Funciones y Procedimientos almacenados

Un procedimiento almacenado (P.A)es un conjunto de comandos SQL que se ejecuta en el servidor de bases de datos para realizar una determinada tarea o función
Un P.A. es un programa que se guarda físicamente en una base de datos
Los PA pueden mejorar el rendimiento ya que se necesita enviar menos información entre el servidor y el cliente
Algunas situaciones donde son útiles los PA:

- Cuando múltiples aplicaciones clientes se escriben en distintos lenguajes o funcionan en distintas plataformas, pero realizan la misma operación en la base de datos
- Cuando la seguridad es muy importante.
- Las aplicaciones y los usuarios no tienen acceso directo a las tablas de la base de datos, sólo pueden ejecutar algunos P.A.

PROCEDIMIENTO

```
DELIMITER $$

Create procedure nombre_procedimiento([IN|OUT|INOUT] nombre_parametro tipo_datos,…)

BEGIN

    cuerpo del procedimiento

END $$

DELIMITER ;
```

Procedimientos

```
Delimiter $$
Create procedure saludo()
    Begin
        select ‘Hola grupo’ as saludo;
    End $$
    Delimiter ;
```

```
Delimiter $$
Create procedure saludo()
Begin
select ‘Hola grupo’ as saludo;
End $$
Delimiter ;
```

Para ejecutar el procedimiento:

```
CALL saludo();
```

Procedimientos

```
Ejemplo2:
Delimiter $$
Create procedure saludo2(IN nombre char(20),OUT edad int)
Begin
      select concat('Bienvenido',upper(nombre),'como esta usted') as 'saludo';
      set edad=30;
End $$
Delimiter ;
```

Para ejecutar el procedimiento:

```
CALL saludo2('pepe',@edad);
Select @edad;
```

Funciones

Delimiter $$

Create function nombre(nombre param tipo datos…)

Returns tipo_dato

Begin

    Cuerpo de la función

End $$

Delimiter;

Para llamar a la función:

Select hola();

```
Delimiter $$
Create function hola()
Returns char(40)
Begin
      return ‘Hola grupo’;
End $$
Delimiter ;
```

DROP PROCEDURE Y DROP FUNCTION

- Para eliminar un procedimiento o una función utilizamos las instrucciones Drop Procedure y Drop function
- Drop procedure if exists nombre_procedimiento
- Drop function if exists nombre_function
- La clausula if exists evita que ocurra un error si la función o procedimiento no existe

  ejemplo: drop function if exists hola;

Parámetros y variables

- Los parámetros,se indican en la definición y pueden ser declarados como variables de entrada (IN), de salida(OUT), o de entrada/salida (INOUT). **Las funciones solo tienen parámetros de entrada.**
- Las variables se deben declarar al comienzo de un bloque begin..end y serán locales o visibles dentro de ese bloque. Para la declaración se utiliza **DECLARE** y debe ser la primera instrucción del BEGIN..END
- Se puede dar un valor por defecto con DEFAULT
- SET y SELECT INTO: asigna valores a las variables.

- DECLARE saludo varchar(50) DEFAULT ‘Hola mundo’
- Set saludo=’Otra cadena’;

Sintaxis: Declare, set, select..into

- Declare var1,var2..type [Default value]

  en un mismo declare solo se pueden declarar variables del mismo tipo
- Set var1=expr,var2=expr…
- SELECT columna1,columna1 INTO var1,var2 from table

  (PARA CADA VARIABLE SOLO SE PUEDE DEVOLVER UN VALOR)

EJEMPLO

```
DELIMITER $$
CREATE FUNCTION saludo
RETURNS varchar(30)
BEGIN
DECLARE salida varchar(30) default ‘Hola mundo’;
Set salida=’Hola mundo del MySQL’;
Return salida;
END $$
```

Para llamar a la función:

```
Select saludo();
```

Ejemplo 2

```
Delimiter $$
drop function if exists totalventas $$
create function totalventas()
returns int
begin
Declare total int;
Select sum(cantidad) into total from ventas;
Return total;
End $$
```

Para llamar a la función:

```
Select totalventas();
```

Instrucciones de control de flujo

IF condicion **THEN**

instruccion

[ELSEIF condicion THEN instrucción]

[ELSE instrucción]

END IF;

Instrucción: Debe ser una sentencia simple o varias sentencias dentro de un bloque BEGIN...END

INSTRUCCIONES DE CONTROL DE FLUJO

CASE expresion o valor

WHEN valor1 THEN instruccion1

WHEN valor2 THEN instruccion2

[ELSE instrucción n]

END CASE

expresion o valor puede ser también una condición a evaluar

Instrucción debe ser simple sino entre BEGIN ..END

INSTRUCCIONES DE CONTROL DE FLUJO

REPEAT

Sentencias

UNTIL condición

END REPEAT;

WHILE condicion DO

sentencias

END WHILE;

EJERCICIOS

- Crea una función que según el color azul,rojo, amarillo, verde y devuelva el color en inglés utilizando la estructura CASE. Cuando no sea ninguno de esos colores que devuelva’no entiendo’

```
DELIMITER $$
DROP FUNCTION if exists color $$
CREATE FUNCION color(dato varchar(50))
RETURNS VARCHAR(30)
BEGIN
declare valor varchar(50)
CASE dato
WHEN ‘azul’ THEN set valor=’blue’;
WHEN ‘rojo’ THEN set valor=’red’;
WHEN ‘amarillo’ THEN set valor=’yellow’;
WHEN ‘verde’  THEN set valor=’green’;
ELSE set valor=’No entiendo’;
END CASE;
RETURN valor;
END $$
DELIMITER ;
```

```
MariaDB [almacen]> select color('verde');
+----------------+
| color('verde') |
+----------------+
| green          |
+----------------+
1 row in set (0.001 sec)

MariaDB [almacen]> select color('marron');
+----------------+
| color('marron') |
+----------------+
| No entiendo     |
+----------------+
1 row in set (0.001 sec)
```

- Crea un procedimiento que sume los primeros n números,crea además una variable de sesion para poner el @resultado

```
Delimiter $$
drop procedure if exists sumar;
Create procedure sumar(in num int)
Begin
declare contador int;
    set contador=1;
    set @resultado=1;
    While contador<num do
        set contador=contador+1;
        Set @resultado=@resultado+contador;
    End while;
END $$
DELIMITER ;;
```

```
MariaDB [almacen]> call sumar(5);
Query OK, 0 rows affected (0.001 sec)

MariaDB [almacen]> select @resultado;
+------------+
| @resultado |
+------------+
|         15 |
+------------+
1 row in set (0.000 sec)
```