LAB 6

Title: Stored procedure and Views

Objective:

- To be familiar with concept of Stored procedure and implement it
- To be familiar with concept of views and analyze the role of views for data securit

Theory:

Stored procedure

- ✓ A stored procedure is a named collection of SQL statements that are precompiled and stored in a database.
- ✓ If the user has an SQL query that you write over and over again, keep it as a stored procedure and execute it.
- ✓ Users can also pass parameters to a stored procedure so that the stored procedure can act based on the parameter value that is given.
- ✓ Based on the statements in the procedure and the parameters we pass, it can perform one or multiple DML operations on the database, and return value, if any.
- ✓ Thus, it allows us to pass the same statements multiple times, thereby, enabling reusability.

Creating stored procedure

To create a stored procedure in MySQL, we can use the following syntax:

```
DELIMITER //
CREATE PROCEDURE procedure_name(parameter1 datatype, parameter2 datatype, ...)
BEGIN
----Statements using the input parametes
END //
DELIMITER ;
```

Note:

- ➢ **DELIMITER // is used to change the delimiter temporarily so that you can use the semicolon ; within the procedure body without ending the entire statement prematurely.**
- ➢ **DELIMITER ; sets the delimiter back to the default semicolon ;**

Executing stored procedure

To execute a stored procedure in MySQL, you can use the CALL statement followed by the name of the procedure and list of parameters if any. The syntax is as follows:

```
CALL procedure_name(parameter_list);
```

Creating stored procedure without parameters

Example:

```
DELIMITER //
CREATE PROCEDURE getallEmployee ()
BEGIN
SELECT * FROM employee;
END //
DELIMITER ;
```

Now, we can execute the above stored procedure as follows

```
CALL getallEmployee();
```

Creating parameterized stored procedure

- ✓ In SQL, a parameterized procedure is a type of stored procedure that can accept input parameters.
- ✓ These parameters can be used to customize the behavior of the procedure and perform operations based on the input values provided.
- ✓ To create a parameterized stored procedure in MySQL, we can define input parameters within the procedure definition.

Example:

```
DELIMITER //
CREATE PROCEDURE getdepartmentEmployee (dept varchar(30))
BEGIN
SELECT *
FROM employee
WHERE department=dept;
END //
DELIMITER ;
```

To call this stored procedure, you can use the CALL statement and pass the parameter value:

```
CALL getdepartmentEmployee('civil');
```

Drop procedure

We can use the DROP PROCEDURE statement followed by the name of the procedure.

Here's the syntax:

```
DROP PROCEDURE procedure_name;
```

Example:

```
DROP PROCEDURE getdepartmentEmployee;
```

Views

- ✓ In SQL, a view is a virtual table based on the result-set of an SQL statement.
- ✓ A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
- ✓ Views are defined based on queries, and they can be used to simplify complex queries, restrict access to certain data, or present a customized perspective of the data to different users or applications.

In some cases, it is not desirable for all users to see all the actual relations stored in the database.

For example consider the following relation

Employee(emp_id,emp_name,postion,salary,dept_id)
Department(dep_id,dept_name,location,budget)

Consider a person who needs to know information of employees with name, position and department name but not salary as well as other information, then this person should see a relation described.

In such case views are created.

A view is created with the CREATE VIEW statement.

Syntax

```
CREATE VIEW view_name AS

SELECT column1, column2, ...

FROM table_name

WHERE condition;
```

Types of views:

1. **simple view:** creating views from single table
2. **complex view:** creating views from multiple table

For dropping view

Syntax: DROP VIEW view_name;

Problem:

stored procedure

1) Create any database
2) Create two tables with following columns where underlined attributes represent primary key

   Employee(emp_id,emp_name,postion,salary,dept_id)
   Department(dep_id,dept_name,location,budget)

3)Insert at least 5 rows in each tables

4)Create stored procedure without using parameters to

i)To Find all the information of employee.

ii) To Find emp_name, position, salary, dept_name of employee

5) Execute the above created stored procedure

6)Create stored procedure to with using parameters

i) To find the information of employee of specified department

ii) To find the information of employee of specified employee name and department location.

Views

1) Create view for display the emp_id ,emp_name, position of employee
2) Create view for display emp_id, emp_name, position, dept_name, location

Now, try to select data from views and fetch different data according to requirements.

Discussion: *(This portion is left for student)*

Conclusion: *(This portion is left for student)*

*******THE END*******

Hint: for creating table and inserting rows in table

Create table

```
CREATE TABLE Department (
  dept_id INT PRIMARY KEY,
  dept_name VARCHAR(50),
  location VARCHAR(50),
  budget DECIMAL(12, 2)
);

CREATE TABLE Employee (
  emp_id INT PRIMARY KEY,
  emp_name VARCHAR(50),
  position VARCHAR(50),
  salary DECIMAL(10, 2),
  dept_id INT,
  FOREIGN KEY (dept_id) REFERENCES Department(dept_id)
);
```

Now inserting some records into table

--inserting data into Department one by one

```
INSERT INTO Department
VALUES (1, 'IT', 'Kathmandu', 50000);

INSERT INTO Department
VALUES (2, 'HR', 'Biratnagar', 30000);

INSERT INTO Department
VALUES  (3, 'Marketing', 'Pokhara', 40000);

INSERT INTO Department
VALUES  (4, 'Finance', 'Butwal', 35000);


INSERT INTO Department
VALUES  (5, 'Operations', 'Dharan', 45000);
```

--- Insert data into the Employee table one by one

```
INSERT INTO Employee
VALUES (1, 'Pradip Paudel', 'Manager', 5000, 1);

INSERT INTO Employee
VALUES (2, 'Roshan Pandey', 'Developer', 3000, 1);

INSERT INTO Employee
VALUES(3, 'Samjhana Bhattrai', 'Analyst', 3500, 2);

INSERT INTO Employee
VALUES(4, 'Anish Sharma', 'Designer', 2500, 3);

INSERT INTO Employee
VALUES  (5, 'Aakritee Bagale', 'Tester', 2000, 3);
```