Stored Procedures and Functions

SCS2209- Database II

Ms. Hiruni Kegalle

Stored Procedures

Stored Procedure

- Procedure that is stored in a DB.
- It can be reused without writing the same SQL statement for each time
- Improves performance by executing the procedure on the server
- Compile only once and if only modified then recompile
- Can provide security since user can execute stored procedure rather accessing tables
- If only the business logic changes then only change the stored procedure
- Return 0 or more values

Example - Query

```
SELECT Product, Quantity
FROM Inventory
WHERE Warehouse = 'Colombo'
```

- Each time a query is executed the database server recompile and execute from the beginning
- Permission to access the table is also required

Example - Stored Procedure

```
CREATE PROCEDURE getLocation (IN location varchar(15))
BEGIN
SELECT Product, Qty
FROM Inventory
WHERE Warehouse = @location
END;

Call getLocation('Colombo');
```

Stored Procedure - Syntax

Create Procedure <proc_name> (param_spec$_1$, param_spec$_2$, ...,param_spec$_n$)

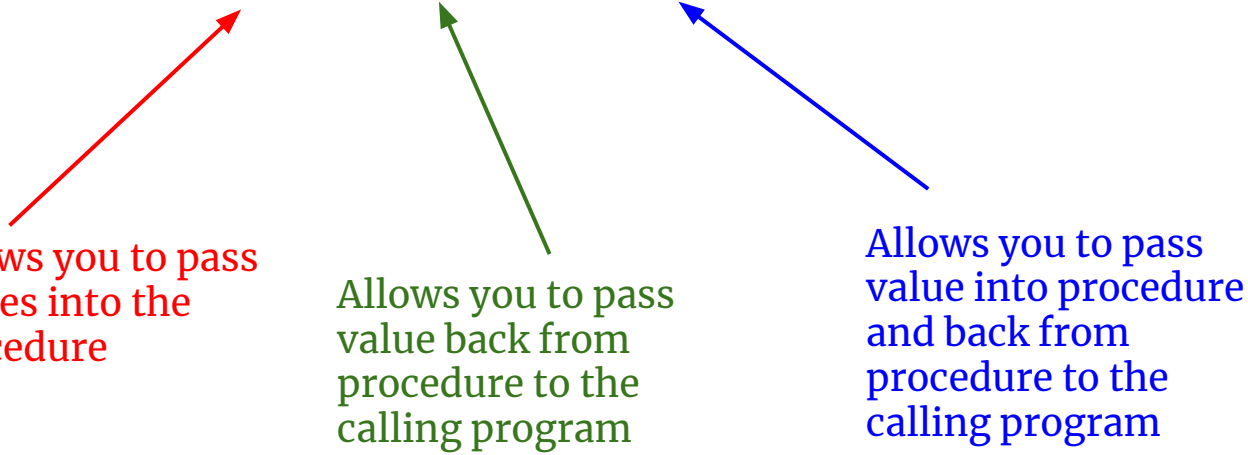
Begin

    Execution code

End;

Stored Procedure - Syntax

param_spec is of the form:
[**IN** | **OUT** | **INOUT**] <param_name> <param_type>



Allows you to pass values into the procedure

Allows you to pass value back from procedure to the calling program

Allows you to pass value into procedure and back from procedure to the calling program

Example - parameter (IN)


```
1   DELIMITER //
2   CREATE PROCEDURE getOfficeByCountry( IN countryName VARCHAR(255))
3
4   BEGIN
5
6   SELECT *
7   FROM offices
8   WHERE country= countryName;
9   END //
10
11  DELIMITER ;
```

```
1   CALL getOfficeByCountry('USA');
```

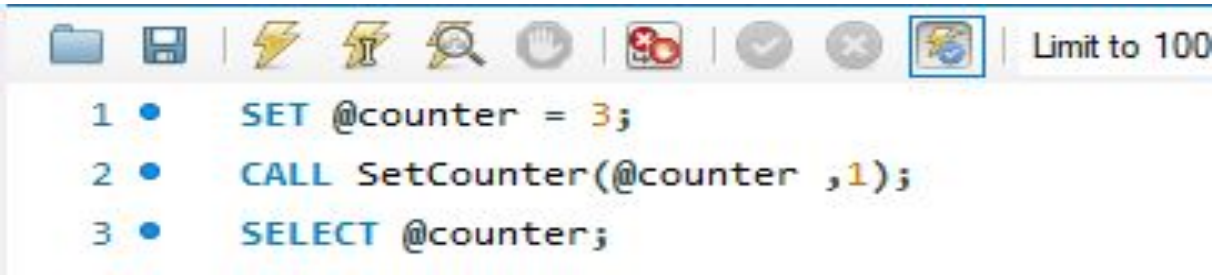
Example - parameter (OUT)


```
1   DELIMITER $$
2
3   CREATE PROCEDURE GetOrderCountByStatus (
4       IN  orderStatus VARCHAR(25),
5       OUT total INT
6   )
7   BEGIN
8       SELECT COUNT(orderNumber)
9       INTO total
10      FROM orders
11      WHERE status = orderStatus;
12  END$$
13
14  DELIMITER ;
```

```
CALL GetOrderCountByStatus('shipped', @total);
SELECT @total;
```

Example - parameter (INOUT)

```
1    DELIMITER $$
2    CREATE PROCEDURE SetCounter( INOUT counter INT, IN inc INT)
3
4    BEGIN
5        SET counter = counter + inc;
6    END $$
7
8    DELIMITER ;
9
```



