SE Comp - B	Roll number : 8942
Experiment no. : 8	Date of Implementation : 21/04/2021

Aim : To implement simple PLSql using Mysql stored procedures

Tool Used : Mysql / PostgreSQL

Related Course outcome : At the end of the course, Students will be able to Use SQL : Standard language of relational database

Rubrics for assessment of Experiment:

Indicator	Poor	Average	Good
Timeliness • Maintains assignment deadline (3)	Assignment not done (0)	One or More than One week late (1-2)	Maintains deadline (3)
Completeness and neatness • Complete all parts of assignment(3)	N/A	< 80% complete (1-2)	100% complete (3)
Originality • Extent of plagiarism(2)	Copied it from someone else(0)	At least few questions have been done without copying(1)	Assignment has been solved completely without copying (2)
Knowledge • In depth knowledge of the assignment(2)	Unable to answer 2 questions(0)	Unable to answer 1 question (1)	Able to answer 2 questions (2)

Assessment Marks :

Timeliness	
Completeness and neatness	
Originality	
Knowledge	
Total	

Total : **(Out of 10)**

Teacher's Sign :

EXPERIMENT 8	Procedural sql
Aim	To implement PLSQL stored procedures
Tools	Link for Mysql: https://www.javatpoint.com/mysql-stored-function
Procedure	

PL/SQL is a combination of SQL along with the procedural features of programming languages. Basic Syntax of PL/SQL which is a block-structured language; this means that the PL/SQL programs are divided and written in logical blocks of code. Each block consists of three sub-parts. Every PL/SQL statement ends with a semicolon (;). Following is the basic structure of a PL/SQL block –

```
DECLARE
<Declaration statements>
BEGIN
<Executable commands>
EXCEPTION
<Exception Handling>
END;
```

Simple example of a PL/SQL block:

```
Declare
Msg varcha2(20)  := 'Helo world'
Begin
Dbms.output.put_line(msg);
End;
```

In Oracle, this simple block without name works. Such blocks are called anonymous blocks, but in Mysql , we cannot have anonymous name. In Mysql this can be implemented as named block or stored procedures.
Anonymous blocks are PL/SQL blocks which do not have any names assigned to them.
They need to be created and used in the same session because they will not be stored in the server as a database objects.
Named blocks are having a specific and unique name for them.
They are stored as the database objects in the server.
Since they are available as database objects, they can be referred to or used as long as it is present in the server.

Example : Unnamed blocks

```
declare
num number:=1;
begin
for num in 1..10 loop
    dbms_output.put_line(num);
end loop;
end;
```

Named block/stored procedure:

```
CREATE PROCEDURE sp_name ([proc_parameter: [ IN | OUT | INOUT
] param_name data_type])

Begin
< declare variable_name data_type>
<control statements if else/loop>
SQL executable statements; End
```

End
Where,
procedure_name:
The name to assign to this procedure in MySQL.
Parameter:
Optional. One or more parameters passed into the procedure. When creating a procedure, there are three types of parameters that can be declared:

1. IN - The parameter can be referenced by the procedure. The value of the parameter can not be overwritten by the procedure.
2. OUT - The parameter can not be referenced by the procedure, but the value of the parameter can be overwritten by the procedure.
3. IN OUT - The parameter can be referenced by the procedure and the value of the parameter can be overwritten by the procedure.

declaration_section
The place in the procedure where you declare local variables.
executable_section
The place in the procedure where you enter the code for the procedure.

Example: Create a procedure for adding two numbers

```
DELIMITER //
CREATE procedure sumtwo ( IN a int, IN b int, OUT c INT )
BEGIN
    set c=a+b;
END //
DELIMITER ;
```

Note: here Delimiter could be any character like // or && or $$
Calling the procedure sumtwo:

```
call sumtwo(10,20,@var);
select @var;
```

Example: If..else.. Create a simple procedure that takes age as an input and gives status as 'senior citizen' or 'Not senior citizen' as output

```
DELIMITER //
CREATE PROCEDURE senior_citizen(IN age int, OUT status varcharacter(20))
BEGIN
        IF age >= 60 THEN
                set status = 'Senior citizen';
        ELSE
                set status = 'Not a senior citizen';
        END IF;
```

```
END //
DELIMITER ;

call senior_citizen(65,@status);
select @status;
```

Example: If..else..if ladder

```
DELIMITER //
CREATE PROCEDURE cal_grades(IN marks int, OUT grade varchar(10))
BEGIN
        IF marks >= 75 THEN
                set grade = 'DISTICTION';
        elseif marks>= 60 then
    set grade = 'First class';
  elseif marks >= 50 then
    set grade = 'second class';
        elseif marks>=40 then
    set grade = 'pass class';
  else
    set  grade = 'fail';
end if;
end //

call cal_grades(45,@grade);
select @grade;
```

