Name :- Rushikesh Kashinath Palve
Roll No. - 31258

Date :	Page No :
/ /20	

Assignment No. 5

DOP :- 15-09-2021 DOS :- 18-09-2021

Title :- Write a PL/SQL stored procedure and function.

Problem Definition :-

Write a stored procedure namely pro_Grade for the categorization of student. If marks stored by students in examination is <= 1500 and marks >= 990 then student will be placed in distinction category if marks stored are between 989 and 900 category is First class, if marks 899 and 825 category is Higher second class. Write a PL/SQL block to use procedure created with above requirement.

Stud_Marks (name, total_marks)
Result (Roll, Name, class)

Objectives :-

(i) Understand IF-THEN condition and FOR loop.
(ii) Understand the PL/SQL stored procedure.
(iii) Understand the PL/SQL stored Function.
(iv) Write PL/SQL block code using stored procedure and stored Function.

Learning Outcomes :-

Students will be able to -
(i) Understand concepts of IF-THEN condition, FOR loop, stored Procedure, stored Function.

(ii) Write PL/SQL block code.

THEORY :-

PROCEDURE :-

A procedure is a subprogeam that perforems a speriffic action or task. A proceduze has two pactu.

(i) The proceduze speriffiration :- the proceduze speriffiration speriffies the proceduze name and the parameteus it accepts. It is not neressazy to ceeate a proceduze that accepts parameteus.

(2) The proceduze Body :- The proceduze body contains the deriarative sertion without DECLARE keyword, the exerutable sertion and an exreption vertion.

Syntax foz ceeating a peoceduze :-

```
Create [oz ceeplare] PROCEDURE peoceduze_name
[(argument 1 [IN/OUT/IN OUT] type),
 (argument 2 [IN/OUT/IN OUT] type),
 ...]
IS/AS
peoceduze_body;
```

Where,

peoceduze_name :- is the name of the peoceduze to be ceeated.

Argument :- is the name of the peoceduze paxameteu

Type :- Is the data type of the associated paxameteu

peoceduze_body :- Is a PL/SQL block that makes up the code of the peoceduze.

IN :- This is default mode. The value of the actual parameter is passed into the procedure. Inside the procedure the formal parameter is considered read only.

OUT :- Any value the actual parameter has when the procedure is called ignored. Inside the procedure, the formal parameters are considered as write only.

IN OUT :- this mode is combination of IN and OUT.

Deleting procedure :- To remove a procedure from the database.

Syntax :-

```
drop procedure <procedure_name>;
```

FUNCTION :-

A function is a subprogram, which is used to compute values. It is similar to a procedure, function also takes arguments and can be in different modes. Function also can be stored in the database. It is a PL/SQL block consisting of declarative, executable and exception section.

Difference between procedure and function is that the procedure call is a PL/SQL statement by itself, while a function call is called as a part of an expression.

A function can return more than one value using OUT parameters.

A function can be called using positional or named notation.

Syntax for creating a function :-

```
Create [or replace] FUNCTION function_name
[(argument1 [IN/OUT/ IN OUT] type),
 (argument2 [IN/OUT/ IN OUT] type),
 ...]
Return return_type IS/AS
Function_body
```

Where

Function_name :- Is the name of the function to be created

Argument :- Is the name of the function parameter

Type :- Is the data type of the associated parameter -

Function_body :- Is a PL/SQL block containing code for the function.

IN :- This is default mode. The value of the actual parameter is passed into the procedure. Inside the procedure the formal parameter is considered read only.

OUT :- Any value the actual parameter has when the procedure is called ignored. Inside the procedure, the formal parameters are considered as write only.

IN OUT :- this mode is combination of IN and OUT.

Deleting a Function :- To remove the subprogram from the database.

Syntax:-

```
Drop function <function_name>;
```

Parkage :-

A parkage is a PL/SQL construct that allows related objects to be stored together. A parkage has 2 separate parts :- the sperifiration and the body. Each of them stored separately in the data dirtionary.

