08 - Stored Procedures

Thursday, April 4, 2024 5:39 AM

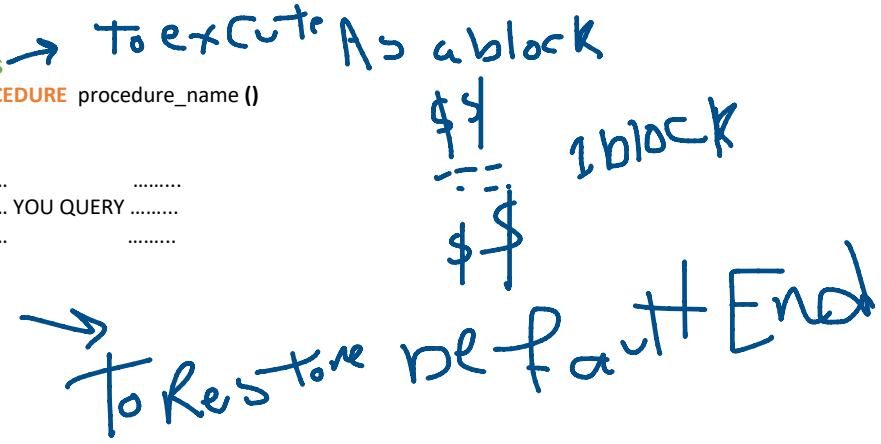
What is Stored Procedures ?

A stored procedure is a set of SQL statements that are stored in a database and can be executed repeatedly as a single unit.
It is a precompiled collection of SQL commands and procedural logic, designed to perform specific tasks or operations within a database system.

How to create Stored Procedures ?

- Just Note You need this to deal with as block of code that must execute together so you must tell mySQL that

```
DELIMITER $$
CREATE PROCEDURE  procedure_name ()

    BEGIN
        ……..                    ……..
        …….. YOU QUERY ……..
        ……..                    ……..
    END $$

DELIMITER ;
```




```
DELIMITER $$
CREATE PROCEDURE get_clients()
BEGIN

  SELECT * FROM clients;

END $$
DELIMITER ;
```

HOW TO CALL THEM ?

You can Call Stored Procedure using this two ways

- CALL Procedure_name()
- USING Calling icon in the DBMS

How To Drop any Stored Procedure ?

```
DROP PROCEDURE procedure_name
DROP PROCEDURE IF EXIST procedure_name  // RECOMMENDED
```

```
DROP PROCEDURE IF EXISTS get_voices_with_balance
```

```
DELIMITER $$
CREATE PROCEDURE get_voices_with_balance()
BEGIN

   SELECT invoice_id , invoice_total,payment_total , invoice_total - payment_total as Balance
   FROM invoices;

END $$
DELIMITER ;
```

THE RECOMMENDED TEMPLATE FOR CREATING PROCEDURE

Stored Procedure With Parameter

```
CREATE PROCEDURE get_clients_by_state(

    Param_name dataType ,
    Param_name dataType


)
```

```
DELIMITER $$
CREATE PROCEDURE get_clients_by_state(

   state CHAR(2) ,

)
BEGIN

   SELECT  *
   FROM clients c
   WHERE c.state = state;

END $$
DELIMITER ;
```

Stored Procedure With DEFAULT Values

[1]

```
DELIMITER $$
CREATE PROCEDURE get_clients_by_state(

   state CHAR(2)
)
BEGIN

   SELECT  *
   FROM clients c
   WHERE c.state = IFNULL( state , c.state);

END $$
```

How To Make Input Validation For params For Your Stored Procedure ?

You Make IT Using

```
IF  condition THEN
      SIGNAL SQLSTATE ' '                    -> https://www.ibm.com/docs/en/db2-for-zos/11?topic=codes-sqlstate-values-common-error
      SET MESSAGE_TEXT = ' ';
END IF ;
```

```
DROP PROCEDURE IF EXISTS update_payments ;
DELIMITER $$
CREATE PROCEDURE update_payments(

   invoice_id INT ,
   payment_amount DECIMAL(9,2) ,
   payment_date date

)
BEGIN

   IF payment_amount <= 0 THEN
   SIGNAL SQLSTATE '22003'
   SET MESSAGE_TEXT = 'NOT VALID DATE RANGE ';
   END IF ;


      UPDATE invoices i
   SET
            i.payment_date = payment_date ,
      i.payment_total = payment_amount
      WHERE i.invoice_id = invoice_id ;

END $$
DELIMITER ;
```

