MIT World Peace University

Database Management Systems
Second Year B. Tech, Semester 4

Stored Procedures and Functions in PL/SQL

Assignment No. 6

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

April 30, 2023

Contents

1 Aim

Write PLSQL Procedures and Function for given problem statements

2 Objectives

1. To study PLSQL procedures and functions

3 Problem Statement

Create tables and solve given queries using Subqueries

4 Theory

4.1 PL/SQL

PL/SQL is Oracle's procedural extension to industry-standard SQL. PL/SQL naturally, efficiently, and safely extends SQL for developers. Its primary strength is in providing a server-side, stored procedural language that is easy-to-use, seamless with SQL, robust, portable, and secure.

4.2 Stored Procedures

A stored procedure is a subroutine available to applications that access a relational database system. Stored procedures (sometimes called a proc, sproc, StoPro, or SP) are actually stored in the database data dictionary.

4.3 Functions

A function is a subroutine available to applications that access a relational database management system (RDBMS). Such applications can include multiple programming languages, APIs, and communication protocols. Functions are also called procedures, modules, or subroutines. Functions are stored in and callable from the database.

4.4 Difference between Stored Procedures and Functions

The following are the key differences between a stored procedure and a function.

1. A function must return a value but in Stored Procedure it is optional.
2. A function can have only input parameters for it whereas a stored procedure can have input/output parameters .
3. Functions can be called from Procedure whereas Procedures cannot be called from a Function.
4. Exception can be handled by try-catch block in a Procedure whereas try-catch block cannot be used in a Function.
5. We can go for Transaction Management in Procedure whereas we can't go in Function.
6. We can use a procedure in a select statement but we can't use Function in a select statement.

7. We can't use a function in DML (insert, update, delete) statement. But we can use a procedure in DML statement.

5 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Draw.io for Drawing the ER diagram.

6 Input

Given Database from the Problem Statement for the Assignment for our batch. (A1 PA 20)