Example: while Loop

Create a procedure to calculate income

```
DELIMITER //
CREATE procedure CalcIncome ( OUT ending_value INT )
BEGIN
  DECLARE income INT;
  SET income = 50;
  label1: WHILE income <= 3000 DO
    SET income = income * 2;
  END WHILE label1;
  SET ending_value = income;
END; //
DELIMITER ;
```

You could then reference your new procedure as follows:

```
Call CalcIncome(@varname)
Selecr @varname;
```

References

1) https://www.techonthenet.com/mysql/procedures.php
2) https://www.mysqltutorial.org/mysql-stored-procedure-tutorial.aspx

Procedure

Task 1: Write PL/Sql block for the following

1. Write a stored procedure to accept salary of the employee and display grade of employee accordingly.
   If salary > 50000 then grade is 'A'
   If salary between 30000 to 50000 the grade is 'B' and
   If salary < 30000 then grade is 'c'
2. Write a block to display sum of 1 to 10 numbers
3. Write a block to display Fibonacci series upto 8th term (start with 0,1)

Task 2:

Create following tables:
Borrower(roll_no, name, DateofIssue, book_name, Status)
Fine(roll_no, Date , Amt)

Insert few values into Borrower table

Borrower Table

Roll_no	Name	DateofIssue	NameofBook	Status
1	Amita	2017-06-25	Java	I
2	Sonakshi	2017-07-10	Networking	I
3	Nira	2017-05-22	MySQL	I
4	Jagdish	2017-06-10	DBMS	I
5	Jayashree	2017-07-05	MySQL	I
6	Kiran	2017-06-30	Java	I

Fine Table

Roll_no	Date	Amt

Write a procedure that Accepts roll_no & name of book from user.
Check the number of days (from date of issue)

1) If no. of days > 30, per day fine will be Rs 50 per day &
2) if days are between 15 to 30 then fine amount will be Rs 10 per day.
3) for days less than 15, Rs. 5 per day.

Whenever student returns the book, update both the tables

1) Make status of that book in Borrower table as 'R' ('R'; for return)
2) If there is a fine , then new row should be added to the fine table.

Table after procedure Run

Borrower Table

Rno	Name	DateofIssue	NameofBook	Status
1	Amita	2017-06-25	Java	I
2	Sonakshi	2017-07-10	Networking	I
3	Nira	2017-05-22	MySQL	I
4	Jagdish	2017-06-10	DBMS	R
5	Jayashree	2017-07-05	MySQL	I
6	Kiran	2017-06-30	Java	I

Fine Table

Roll_no	Date	Amt
4	2017-06-30	100

Post Lab Questions:

1. Give advantages of PLSQL vs SQL
2. Explain data types of PLSQL in Mysql

1. **Creating a database as exp_8**

Query:

create database exp_8;

Task 1: Write PL/Sql block for the following

1. Write a stored procedure to accept salary of the employee and display grade of employee accordingly.
If salary > 50000 then grade is 'A'
If salary between 30000 to 50000 the grade is 'B' and
If salary < 30000 then grade is 'c'

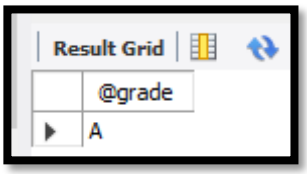
CODE:

```
DELIMITER //
CREATE DEFINER='root'@'localhost' PROCEDURE cal_grade(IN salary int
, OUT grade varchar(10))
BEGIN
if salary >= 50000 then
   set grade = 'A';
elseif salary >= 30000 and salary < 50000 then
   set grade = 'B';
elseif salary < 30000 then
   set grade = 'C';
end if;
end //
```

OUTPUT:

Test Case 1:

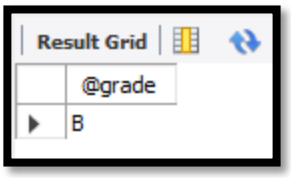
```
call cal_grade(550000,@grade);
select @grade;
```



Test Case 2:

call cal_grade(45000,@grade);

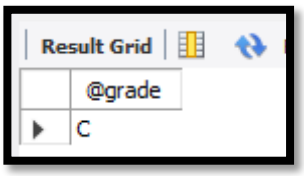
select @grade;



Test Case 3:

call cal_grade(5000,@grade);

select @grade;



2. Write a block to display sum of 1 to 10 numbers

CODE:

```
DELIMITER //
CREATE PROCEDURE sum_of_ten (IN n int, OUT total int)
BEGIN
    declare i, answer INT;
    set i=0;
    set answer=0;
    while i<=n do
            set answer=answer+i;
        set i=i+1;
    end while;
    set total=answer;
end //
```

