher in the Rye', '
   2024-09-05', '2024-09-19', NULL);
```

```

CREATE OR REPLACE FUNCTION Calculate_Late_Fee(bid_in IN NUMBER)
RETURN NUMBER
IS
    late_days NUMBER;
    late_fee NUMBER;
BEGIN
    SELECT COALESCE(Returned_Date, SYSDATE) - Due_Date
    INTO late_days
    FROM Library_Borrowing
    WHERE B_ID = bid_in;

    late_fee := GREATEST(0, late_days);
    RETURN late_fee;
END;
/

CREATE OR REPLACE PROCEDURE List_Overdue_Books
IS
BEGIN
    DBMS_OUTPUT.PUT_LINE('Overdue Books:');
    FOR rec IN (
        SELECT Name, Book_Title, SYSDATE - Due_Date AS days_overdue
        FROM Library_Borrowing
        WHERE Returned_Date IS NULL AND Due_Date < SYSDATE
    ) LOOP
        DBMS_OUTPUT.PUT_LINE(rec.Name || ' - ' || rec.Book_Title || ' (' || rec.days_overdue || ' days overdue)');
    END LOOP;
END;
/

BEGIN
    DBMS_OUTPUT.PUT_LINE('Late Fee for B_ID 2: ' || Calculate_Late_Fee(2));
    List_Overdue_Books;
END;
/
```

4.2 Explanation

This section implements a library borrowing system:

1. **Table Creation**: A table *Library_Borrowing* stores book borrowing data.
2. **Calculate_Late_Fee Function**: This function calculates late fees for books not returned on time.
3. **List_Overdue_Books Procedure**: A procedure to list books that are overdue.

5 Task 4: Additional Procedures and Functions

5.1 Code

```
DECLARE
  current_year NUMBER := EXTRACT(YEAR FROM SYSDATE);
  is_decade_start BOOLEAN;
BEGIN
  is_decade_start := MOD(current_year, 10) = 0;

  IF is_decade_start THEN
    DBMS_OUTPUT.PUT_LINE('Yes, it is the start of a decade.')
   ;
  ELSE
    DBMS_OUTPUT.PUT_LINE('No, it is not the start of a decade
   .');
  END IF;

  -- Calculate and display the decade
  DBMS_OUTPUT.PUT_LINE('The ' || (current_year - MOD(
   current_year, 10)) || 's');
END;
/

CREATE OR REPLACE FUNCTION prime_generator(s IN NUMBER)
   RETURN VARCHAR2 IS
  prime_list VARCHAR2(1000) := '';
  num NUMBER := 2;
```

```sql
BEGIN
  WHILE num <= s LOOP
    IF MOD(num, 2) = 0 AND num != 2 THEN
      num := num + 1;
      CONTINUE;
    END IF;

    FOR i IN 2..FLOOR(SQRT(num)) LOOP
      IF MOD(num, i) = 0 THEN
        num := num + 1;
        CONTINUE;
      END IF;
    END LOOP;

    -- Add the prime number to the list
    prime_list := prime_list || num || ', ';

    num := num + 1;
  END LOOP;

  -- Return the list of primes, removing trailing comma and
   space
  RETURN RTRIM(prime_list, ', ');
END;
/

CREATE TABLE students (
  student_id NUMBER PRIMARY KEY,
  name VARCHAR2(50),
  attendance NUMBER(5,2),
  quiz NUMBER(5,2),
  mid NUMBER(5,2),
  final NUMBER(5,2)
);

INSERT INTO students VALUES (1, 'Alice', 9, 14, 23, 45);
INSERT INTO students VALUES (2, 'Bob', 8, 10, 20, 40);
INSERT INTO students VALUES (3, 'Charlie', 10, 12, 22, 48);
INSERT INTO students VALUES (4, 'David', 7, 13, 21, 42);
INSERT INTO students VALUES (5, 'Eve', 9, 15, 25, 50);

CREATE OR REPLACE PROCEDURE calculate_total_marks IS
  CURSOR scu IS
    SELECT student_id, attendance, quiz, mid, final FROM
   students;
  total_marks NUMBER;
BEGIN
  FOR student IN scu LOOP
    total_marks := student.attendance * 0.1 + student.quiz *
   0.15 + student.mid * 0.25 + student.final * 0.5;
    DBMS_OUTPUT.PUT_LINE('Total marks for student ' ||
   student.student_id || ' is: ' || total_marks);
  END LOOP;
END;
/

-- Assign grades to students based on total marks
CREATE OR REPLACE PROCEDURE assign_grades IS
  CURSOR scu IS
    SELECT student_id, attendance, quiz, mid, final FROM
   students;
  total_marks NUMBER;
  grade CHAR(1);
BEGIN
  FOR student IN scu LOOP
    total_marks := student.attendance * 0.1 + student.quiz *
   0.15 + student.mid * 0.25 + student.final * 0.5;
    IF total_marks >= 80 THEN
      grade := 'A';
    ELSIF total_marks >= 70 THEN
      grade := 'B';
    ELSIF total_marks >= 60 THEN
      grade := 'C';
    ELSIF total_marks >= 40 THEN
      grade := 'D';
    ELSE
      grade := 'F';
    END IF;
    DBMS_OUTPUT.PUT_LINE('Student ' || student.student_id ||
   ' has grade: ' || grade);
  END LOOP;
END;
/
```

5.2 Explanation

This section introduces additional PL/SQL functionalities:

- **Decade Calculation**: The block checks if the current year marks the start of a new decade and calculates the corresponding decade.

- **Prime Generator Function**: A function that generates a list of prime numbers up to a given value.
- **Student Marks and Grade Assignment**: Procedures to calculate total marks for students based on attendance, quizzes, midterms, and final exams, followed by grade assignment.

6 Conclusion

This lab provided a hands-on experience with PL/SQL, focusing on key concepts like blocks, procedures, and functions. These tasks demonstrated the creation and usage of reusable code components to handle database operations efficiently.