Parkage sperifiration :-

```
CREATE (OR REPLACE) PACKAGE parkage_name
{
  IS/AS}
  type_definition |
  peoreduce_sperifiration | Funrtion sperifiration |
  variable_dertaration |
  exreption_dertaration |
  rursor_dertaration |
  pragma dertaration |
  end [peocedure_name];
```

Parkage Body :-

The parkage bode is separate data dirtionary object from the parkage header. It rannot be surressfully rompiled unless the parkage header has already been successfully compiled

syntax :-

```
CREATE OR REPLACE PACKAGE BODY parkage_nameAS
Proreduce definition ;
Function definition ;
....
End parkage_name
```

To drop the package (both specification & the body) use the drop package command as follows :-

Syntax :-

Drop package <package_name> ;

Conclusion :-

We successfully understood concepts of IF-THEN condition, FOR loop, stored procedure and stored function. Also, wrote PL/SQL block code using stored procedure and stored function.

OUTPUT :-

```
/*
  Problem statement :-
     Write a Stored Procedure namely proc_Grade for the categorization of
student.
        If marks scored by students in examination is <=1500 and marks>=990
then
        student will be placed in distinction category,
        if marks scored are between 989 and 900 category is first class,
        if marks 899 and 825 category is Higher Second Class.
        Write a PL/SQL block to use procedure created with above requirement.
        Stud_Marks(name, total_marks)
        Result(Roll,Name, Class)
*/


CREATE TABLE Stud_Marks(
  RollNo INT PRIMARY KEY,
    Sname VARCHAR(20),
    Total_Marks INT
    );


INSERT INTO Stud_Marks VALUES
  (1, 'Vidyut', 995),
  (2, 'Pratap', 828),
  (3, 'Kailash', 945),
  (4, 'Mukund', 1500),
  (5, 'Girish', 900),
  (6, 'Neeraj', 850),
  (7, 'Prashant', 800),
  (8, 'Raj', 899),
  (9, 'Hari', 1300),
  (10, 'Aditya', 920);


CREATE TABLE Result(
  RollNo INT REFERENCES Stud_Marks(RollNo),
    Sname VARCHAR(20),
    Marks INT,
    Grade VARCHAR(20)
    );


-- Procedure with Parameters

mysql> DELIMITER $$
mysql> CREATE PROCEDURE proc_Grade(IN roll INT, IN marks INT)
    -> BEGIN
    -> DECLARE student VARCHAR(20);
    -> DECLARE EXIT HANDLER FOR SQLEXCEPTION SELECT 'ENTRY NOT FOUND' AS
EXCEPTION;
```

```
    -> IF NOT EXISTS(SELECT * FROM  Stud_Marks WHERE RollNo = roll AND
Total_Marks = marks) THEN
    -> SIGNAL SQLSTATE '45000';
    -> END IF;
    -> SELECT Sname INTO student FROM Stud_Marks WHERE RollNo = roll AND
Total_Marks = marks;
    -> IF marks >= 990 AND marks <= 1500 THEN
    -> INSERT INTO Result VALUES(roll, student, marks, "Distinction");
    -> ELSEIF marks >= 900 AND marks <= 989 THEN
    -> INSERT INTO Result VALUES(roll, student, marks, "First Class");
    -> ELSEIF marks >= 825 AND marks <= 899 THEN
    -> INSERT INTO Result VALUES(roll, student, marks, "Higher Second Class");
    -> ELSE
    -> INSERT INTO Result VALUES(roll, student, marks, NULL);
    -> END IF;
    -> SELECT * FROM Result;
    -> END $$


DELIMITER ;
CALL proc_Grade(1, 995);
CALL proc_Grade(2, 828);
CALL proc_Grade(3, 945);
CALL proc_Grade(4, 1500);
CALL proc_Grade(5, 900);
CALL proc_Grade(6, 850);
CALL proc_Grade(7, 800);
CALL proc_Grade(8, 899);
CALL proc_Grade(9, 1300);
CALL proc_Grade(10, 920);


mysql> SELECT * FROM Result;
```

RollNo	Sname	Marks	Grade
1	Vidyut	995	Distinction
2	Pratap	828	Higher Second Class
3	Kailash	945	First Class
4	Mukund	1500	Distinction
5	Girish	900	First Class
6	Neeraj	850	Higher Second Class
7	Prashant	800	NULL
8	Raj	899	Higher Second Class
9	Hari	1300	Distinction
10	Aditya	920	First Class

```
mysql> CALL proc_Grade(11, 828);
+------------------+
| EXCEPTION        |
+------------------+
| ENTRY NOT FOUND |
+------------------+
```

```
-- Function with Parameter

mysql> DELIMITER $$
mysql> CREATE FUNCTION fun_Grade(marks INT)
    -> RETURNS VARCHAR(20)
    -> DETERMINISTIC
    -> BEGIN
    -> DECLARE grade VARCHAR(20);
    -> IF marks >= 990 AND marks <= 1500 THEN
    -> SET grade = 'Distinction';
    -> ELSEIF marks >= 900 AND marks <= 989 THEN
    -> SET grade = 'First Class';
    -> ELSEIF marks >= 825 AND marks <= 899 THEN
    -> SET grade = 'Higher Second Class';
    -> ELSE
    -> SET grade = NULL;
    -> END IF;
    -> RETURN (grade);
    -> END$$

DELIMITER ;

mysql> SELECT RollNo, Sname, Total_Marks, fun_Grade(Total_Marks) AS GRADE
FROM Stud_marks ORDER BY RollNo;
```

RollNo	Sname	Total_Marks	GRADE
1	Vidyut	995	Distinction
2	Pratap	828	Higher Second Class
3	Kailash	945	First Class
4	Mukund	1500	Distinction
5	Girish	900	First Class
6	Neeraj	850	Higher Second Class
7	Prashant	800	NULL
8	Raj	899	Higher Second Class
9	Hari	1300	Distinction
10	Aditya	920	First Class

```
CREATE TABLE Result2(
  RollNo INT REFERENCES Stud_Marks(RollNo),
    Sname VARCHAR(20),
    Marks INT,
    Grade VARCHAR(20)
    );

-- -- Procedure without Parameters

mysql> DELIMITER $$
mysql> CREATE PROCEDURE NewGrade_Proc()
    -> BEGIN
    -> DECLARE roll INT;
    -> DECLARE m INT;
    -> DECLARE student VARCHAR(20);
```

```
-> DECLARE exit_loop BOOLEAN;
-> DECLARE C1 CURSOR FOR SELECT * FROM Stud_Marks;
-> DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit_loop = TRUE;
-> OPEN C1;
-> L1: LOOP
-> FETCH C1 INTO roll, student, m;
-> IF exit_loop THEN
-> CLOSE c1;
-> LEAVE L1;
-> END IF;
-> IF m >= 990 AND m <= 1500 THEN
-> INSERT INTO Result2 VALUES(roll, student, m, "Distinction");
-> ELSEIF m >= 900 AND m <= 989 THEN
    -> INSERT INTO Result2 VALUES(roll, student, m, "First Class");
    -> ELSEIF m >= 825 AND m <= 899 THEN
    -> INSERT INTO Result2 VALUES(roll, student, m, "Higher Second Class");
    -> ELSE
    -> INSERT INTO Result2 VALUES(roll, student, m, NULL);
    -> END IF;
    -> END LOOP L1;
    -> END $$

delimiter ;

mysql> CALL NewGrade_Proc();

mysql> SELECT * FROM Result2;
```

```
mysql> SELECT * FROM Result2;
+--------+----------+-------+---------------------+
| RollNo | Sname    | Marks | Grade               |
+--------+----------+-------+---------------------+
|      1 | Vidyut   |   995 | Distinction         |
|      2 | Pratap   |   828 | Higher Second Class |
|      3 | Kailash  |   945 | First Class         |
|      4 | Mukund   |  1500 | Distinction         |
|      5 | Girish   |   900 | First Class         |
|      6 | Neeraj   |   850 | Higher Second Class |
|      7 | Prashant |   800 | NULL                |
|      8 | Raj      |   899 | Higher Second Class |
|      9 | Hari     |  1300 | Distinction         |
|     10 | Aditya   |   920 | First Class         |
+--------+----------+-------+---------------------+
```