IF-THEN statements in SP

```
DELIMITER //
CREATE PROCEDURE getCustomerLevel(
    IN NUMBER INT,
    OUT LEVEL VARCHAR(30))
BEGIN
    DECLARE credit DECIMAL(10, 2) ;
    SELECT creditLimit
    INTO credit
    FROM customers
    WHERE customerNumber = NUMBER ;

    IF(credit > 50000) THEN
        SET LEVEL = "PLATINUM" ;
    END IF ;
END //
```

```
call getCustomerLevel(141, @level);
select @level;
```

IF-THEN-ELSE statements in SP

```
DELIMITER //
CREATE PROCEDURE getCustomerLevel2(
    IN NUMBER INT,
    OUT LEVEL VARCHAR(30))
BEGIN
    DECLARE credit DECIMAL(10, 2) ;
    SELECT creditLimit
    INTO credit
    FROM customers
    WHERE customerNumber = NUMBER ;

    IF(credit > 50000) THEN
        SET LEVEL = "PLATINUM" ;
    ELSE
         SET LEVEL = "NOT PLATINUM" ;
    END IF ;
END //
```

```
call getCustomerLevel2(447, @level2);
select @level2;
```

Activity

Create a Procedure to get customer level for a given customer Number. (Modify the previous SP)

- Credit limit greater than 50,000 -> Platinum
- Credit limit between 50,000 and 9999 -> Gold
- Otherwise -> Silver

Try the procedure with **125, 447, 141** customer Numbers

Activity - Answer

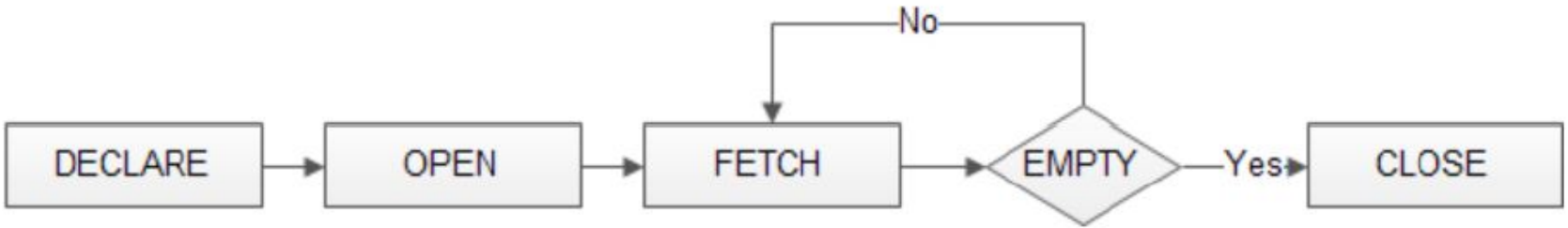
```
DELIMITER //
CREATE PROCEDURE getCustomerLevel3(
    IN NUMBER INT,
    OUT LEVEL VARCHAR(30))
BEGIN
    DECLARE credit DECIMAL(10, 2) ;
    SELECT creditLimit
    INTO credit
    FROM customers
    WHERE customerNumber = NUMBER ;

    IF(credit > 50000) THEN
        SET LEVEL = "PLATINUM" ;
    ELSEIF (credit <= 50000 AND credit > 10000) THEN
         SET LEVEL = "GOLD" ;
    ELSE
        SET LEVEL = "SILVER" ;
    END IF ;
END //
```

MySQL Cursor

- To handle a result set inside a SP, we use a cursor.
- A cursor allows us to iterate set of rows returned by a query and process each row individually.
- Can be used in triggers, SP and functions.
- Always associate with a SELECT statement
- A cursor is,
    - Read-only: Cannot update data of a tale using cursor
    - Non-scrollable: Fetching can be done only in given order (no skipping/ jumping to specific rows)

MySQL Cursor - Steps

1. Declare a cursor
2. Open (Initialize result set for the cursor)
3. Fetch (retrieve the next row pointed by cursor + move cursor to next row)
4. Check if there are more rows to fetch
5. Declare “Not Found” handler
6. Close the cursor (Deactivate)



We'll create SP to get the email list of all employees

```
set @emailLst ="";
call createEmailList(@emailLst);
SELECT @emailLst;
```

```
DELIMITER //
CREATE PROCEDURE createEmailList (INOUT emailList varchar (4000))
BEGIN
    DECLARE finished INT DEFAULT 0;
    DECLARE emailAdd varchar(100);

    DECLARE curEmail CURSOR FOR
        SELECT email FROM employees;

   DECLARE CONTINUE HANDLER FOR
        NOT FOUND SET finished =1;

   OPEN curEmail;

   getEmail: LOOP
        FETCH curEmail INTO emailAdd;
        IF (finished =1) THEN
            LEAVE getEmail;
        END IF;

        SET emailList = Concat( emailAdd, ";", emailList);
        END LOOP getEmail;

        CLOSE curEmail;
END //
```

Functions

Functions

- Functions are user defined and use to compute values. But cannot use to change the DB
- Comparing to SP,
    - Functions must return a value
    - Only have input parameters
    - Can use inside a SP, but not vice versa
    - Compiled and execute each time it is being called

Function - Syntax

Create Function <function_name> (param1, param2, ...)

Returns <data type>

[Not] Deterministic

Begin

    Execution code

End;

Function - Example

```
DELIMITER //
create FUNCTION getCustomerLevel (credit decimal (10,2))

returns varchar (20)
DETERMINISTIC
BEGIN
    DECLARE customerLevel varchar(20);
    IF credit > 50000 THEN
        SET customerLevel ="PLATINUM";
    ELSEIF credit <= 50000 and credit > 10000 THEN
        SET customerLevel ="GOLD";
    ELSEIF credit <= 10000 THEN
        SET customerLevel ="SILVER";
    END iF;

    RETURN (customerLevel);
END//
```

Function - Example

```
SELECT customerName, getCustomerLevel(creditLimit)
FROM customers
ORDER BY customerName
```