7 Creation and Insertion of Values in the Tables

8 Queries

```
-- Procedures and Functions

-- BATCH 2 EXERCISE 1 - PROCEDURES

-- product(prod_id, prod_name, qty_on_hand)
-- Order(cust_id, prod_id, order_date, qty_ordered)
-- Customer(cust_id, cust_name, phone, address)

-- Write a stored procedure to take the cust_id, prod_id and qty_ordered as
-- input. Procedure should check if the order for a particular customer can be
-- fulfilled and if yes then insert the new order and update the product
-- quantity on hand. Display appropriate message if the order cannot be
-- fulfilled. Output parameter must have updated value of the qty_on_hand

-- 1. Create database and tables

create database if not exists lab_procedures;
use lab_procedures;

CREATE TABLE `product` (
  `product_id` INT NOT NULL AUTO_INCREMENT,
  `prod_name` varchar(255) NOT NULL,
  `qty_on_hand` INT(255) NOT NULL,
  PRIMARY KEY (`product_id`)
);

CREATE TABLE `customer` (
  `cust_id` INT NOT NULL AUTO_INCREMENT,
  `cust_name` varchar(255) NOT NULL,
  `phone` VARCHAR(255) NOT NULL,
  `address` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`cust_id`)
);

CREATE TABLE `order_details` (
  `cust_id` INT NOT NULL,
```

```
  `product_id` INT NOT NULL,
  `order_date` DATE NOT NULL,
  `qty_order` INT NOT NULL
);

INSERT INTO product (prod_name, qty_on_hand) VALUES
    ('Product A', 50),
    ('Product B', 100),
    ('Product C', 75);

INSERT INTO customer (cust_name, phone, address) VALUES
    ('John Smith', '123-456-7890', '123 Main St'),
    ('Jane Doe', '555-555-1212', '456 Oak Ave'),
    ('Bob Johnson', '555-123-4567', '789 Elm St');

INSERT INTO order_details values
    (1, 1, '2023-04-25', 10),
    (1, 2, '2023-04-26', 5),
    (2, 3, '2023-04-27', 8),
    (3, 1, '2023-04-26', 15),
    (3, 3, '2023-04-27', 3);


ALTER TABLE `order_details` ADD CONSTRAINT `order_fk0` FOREIGN KEY (`cust_id`)
    REFERENCES `customer`(`cust_id`);

ALTER TABLE `order_details` ADD CONSTRAINT `order_fk1` FOREIGN KEY (`product_id`)
    REFERENCES `product`(`product_id`);


-- Creating Procedure

DELIMITER $$
CREATE PROCEDURE Fulfill_Order_proc3 (
    IN p_cust_id INT,
    IN p_prod_id INT,
    IN p_qty_ordered INT,
    OUT p_qty_on_hand INT
)


BEGIN
    DECLARE v_qty_on_hand INT;

    -- Get the current quantity on hand for the product
    SELECT qty_on_hand INTO v_qty_on_hand
    FROM product
    WHERE product_id = p_prod_id;

    -- Check if the order can be fulfilled
    IF v_qty_on_hand >= p_qty_ordered THEN
        -- Insert the new order
        INSERT INTO order_details (cust_id, product_id, order_date, qty_order)
        VALUES (p_cust_id, p_prod_id, CURDATE(), qty_order);

        -- Update the quantity on hand for the product
        UPDATE product
        SET qty_on_hand = qty_on_hand - p_qty_ordered
        WHERE product_id = p_prod_id;
```

```
94
95          -- Set the output parameter to the updated quantity on hand
96          SELECT qty_on_hand INTO p_qty_on_hand
97          FROM product
98          WHERE product_id = p_prod_id;
99
100         -- Display a success message
101         SELECT CONCAT('Order fulfilled. New quantity on hand for product ',
        p_prod_id, ' is ', p_qty_on_hand) AS message;
102     ELSE
103         -- Display an error message
104         SELECT CONCAT('Order cannot be fulfilled. Only ', v_qty_on_hand, ' units
        of product ', p_prod_id, ' are available.') AS message;
105     END IF;
106 END$$
107 DELIMITER ;
108
109
110 -- Calling Procedure
111
112 -- Get the current quantity on hand for product 1
113 SELECT qty_on_hand FROM product WHERE product_id = 1;
114
115 -- Attempt to place an order for 20 units of product 1 for customer 1
116 CALL Fulfill_Order_proc3(1, 1, 20, @qty_on_hand);
117
118 -- Get the error message returned by the stored procedure
119 SELECT message FROM (SELECT @p_qty_on_hand AS message) AS result;
120
121 -- BATCH 2 EXERCISE 2 - FUNCTIONS
122
123 -- Write a function to find total quantity ordered by taking
124 -- cust_id and prod_id as input parameter
125 -- Also write a code to call the function
126
127 -- Creating Function
128
129 DELIMITER $$
130 CREATE FUNCTION Total_Qty_Ordered2(cust_id INT, prod_id INT)
131 RETURNS INT deterministic
132 BEGIN
133     DECLARE total_qty INT;
134     SELECT SUM(qty_order) INTO total_qty
135     FROM order_details
136     WHERE cust_id = cust_id AND prod_id = product_id;
137     RETURN total_qty;
138 END $$
139
140 DELIMITER ;
141
142 -- Calling Function
143
144 SELECT Total_Qty_Ordered2(1, 1) AS total_qty;
```

9 Outputs

```
1 MariaDB [(none)]> -- Procedures and Functions
2 MariaDB [(none)]>
```

```
MariaDB [(none)]> -- BATCH 2 EXERCISE 1 - PROCEDURES
MariaDB [(none)]>
MariaDB [(none)]> -- product(prod_id, prod_name, qty_on_hand)
MariaDB [(none)]> -- Order(cust_id, prod_id, order_date, qty_ordered)
MariaDB [(none)]> -- Customer(cust_id, cust_name, phone, address)
MariaDB [(none)]>
MariaDB [(none)]> -- Write a stored procedure to take the cust_id, prod_id and
    qty_ordered as
MariaDB [(none)]> -- input. Procedure should check if the order for a particular
    customer can be
MariaDB [(none)]> -- fulfilled and if yes then insert the new order and update the
     product
MariaDB [(none)]> -- quantity on hand. Display appropriate message if the order
    cannot be
MariaDB [(none)]> -- fulfilled. Output parameter must have updated value of the
    qty_on_hand
MariaDB [(none)]>
MariaDB [(none)]> -- 1. Create database and tables
MariaDB [(none)]>
MariaDB [(none)]> create database if not exists lab_procedures;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> use lab_procedures;
Database changed
MariaDB [lab_procedures]>
MariaDB [lab_procedures]> CREATE TABLE `product` (
    -> `product_id` INT NOT NULL AUTO_INCREMENT,
    -> `prod_name` varchar(255) NOT NULL,
    -> `qty_on_hand` INT(255) NOT NULL,
    -> PRIMARY KEY (`product_id`)
    -> );
Query OK, 0 rows affected (0.004 sec)

MariaDB [lab_procedures]>
MariaDB [lab_procedures]> CREATE TABLE `customer` (
    -> `cust_id` INT NOT NULL AUTO_INCREMENT,
    -> `cust_name` varchar(255) NOT NULL,
    -> `phone` VARCHAR(255) NOT NULL,
    -> `address` VARCHAR(255) NOT NULL,
    -> PRIMARY KEY (`cust_id`)
    -> );
Query OK, 0 rows affected (0.004 sec)

MariaDB [lab_procedures]>
MariaDB [lab_procedures]> CREATE TABLE `order_details` (
    -> `cust_id` INT NOT NULL,
    -> `product_id` INT NOT NULL,
    -> `order_date` DATE NOT NULL,
    -> `qty_order` INT NOT NULL
    -> );
Query OK, 0 rows affected (0.003 sec)

MariaDB [lab_procedures]>
MariaDB [lab_procedures]> INSERT INTO product (prod_name, qty_on_hand) VALUES
    ->     ('Product A', 50),
    ->     ('Product B', 100),
    ->     ('Product C', 75);
Query OK, 3 rows affected (0.001 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
57
58 MariaDB [lab_procedures]>
59 MariaDB [lab_procedures]> INSERT INTO customer (cust_name, phone, address) VALUES
60     ->     ('John Smith', '123-456-7890', '123 Main St'),
61     ->     ('Jane Doe', '555-555-1212', '456 Oak Ave'),
62     ->     ('Bob Johnson', '555-123-4567', '789 Elm St');
63 Query OK, 3 rows affected (0.001 sec)
64 Records: 3  Duplicates: 0  Warnings: 0
65
66 MariaDB [lab_procedures]>
67 MariaDB [lab_procedures]> INSERT INTO order_details values
68     ->     (1, 1, '2023-04-25', 10),
69     ->     (1, 2, '2023-04-26', 5),
70     ->     (2, 3, '2023-04-27', 8),
71     ->     (3, 1, '2023-04-26', 15),
72     ->     (3, 3, '2023-04-27', 3);
73 Query OK, 5 rows affected (0.001 sec)
74 Records: 5  Duplicates: 0  Warnings: 0
75
76 MariaDB [lab_procedures]>
77 MariaDB [lab_procedures]>
78 MariaDB [lab_procedures]> ALTER TABLE `order_details` ADD CONSTRAINT `order_fk0`
    FOREIGN KEY (`cust_id`) REFERENCES `customer`(`cust_id`);
79 Query OK, 5 rows affected (0.011 sec)
80 Records: 5  Duplicates: 0  Warnings: 0
81
82 MariaDB [lab_procedures]>
83 MariaDB [lab_procedures]> ALTER TABLE `order_details` ADD CONSTRAINT `order_fk1`
    FOREIGN KEY (`product_id`) REFERENCES `product`(`product_id`);
84 Query OK, 5 rows affected (0.011 sec)
85 Records: 5  Duplicates: 0  Warnings: 0
86
87 MariaDB [lab_procedures]>
88 MariaDB [lab_procedures]>
89 MariaDB [lab_procedures]> -- Creating Procedure
90 MariaDB [lab_procedures]>
91 MariaDB [lab_procedures]> DELIMITER $$
92 MariaDB [lab_procedures]> CREATE PROCEDURE Fulfill_Order_proc3 (
93     ->     IN p_cust_id INT,
94     ->     IN p_prod_id INT,
95     ->     IN p_qty_ordered INT,
96     ->     OUT p_qty_on_hand INT
97     -> )
98     ->
99     ->
100    -> BEGIN
101    ->     DECLARE v_qty_on_hand INT;
102    ->
103    ->     -- Get the current quantity on hand for the product
104    ->     SELECT qty_on_hand INTO v_qty_on_hand
105    ->     FROM product
106    ->     WHERE product_id = p_prod_id;
107    ->
108    ->     -- Check if the order can be fulfilled
109    ->     IF v_qty_on_hand >= p_qty_ordered THEN
110    ->         -- Insert the new order
111    ->         INSERT INTO order_details (cust_id, product_id, order_date,
    qty_order)
112    ->         VALUES (p_cust_id, p_prod_id, CURDATE(), qty_order);
```

```
    ->
    ->         -- Update the quantity on hand for the product
    ->         UPDATE product
    ->         SET qty_on_hand = qty_on_hand - p_qty_ordered
    ->         WHERE product_id = p_prod_id;
    ->
    ->         -- Set the output parameter to the updated quantity on hand
    ->         SELECT qty_on_hand INTO p_qty_on_hand
    ->         FROM product
    ->         WHERE product_id = p_prod_id;
    ->
    ->         -- Display a success message
    ->         SELECT CONCAT('Order fulfilled. New quantity on hand for product ',
     p_prod_id, ' is ', p_qty_on_hand) AS message;
    ->     ELSE
    ->         -- Display an error message
    ->         SELECT CONCAT('Order cannot be fulfilled. Only ', v_qty_on_hand, '
    units of product ', p_prod_id, ' are available.') AS message;
    ->     END IF;
    -> END$$
Query OK, 0 rows affected (0.001 sec)

MariaDB [lab_procedures]> DELIMITER ;
MariaDB [lab_procedures]>
MariaDB [lab_procedures]>
MariaDB [lab_procedures]> -- Calling Procedure
MariaDB [lab_procedures]>
MariaDB [lab_procedures]> -- Get the current quantity on hand for product 1
MariaDB [lab_procedures]> SELECT qty_on_hand FROM product WHERE product_id = 1;
+-------------+
| qty_on_hand |
+-------------+
|          50 |
+-------------+
1 row in set (0.000 sec)

MariaDB [lab_procedures]>
MariaDB [lab_procedures]> -- Attempt to place an order for 20 units of product 1
    for customer 1
MariaDB [lab_procedures]> CALL Fulfill_Order_proc3(1, 1, 20, @qty_on_hand);
+-------------------------------------------------------------+
| message                                                     |
+-------------------------------------------------------------+
| Order fulfilled. New quantity on hand for product 1 is 30 |
+-------------------------------------------------------------+
1 row in set (0.002 sec)

Query OK, 4 rows affected (0.002 sec)

MariaDB [lab_procedures]>
MariaDB [lab_procedures]> -- Get the error message returned by the stored
    procedure
MariaDB [lab_procedures]> SELECT message FROM (SELECT @p_qty_on_hand AS message)
    AS result;
+---------+
| message |
+---------+
| NULL    |
+---------+
```

```
1 row in set (0.000 sec)

MariaDB [lab_procedures]>
MariaDB [lab_procedures]> -- BATCH 2 EXERCISE 2 - FUNCTIONS
MariaDB [lab_procedures]>
MariaDB [lab_procedures]> -- Write a function to find total quantity ordered by
    taking
MariaDB [lab_procedures]> -- cust_id and prod_id as input parameter
MariaDB [lab_procedures]> -- Also write a code to call the function
MariaDB [lab_procedures]>
MariaDB [lab_procedures]> -- Creating Function
MariaDB [lab_procedures]>
MariaDB [lab_procedures]> DELIMITER $$
MariaDB [lab_procedures]> CREATE FUNCTION Total_Qty_Ordered2(cust_id INT, prod_id
    INT)
    -> RETURNS INT deterministic
    -> BEGIN
    ->     DECLARE total_qty INT;
    ->     SELECT SUM(qty_order) INTO total_qty
    ->     FROM order_details
    ->     WHERE cust_id = cust_id AND prod_id = product_id;
    ->     RETURN total_qty;
    -> END $$
Query OK, 0 rows affected (0.003 sec)

MariaDB [lab_procedures]>
MariaDB [lab_procedures]> DELIMITER ;
MariaDB [lab_procedures]>
MariaDB [lab_procedures]> -- Calling Function
MariaDB [lab_procedures]>
MariaDB [lab_procedures]> SELECT Total_Qty_Ordered2(1, 1) AS total_qty;
+-----------+
| total_qty |
+-----------+
|        25 |
+-----------+
1 row in set (0.001 sec)

MariaDB [lab_procedures]> select * from product;
+------------+-----------+-------------+
| product_id | prod_name | qty_on_hand |
+------------+-----------+-------------+
|          1 | Product A |          30 |
|          2 | Product B |         100 |
|          3 | Product C |          75 |
+------------+-----------+-------------+
3 rows in set (0.001 sec)

MariaDB [lab_procedures]> select * from customer;
+---------+-------------+--------------+-------------+
| cust_id | cust_name   | phone        | address     |
+---------+-------------+--------------+-------------+
|       1 | John Smith  | 123-456-7890 | 123 Main St |
|       2 | Jane Doe    | 555-555-1212 | 456 Oak Ave |
|       3 | Bob Johnson | 555-123-4567 | 789 Elm St  |
+---------+-------------+--------------+-------------+
3 rows in set (0.000 sec)

MariaDB [lab_procedures]> select * from order_details;
```

```
+---------+------------+------------+-----------+
| cust_id | product_id | order_date | qty_order |
+---------+------------+------------+-----------+
|       1 |          1 | 2023-04-25 |        10 |
|       1 |          2 | 2023-04-26 |         5 |
|       2 |          3 | 2023-04-27 |         8 |
|       3 |          1 | 2023-04-26 |        15 |
|       3 |          3 | 2023-04-27 |         3 |
|       1 |          1 | 2023-04-27 |         0 |
+---------+------------+------------+-----------+
6 rows in set (0.001 sec)

MariaDB [lab_procedures]>
```

10 Conclusion

Thus, we have learned PLSQL Database Programming.

11 FAQ

1. **What is PLSQL? What are Applications of PLSQL?**

   PL/SQL (Procedural Language/Structured Query Language) is a procedural extension of SQL that is used to write and execute program units such as stored procedures, functions, and triggers in Oracle Database. It offers a wide range of features such as exception handling, variable declaration, loops, conditional statements, and more, which make it a powerful tool for developing complex database applications. The applications of PL/SQL include building database applications, automating database administration tasks, creating reports, and more.

2. **What is deterministic in stored funcions mean?**

   In PL/SQL, a stored function is deterministic if it always returns the same result for the same set of input parameters. This means that if the input parameters for a deterministic function remain the same, the function will always return the same output. This property is important because it enables developers to write functions that can be used in a wider range of contexts and can be optimized by the Oracle database engine.

3. **Explain Various Input Parameter in PLSQL**

   Various input parameters in PL/SQL include:

   (a) IN: This parameter is used to pass values into a stored procedure or function. The values of the IN parameter are read-only within the program unit and cannot be modified.

   (b) OUT: This parameter is used to return values from a stored procedure or function. The values of the OUT parameter are write-only within the program unit and must be assigned a value before the program unit completes.

   (c) IN OUT: This parameter is used to pass values into a stored procedure or function and return values back to the calling program. The values of the IN OUT parameter can be read and modified within the program unit.

   (d) DEFAULT: This parameter is used to provide a default value for a parameter. If a value is not specified for the parameter when the program unit is called, the default value will be used instead.