DSS Practical – 7

Pre Lab

Q1. Why do we need hashing algorithms for integrity checking?

Answer: A hash value is a numeric value of a fixed length that uniquely identifies data. Hash valuesare useful for verifying the integrity of data sent through insecure channels. The hash value of received data can be compared to the hash value of data as it was sent to determine whether the data was altered. In this way, we can maintain integrity of data by using hashing.

Q2. How can credentials be validated using hashing?

Answer: When users create a new account and input their chosen password, the application code passes that password through a hashing function and stores the result in the database. When the user wants to authenticate later, the process is repeated and the result is compared to the value fromthe database. If it's a match, the user provided the right password.

Q3. What is SELECT INTO in PL/SQL?

Answer: The SELECT INTO is actually a standard SQL query where the SELECT INTO clauseis used to place the returned data into predefined variables. The SELECT INTO clause of SQL is used to retrieve one row or set of columns from the Oracle database.

Q4. How do you execute a PL/SQL procedure?

Answer: A PL/SQL procedure is a reusable unit that encapsulates specific business logic of the application. Below are the syntax/commands to be given to execute a PL/SQL procedure- *EXECUTE procedure_name(arguments);* **or** *EXEC procedure_name(arguments);*

IN – LAB

Q1. Drop the table ‘app_users’ created in the previous lab and create a table ‘app_users’ with the following columns: id, username, password. ‘id’ is the primary key and ‘username’ is unique.

```
Run SQL Command Line

SQL*Plus: Release 11.2.0.2.0 Production on Mon Apr 18 10:09:57 2022

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

SQL> CONNECT System/root;
Connected.
SQL> DROP TABLE app_users;

Table dropped.

SQL>
```

```
SQL> DROP TABLE app_users;

Table dropped.

SQL> CREATE TABLE app_users(id NUMBER(10) PRIMARY KEY, username VARCHAR2(30) UNIQUE, password VARCHAR2(30) NOT NULL);

Table created.

SQL>
```

Q2. Drop the sequence created in the previous lab and create a sequence ‘app_users_seq’.

```
SQL> DROP SEQUENCE app_users_sequence;

Sequence dropped.

SQL> CREATE SEQUENCE app_users_seq START WITH 3001;

Sequence created.

SQL>
```

Q3. Similar to the previous lab, grant execute on dbms_crypto package to the user to be ableto use that to hash.

```
SQL> CONNECT System/root as SYSDBA;
Connected.
SQL> GRANT EXECUTE ON sys.dbms_crypto to System;

Grant succeeded.

SQL> _
```

Q4. Similar to the previous lab, create a function ‘get_hash’ in PL/SQL to hash the given username and password usingSH1 algorithm.

```
SQL> ed
Wrote file afiedt.buf

  1  CREATE OR REPLACE FUNCTION get_hash(p_username IN VARCHAR2, p_password IN VARCHAR2)
  2      RETURN VARCHAR2 AS
  3      l_salt VARCHAR2(30) := 'mysaltvalue';
  4    BEGIN
  5       RETURN DBMS_CRYPTO.HASH(UTL_RAW.CAST_TO_RAW(UPPER(p_username) || l_salt || UPPER(p_password)),DBMS_CRYPTO.HASH_SH1);
  6*   END;
SQL> /

Function created.

SQL>
```

Q5. Similar to the previous lab, create a procedure ‘add_user’ which executes the ‘get_hash’ function and stores the given username and hashed password in the table ‘app_users’

Q6. Execute the function ‘add_user’ with inputs as ‘labtest’ (for username) and ‘labtest1’(for password) and display the table ‘app_users’ from

```
Run SQL Command Line
SQL> /

Procedure created.

SQL> execute add_user('labtest','labtest1');

PL/SQL procedure successfully completed.

SQL>
```

```
Run SQL Command Line
SP2-0734: unknown command beginning "2.select *..." - rest of line ignored.
SQL> select * from app_users1;

        ID USERNAME
---------- ------------------------------
PASSWORD
----------------------------------------
      3001 LABTEST
4FA42E2CBA36BCD69EEAAB5395A2433545CF3199


SQL>
```

Q7. Create a procedure ‘valid_user’ that will check if the username and password given asinput by user is valid or not by checking with the previously stored values in the ‘app_users’ table and validating.

```
Run SQL Command Line


SQL> ed
Wrote file afiedt.buf

  1  CREATE OR REPLACE PROCEDURE valid_user (p_username  IN  VARCHAR2,
  2                        p_password  IN  VARCHAR2) AS
  3                    v_dummy  VARCHAR2(1);
  4      BEGIN
  5                    SELECT '1'
  6                    INTO   v_dummy
  7                    FROM   app_users1
  8                    WHERE  username = UPPER(p_username)
  9                    AND    password = get_hash(p_username, p_password);
 10  dbms_output.put_line('Valid user');
 11      EXCEPTION
 12            WHEN NO_DATA_FOUND THEN
 13            RAISE_APPLICATION_ERROR(-20000, 'Invalid username/password.');
 14*    END;
SQL> /

Procedure created.

SQL>
```

```
Run SQL Command Line

SQL> set serveroutput on
SQL> EXECUTE valid_user('labtest', 'labtest1');
Valid user

PL/SQL procedure successfully completed.

SQL>
```

Since the username and password we gave matches the original inputs before hashing, we get

```
Run SQL Command Line

SQL> EXECUTE valid_user('labtest1', 'labtest2');
BEGIN valid_user('labtest1', 'labtest2'); END;

*
ERROR at line 1:
ORA-20000: Invalid username/password.
ORA-06512: at "SYS.VALID_USER", line 13
ORA-06512: at line 1


SQL>
```

theoutput as ‘Valid User’.

Since the username and password we gave does not match the original inputs before hashing, weget the output as 'Invalid username/password.'

Post-Lab:

Q1. Write the differences between a procedure and a function in PL/SQL.

PROCEDURE	FUNCTION
Used mainly to execute certain business logic with DML and DRL statements	Used mainly to perform some computational process and returning the result of that process.
Procedure cannot call with select statement, but can call from a block or from a procedure	Function can call with select statement , if function does not contain any DML statements and DDL statements.
Procedure can return zero or more values as output.	Function can return only single value as output.
It is not mandatory to return the value.	It is mandatory to return the value.
RETURN will simply exit the control from subprogram.	RETURN will exit the control from subprogram and also returns the value.

Q2. Create a procedure to protect access to the hrview_role

Answer:

```
CREATE OR REPLACE PROCEDURE appsec.p_check_hrview_access AUTHID CURRENT_USER
AS BEGIN
IF( ( SYS_CONTEXT( 'USERENV', 'IP_ADDRESS' ) LIKE '192.168.%'
) AND
TO_CHAR( SYSDATE, 'HH24' ) BETWEEN 9 AND
) THEN
EXECUTE IMMEDIATE 'SET ROLE hrview_role'; END IF;
END;
/
```

Here, we are prohibiting clients with IP address other than those whose IP begins with the string "192.168" to access the data. The LIKE syntax indicates that the address should consist of the specified characters followed by zero or more characters. And also, we setting up a constraint that only during our normal office hours of 9 AM to 5 PM, the clients can access the data.

Q3. Create a role named hrview_role. Through that role we will grant access to the data needed by a variety of applications that we plan to build. At the outset, we only want folks who are on our internal network to access this data, and only during our normal office hours of 7 AM to 7 PM

Answer: CREATE ROLE hrview_role IDENTIFIED USING appsec.p_check_hrview_access;

CREATE OR REPLACE PROCEDURE appsec.p_check_hrview_access AUTHID CURRENT_USER

AS BEGIN

IF((SYS_CONTEXT('USERENV', 'IP_ADDRESS') LIKE '192.168.%' OR SYS_CONTEXT('USERENV', 'IP_ADDRESS') = '127.0.0.1')

AND

TO_CHAR(SYSDATE, 'HH24') BETWEEN 7 AND 19

) THEN

EXECUTE IMMEDIATE 'SET ROLE hrview_role'; END IF;

END;

/