Islamic University of Technology

RDBMS

CSE 4508

Lab Report 6

Name : Abdullah

Student ID : 200041126

Section : 1

Lab Group : 1B

Date of Performance : 18/10/23

Date of Submission : 19/10/23

Task A

In task A, I initiated the process by establishing a database table, named "NAME_OF_TABLE," featuring two distinct columns, namely "username" and "password_length." Subsequently, I crafted a PL/SQL procedure specifically designed to identify and print the maximum password length present within the table.

Furthermore, within this procedure, I incorporated a computation to determine the number of permutations required for a potential hacker to crack the password. This computation leveraged a for loop, systematically calculating the various possible permutations for password combinations.

Table creation and PL/SQL procedure:

```sql
DROP TABLE NAME_OF_TABLE;

CREATE TABLE NAME_OF_TABLE
(
    username VARCHAR2(30),
    password_length NUMBER
);

INSERT INTO name_of_table (Username, Password_Length) VALUES ('user1', 8);
INSERT INTO name_of_table (Username, Password_Length) VALUES ('user2', 7);
INSERT INTO name_of_table (Username, Password_Length) VALUES ('user3', 9);
INSERT INTO name_of_table (Username, Password_Length) VALUES ('user4', 8);
INSERT INTO name_of_table (Username, Password_Length) VALUES ('user5', 7);
INSERT INTO name_of_table (Username, Password_Length) VALUES ('user6', 6);
INSERT INTO name_of_table (Username, Password_Length) VALUES ('user7', 8);
INSERT INTO name_of_table (Username, Password_Length) VALUES ('user8', 8);
INSERT INTO name_of_table (Username, Password_Length) VALUES ('user9', 6);
INSERT INTO name_of_table (Username, Password_Length) VALUES ('user10', 6);
```

```sql
CREATE OR REPLACE PROCEDURE find_mxpl_and_noperm
IS
mxpl NUMBER;
noac NUMBER := 52;
noperm NUMBER;
BEGIN
    SELECT MAX(PASSWORD_LENGTH)
    INTO mxpl
    FROM NAME_OF_TABLE;

    DBMS_OUTPUT.PUT_LINE('Maximum length of password in table: ' || mxpl);

    noperm := 1;

    FOR i in 1 .. mxpl
    LOOP
        noperm := noperm * noac;
        noac := noac - 1;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Number of permutions needed: ' || noperm);
END find_mxpl_and_noperm;
/
```

Result:

```
SQL> SET SERVEROUTPUT ON;
SQL>
SQL> BEGIN
  2      find_mxpl_and_noperm;
  3  END;
  4  /
Maximum length of password in table: 9
Number of permutions needed: 1335062881152000

PL/SQL procedure successfully completed.
```

Task B

In this assignment, our objective was to develop a procedure that determines both the nearest prime number greater than a given value and the nearest prime number smaller than the same value.

To accomplish this, I devised two distinct procedures. The first procedure is responsible for finding the prime number that is smaller than the provided value, while the second procedure identifies the prime number that is greater than the given number. These two procedures are then called within the main procedure to achieve the desired outcome.

Below is the implementation of the procedures :

```
CREATE OR REPLACE PROCEDURE find_greater
(gnum IN NUMBER)
IS
num NUMBER;
inum NUMBER;

is_prime BOOLEAN;
BEGIN
    num := gnum;

    WHILE (TRUE)
    LOOP
        num := num + 1;

        inum := 2;
        is_prime := TRUE;

        WHILE (inum * inum <= num)
        LOOP
            IF (MOD(num, inum) = 0) THEN
                is_prime := FALSE;
                EXIT;
            ELSE
                inum := inum + 1;
            END IF;
        END LOOP;

        IF (is_prime = TRUE) THEN
            DBMS_OUTPUT.PUT_LINE('Nearest prime number greater than ' || gnum || ': ' || num);
            EXIT;
        END IF;
    END LOOP;
END find_greater;
/
```

```
CREATE OR REPLACE PROCEDURE find_less
(gnum IN NUMBER)
IS
num NUMBER;
inum NUMBER;

is_prime BOOLEAN;
BEGIN
    num := gnum;

    IF (num < 3) THEN
        DBMS_OUTPUT.PUT_LINE('No nearest prime number less than ' || gnum);
    END IF;

    WHILE (TRUE)
    LOOP
        num := num - 1;

        inum := 2;
        is_prime := TRUE;

        WHILE (inum * inum <= num)
        LOOP
            IF (MOD(num, inum) = 0) THEN
                is_prime := FALSE;
                EXIT;
            ELSE
                inum := inum + 1;
            END IF;
        END LOOP;

        IF (is_prime = TRUE) THEN
            DBMS_OUTPUT.PUT_LINE('Nearest prime number less than ' || gnum || ': ' || num);
            EXIT;
        END IF;
    END LOOP;
END find_less;
/
```

```
CREATE OR REPLACE PROCEDURE nearest_primes
(num IN NUMBER)
IS
BEGIN
    find_less(num);
    find_greater(num);
END nearest_primes;
/
```

Result:

```
SQL> SET SERVEROUTPUT ON;
SQL>
SQL> BEGIN
  2      nearest_primes(19);
  3  END;
  4  /
Nearest prime number less than 19: 17
Nearest prime number greater than 19: 23

PL/SQL procedure successfully completed.
```

Task C

In this task, our task was to develop a procedure that accepts a string as input and displays the string with spaces inserted after each character and find out if the string is palindrome or not.

I devised two procedures for this purpose. The first one appends a space after each character in the given string, creating a new modified string with spaces in between. The second procedure is designed to identify whether the input string is a palindrome.

Below is the implementation of the procedure :

```
CREATE OR REPLACE PROCEDURE checkPalindrome
(str IN VARCHAR2)
IS
strlen NUMBER;
lim NUMBER;
is_palindrome BOOLEAN := TRUE;
BEGIN
    strlen := LENGTH(str);
    lim := TRUNC(strlen / 2);

    FOR i IN 1 .. lim
    LOOP
        IF (SUBSTR(str, i, 1) != SUBSTR(str, strlen - i + 1, 1)) THEN
            is_palindrome := FALSE;
            EXIT;
        END IF;
    END LOOP;

    IF (is_palindrome = TRUE) THEN
        DBMS_OUTPUT.PUT_LINE('Given string is palindrome: YES');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Given string is palindrome: NO');
    END IF;
END checkPalindrome;
/
```

```
CREATE OR REPLACE PROCEDURE addSpace
(str IN VARCHAR2)
IS
newstr VARCHAR2(255);
strlen NUMBER;
BEGIN
    strlen := LENGTH(str);

    FOR i IN 1 .. strlen
    LOOP
        newstr := newstr || SUBSTR(str, i, 1);
        IF (i != strlen) THEN
            newstr := newstr || ' ';
        END IF;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('New string: ' || newstr);
END addSpace;
/
```

```
CREATE OR REPLACE PROCEDURE addSpace_and_checkPalindrome
(str IN VARCHAR2)
IS
BEGIN
    addSpace(str);
    checkPalindrome(str);
END addSpace_and_checkPalindrome;
/
```

Result:

```
SQL> SET SERVEROUTPUT ON;
SQL>
SQL> BEGIN
  2      addSpace_and_checkPalindrome('racecar');
  3      addSpace_and_checkPalindrome('notracecar');
  4  END;
  5  /
New string: r a c e c a r
Given string is palindrome: YES
New string: n o t r a c e c a r
Given string is palindrome: NO

PL/SQL procedure successfully completed.
```