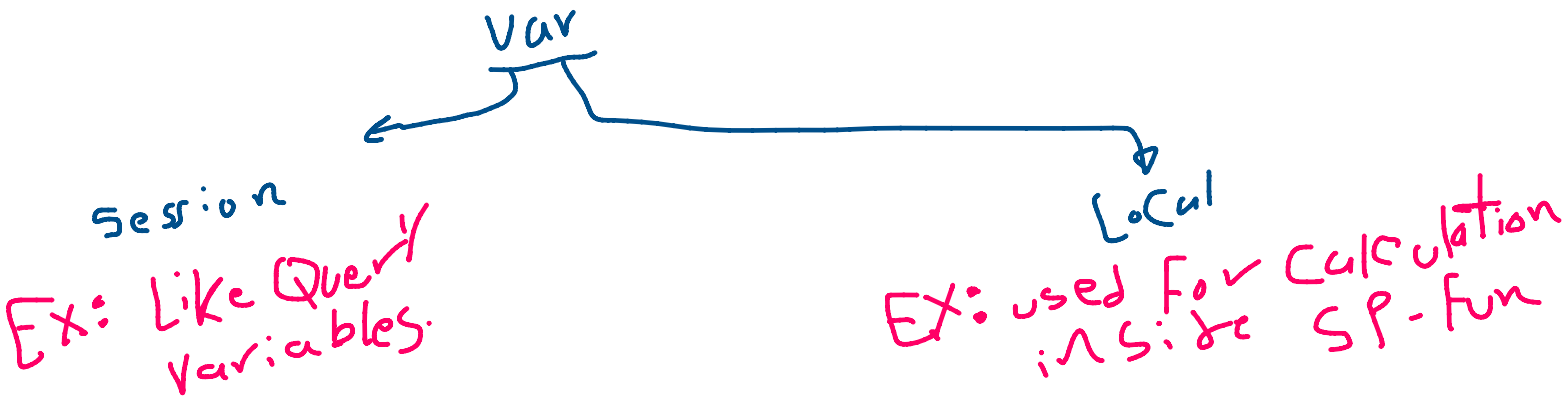
validation - part

Output Values / Param

```
DROP PROCEDURE  IF EXISTS get_unpaid_invoices;
DELIMITER $$

CREATE PROCEDURE get_unpaid_invoices(

   client_id INT ,
   OUT countX INT ,
   OUT total_unPaid DECIMAL(9,2)

)
BEGIN


   SELECT COUNT(*) , SUM(invoice_total)
   FROM invoices i
   WHERE i.client_id = client_id AND payment_total = 0;

END $$

DELIMITER ;
```

by default all are input

Variables In SQL



- **Session Variables**

- **Local Variables**

```
CREATE PROCEDURE Calc_Risk()
BEGIN

   DECLARE risk_factor DECIMAL(9 , 2) DEFAULT 0;
   DECLARE invoices_total DECIMAL(9 , 2) ;
   DECLARE invoices_count INT DEFAULT 0;


   SELECT COUNT(*) , SUM(invoices_total)
   INTO invoices_count , invoices_total
   FROM invoices ;


   SET risk_factor = invoices_total / invoices_count * 5;
   SELECT risk_factor;

END
```

Functions

Same AS Stored Procedure but the difference here **is that it must return 1 value**

```
CREATE FUNCTION `FUN_NAME` ()
RETURNS INTEGER
-
-
-
BEGIN
```

```
RETURN 1;
END



CREATE  FUNCTION fun_risk(
      client_id INT
)

RETURNS INT
   READS SQL DATA
BEGIN
  DECLARE risk_factor DECIMAL(9 , 2) DEFAULT 0;
   DECLARE invoices_total DECIMAL(9 , 2) ;
   DECLARE invoices_count INT DEFAULT 0;


   SELECT COUNT(*) , SUM(invoices_total)
   INTO invoices_count , invoices_total
   FROM invoices i
   WHERE i.client_id = client_id;


   SET risk_factor =  invoices_total / invoices_count  * 5;

RETURN risk_factor;
END
```