OUTPUT:

```
call sum_of_ten(20,@total);
select @total;
```

Result Grid

	@total
▶	210

3. Write a block to display Fibonacci series upto 8th term (start with 0,1)

CODE:

```
DELIMITER //
CREATE procedure fibbonacci_series ( IN no_of_terms INT )
BEGIN
   declare i INT;
   declare f0,f1,f INT;
   set i = 2;
   set f0 = 0;
   set f1 = 1;
   select f0,f1;
   set f = f0+f1;
   label1: while i <= no_of_terms
   DO
       select f;
     set i = i + 1;
     set f0 = f1;
     set f1 = f;
     set f = f0+f1;
   end while label1;
end //
```

OUTPUT:

Result Grid

	f0	f1
▶	0	1

Result Grid

	f
▶	2

Result Grid

	f
▶	3

Result Grid

	f
▶	5

Result Grid

	f
▶	8

Result Grid

	f
▶	13

Result Grid

	f
▶	21

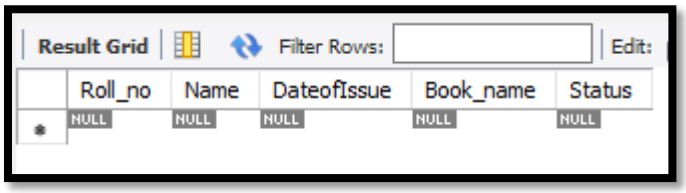
Task 2:

Create following tables:
Borrower(roll_no, name, DateofIssue, book_name, Status)
Fine(roll_no, Date , Amt)

A.] Borrower
1.] Creating table
Query:

```
create table Borrower(
Roll_no varchar(6), PRIMARY KEY(Roll_no),
Name varchar(20) not null,
DateofIssue date,
Book_name varchar(15),
Status  char(1), check(Status in ('I','R')));
```

Screenshot:



2.] Inserting values
Query:

```
Insert into Borrower value('1', 'Praisy', '2020-06-25', 'Java', 'I');
Insert into Borrower value('2', 'Jerry', '2020-07-10', 'MYSQL', 'I');
Insert into Borrower value('3', 'Gini', '2020-05-22', 'Networking', 'I');
Insert into Borrower value('4', 'Ashley', '2020-06-15', 'MYSQL', 'I');
Insert into Borrower value('5', 'Elwin', '2020-07-05', 'DBMS', 'I');
Insert into Borrower value('6', 'Elaine', '2020-06-30', 'Java', 'I');
```

Screenshot:

Result Grid | Filter Rows: | Edit:

Roll_no	Name	DateofIssue	Book_name	Status
1	Praisy	2020-06-25	Java	I
2	Jerry	2020-07-10	MYSQL	I
3	Gini	2021-01-22	Networking	I
4	Ashley	2020-06-15	MYSQL	I
5	Elwin	2021-02-05	DBMS	I
6	Elaine	2021-03-30	Java	I
NULL	NULL	NULL	NULL	NULL

B.]Fine

1.] Creating Table

Query:

```
create table Fine (
Roll_no varchar(6), PRIMARY KEY(Roll_no),
Date date,
Amt numeric(8));
```

Screenshot:

Result Grid | Filter Rows:

Roll_no	Date	Amt
NULL	NULL	NULL

A.] Write a procedure that Accepts roll_no & name of book from user. Check the number of days (from date of issue)
1) If no. of days > 30, per day fine will be Rs 50 per day &
2) if days are between 15 to 30 then fine amount will be Rs 10 per day.
3) for days less than 15, Rs. 5 per day.

CODE:

```
DELIMITER //
create procedure book_fine (IN rno int, IN bookname varchar(2), OUT fine
float, OUT days INT)
begin
    declare idate date;
    select issue_date into idate
    from borrower
    where rollno = rno and book_name = bookname;
    set days =  DATEDIFF(current_date(), idate);
    if days > 30 then
            set fine = 50.0 * days;
    elseif days >= 15 then
            set fine = 10.0*days;
    elseif day >= 10 then
            set fine = 5.0 * days;
    else
            set fine = 0;
      end if;
end//
```

OUTPUT:

Test Case 1:

call book_fine(3,"Networking", @fine, @days);
select @fine, @days;

Result Grid

	@fine	@days
▶	4200	84

Test Case 2:

call book_fine(6,"Java", @fine, @days);
select @fine, @days;

Result Grid

	@fine	@days
▶	170	17

Test Case 3:

call book_fine(2,"MYSQL", @fine, @days);
select @fine, @days;

Result Grid

	@fine	@days
▶	14000	280

B.] Whenever student returns the book, update both the tables
1) Make status of that book in Borrower table as ‘R’ (‘R’; for return)
2) If there is a fine , then new row should be added to the fine table.

CODE:

```
DELIMITER //
create procedure B(in roll_new int, in book_name varchar(20))
begin
    declare X integer;
    declare continue handler for not found
begin
    select 'NOT FOUND';
end;
    select datediff(curdate(),DateofIssue) into X from borrower
    where Roll_no = roll_new;
    if (X > 15 and X < 30) then
      insert into fine value(roll_new,curdate(),(X*10));
    end if;
    if (X < 15) then
      insert into fine value(roll_new,curdate(),(X*5));
    end if;
    if (X > 30) then
      insert into fine values(roll_new,curdate(),(X*50));
    end if;
    SET SQL_SAFE_UPDATES =0;
    update borrower set Status='R'
   where Roll_no = roll_new;
end //
```

OUTPUT:

Query:

call B(4,'MYSQL');

call B(1,'Java');

A.] Table before Updation:

Borrower Table:

Result Grid | Filter Rows: | Edit:

Roll_no	Name	DateofIssue	Book_name	Status
1	Praisy	2020-06-25	Java	I
2	Jerry	2020-07-10	MYSQL	I
3	Gini	2021-01-22	Networking	I
4	Ashley	2020-06-15	MYSQL	I
5	Elwin	2021-02-05	DBMS	I
6	Elaine	2021-03-30	Java	I
NULL	NULL	NULL	NULL	NULL

Fine Table:

Result Grid | Filter Rows:

Roll_no	Date	Amt
NULL	NULL	NULL

B.] Table after Updation:

Borrower Table:

Result Grid | Filter Rows: | Edit:

Roll_no	Name	DateofIssue	Book_name	Status
1	Praisy	2020-06-25	Java	R
2	Jerry	2020-07-10	MYSQL	I
3	Gini	2021-01-22	Networking	I
4	Ashley	2020-06-15	MYSQL	R
5	Elwin	2021-02-05	DBMS	I
6	Elaine	2021-03-30	Java	I
NULL	NULL	NULL	NULL	NULL

Fine Table:

Result Grid | Filter Rows:

Roll_no	Date	Amt
1	2021-04-21	15000
4	2021-04-21	15500
NULL	NULL	NULL

POSTLAB QUESTIONS:

1. **Give advantages of PLSQL vs SQL**

Ans:

SQL	PLSQL
It is a database Structured Query Language.	It is a database programming language using SQL.
Data variable are not available	Data variable are available.
No Supported Control Structures.	Control Structures are available Like, For loop, While loop.
Query performs single operation.	PLSQL block performs Group of Operation as single bloack.
SQL is declarative language.	PLSQL is procedural language.
SQL can be embedded in PLSQL.	PLSQL can’t be embedded in SQL.
It directly interacts with the database server.	It does not interacts directly with the database server.
It is Data oriented language.	It is application oriented language.
It is used to write queries, DDL and DML statements.	It is accustomed write program blocks, functions, procedures triggers,and packages.

2. **Explain data types of PLSQL in Mysql**

Ans: The PL/SQL variables, constants and parameters must have a valid data type, which specifies a storage format, constraints, and a valid range of values.

S.No	Date Type & Description
1	**Numeric** Numeric values on which arithmetic operations are performed.
2	**Character** Alphanumeric values that represent single characters or strings of characters.
3	**Boolean** Logical values on which logical operations are performed.
4	**Datetime** Dates and times.

- A scalar type has no internal components. It holds a single value, such as a number or character string. The scalar types fall into four families, which store number, character, Boolean, and date/time data. The scalar families with their datatypes are:
  - PL/SQL Number Types

    BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, DEC, DECIMAL, DOUBLE PRECISION, FLOAT, INT, INTEGER, NATURAL, NATURALN, NUMBER, NUMERIC, PLS_INTEGER, POSITIVE, POSITIVEN, REAL, SIGNTYPE, SMALLINT

- PL/SQL Character and String Types and PL/SQL National Character Types

  CHAR, CHARACTER, LONG, LONG RAW, NCHAR, NVARCHAR2, RAW, ROWID, STRING, UROWID, VARCHAR, VARCHAR2

  Note that the LONG and LONG RAW datatypes are supported only for backward compatibility; see "LONG and LONG RAW Datatypes" for more information.

- PL/SQL Boolean Types

  BOOLEAN

- PL/SQL Date, Time, and Interval Types

  DATE, TIMESTAMP, TIMESTAMP WITH TIMEZONE, TIMESTAMP WITH LOCAL TIMEZONE, INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND