



# TAREA HITO 3

MELANIE INGRID VILLCA COPA

# MANEJO DE CONCEPTOS

## 1. Defina que es un lenguaje procedural en MySQL.

Los procedimientos almacenados MySQL, también conocidos como Stored Procedure, se presentan como conjuntos de instrucciones escritas en el lenguaje SQL. Su objetivo es realizar una tarea determinada, desde operaciones sencillas hasta tareas muy complejas. Los procedimientos almacenados MySQL contienen una o más instrucciones SQL además de un procesamiento manipulador o lógico. La característica fundamental de los procedimientos almacenados MySQL es que estos comandos se quedan almacenados y se ejecutan en el servidor o en el motor de bases de datos.

## 2. Defina que es una función en MySQL.

Una función en MySQL es una rutina creada para tomar unos parámetros, procesarlos y retornar en un salida.

## 3. ¿Qué cosas características debe de tener una función? Explique sobre el nombre, el return, parámetros, etc.

- Solamente pueden tener parámetros de entrada IN y no parámetros de salida OUT o INOUT
- Deben retornar en un valor con algún tipo de dato definido
- Pueden usarse en el contexto de una sentencia SQL
- Solo retornan un valor individual, no un conjunto de registros

# MANEJO DE CONCEPTOS

## 4. ¿Cómo crear, modificar y cómo eliminar una función? Adjunte un ejemplo de su uso.

Para crear una función debemos de usar la sentencia CREATE FUNCTION. La sintaxis para crear una función es casi idéntica a la de crear un procedimiento, veamos:

**CREATE FUNCTION** nombre\_función (parametro1,parametro2,...)

**RETURNS** tipoDato

[atributos de la rutina]

<bloque de instrucciones>

Para modificar una función usamos el comando ALTER FUNCTION. Con esta sentencia podemos cambiar los atributos de la función, pero no podremos cambiar el cuerpo. Veamos la sintaxis:

**ALTER FUNCTION** nombre\_funcion

[SQL SECURITY {DEFINER|INVOKER}]

[COMMENT descripción ]

Para eliminar una función usamos el comando DROP FUNCTION. Simplemente especificamos el nombre de la función y esta se borrará de la base de datos. Su sintaxis esta definida de la siguiente forma:

**DROP FUNCTION** nombre\_funcion

# MANEJO DE CONCEPTOS

## 5. Para qué sirve la función **CONCAT** y como funciona en **MYSQL**

- ¿Crear una función que muestre el uso de la función **CONCAT**?
- La función debe concatenar 3 cadenas.

**CONCAT** es una función de cadena compatible con MySQL para combinar o unir dos o más cadenas y devolverlas como un solo valor. El nombre **CONCAT** proviene del verbo concatenación, que significa unir 2 o más entidades juntas.

```
CREATE FUNCTION uso_concat(limiteliteracion
INTEGER) RETURNS TEXT
BEGIN
DECLARE respuesta TEXT DEFAULT '';
DECLARE x INTEGER DEFAULT limiteliteracion;
REPEAT
IF (x % 2 = 0)
THEN
SET respuesta = CONCAT(respuesta, x, ' -AA- ');
ELSE
SET respuesta = CONCAT(respuesta, x, ' -BB- ');
END IF;
SET x = x - 1;
UNTIL x <= 0 END REPEAT ;
RETURN respuesta;
END;
SELECT uso_concat(10);
```

# MANEJO DE CONCEPTOS

## 6. Para qué sirve la función **SUBSTRING** y como funciona en **MYSQL**

¿Crear una función que muestre el uso de la función **SUBSTRING**?

- La función recibe un nombre completo.

- **INPUT: Ximena Condori Mar**

- La función solo retorna el nombre.

- **OUTPUT: Ximena**

La función de subcadena de MySQL se utiliza para extraer una subcadena o una parte de la cadena contra la cadena de entrada. Como sugiere el nombre, la función Substring opera en una cadena de entrada y devuelve una subcadena más pequeña contra las opciones especificadas.

```
create function USAR_SUBSTRING(primero
VARCHAR(30))
returns TEXT
begin
declare str TEXT;
declare x integer default 1;
repeat
set str = SUBSTRING(primero,x,15 );
set x = x + 1;
until x <= 20 END REPEAT;
return str;
end;
Select USAR_SUBSTRING('Ximena');
```

# MANEJO DE CONCEPTOS

## 7. Para qué sirve la función **STRCMP** y como funciona en **MYSQL**

- **¿Crear una función que muestre el uso de la función **STRCMP**?**
- **La función debe comparar 3 cadenas. Y deberá determinar si dos de ellas son iguales.**

La función **STRCMP()** en MySQL se usa para comparar dos strings. Si ambas strings son iguales, devuelve 0, si el primer argumento es más pequeño que el segundo según el orden definido, devuelve -1 y devuelve 1 cuando el segundo es más pequeño que el primero.

```
Select STRCMP('Geeks', 'Geeks') As 'Cmp_Value'
```

# MANEJO DE CONCEPTOS

## 8. Para qué sirve la función **CHAR\_LENGTH** y **LOCATE** y como funciona en **MYSQL**

- **¿Crear una función que muestre el uso de ambas funciones?**

La función **CHAR\_LENGTH()** en MySQL se usa para encontrar la longitud de una string dada (en caracteres). Cuenta el número de caracteres e ignora si los caracteres son de un solo byte o de varios bytes.

La función **LOCATE()** en MySQL se usa para encontrar la ubicación de una substring en una string. Devolverá la ubicación de la primera aparición de la substring en la string. Si la substring no está presente en la string, devolverá 0.

# MANEJO DE CONCEPTOS

## 9. ¿Cual es la diferencia entre las funciones de agresión y funciones creados por el DBA? Es decir funciones creadas por el usuario.

Las funciones de agregación en SQL nos permiten efectuar operaciones sobre un conjunto de resultados, pero devolviendo un único valor agregado para todos ellos. Es decir, nos permiten obtener medias, máximos, etc... sobre un conjunto de valores

Una funciones definidas por el usuario (UDF) es un modo de extender MySQL con una nueva función que funciona como una función nativa de MySQL tal como ABS() o CONCAT() . function\_name es el nombre que debe usarse en comandos SQL para invocar la función.

## 10. ¿Busque y defina a qué se referirá cuando se habla de parámetros de entrada y salida en MySQL?

IN: Es el tipo de parámetro que se usa por defecto. La aplicación o código que invoque al procedimiento tendrá que pasar un argumento para este parámetro. El procedimiento trabajará con una copia de su valor, teniendo el parámetro su valor original al terminar la ejecución del procedimiento.

OUT: El valor de este parámetros pude ser cambiado en el procedimiento, y además su valor modificado será enviado de vuelta al código o programa que invoca el procedimiento.

INOUT: Es una mezcla de los dos conceptos anteriores. La aplicación o código que invoca al procedimiento puede pasarle un valor a éste, devolviendo el valor modificado al terminar la ejecución. En caso de resultarte confuso, echa un ojo al ejemplo que verás más adelante.



# PARTE PRÁCTICA

## 11. Crear la siguiente Base de datos y sus registros.

```
CREATE DATABASE Defensa_Hito3;
USE Defensa_Hito3;
CREATE TABLE estudiantes
(
    id_est INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombres VARCHAR(50),
    apellidos VARCHAR(50),
    edad INTEGER,
    fono INTEGER,
    email VARCHAR(100),
    direccion VARCHAR(100),
    sexo VARCHAR(10)
);
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email, direccion, sexo)
VALUES ('Miguel', 'Gonzales Veliz', 20, 2832115, 'miguel@gmail.com', 'Av. 6 de Agosto', 'masculino');
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email, direccion, sexo)
VALUES ('Sandra', 'Mavir Uria', 25, 2832116, 'sandra@gmail.com', 'Av. 6 de Agosto', 'femenino');
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email, direccion, sexo)
VALUES ('Joel', 'Adubiri Mondar', 30, 2832117, 'joel@gmail.com', 'Av. 6 de Agosto', 'masculino');
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email, direccion, sexo)
VALUES ('Andrea', 'Arias Ballesteros', 21, 2832118, 'andrea@gmail.com', 'Av. 6 de Agosto', 'femenino');
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email, direccion, sexo)
VALUES ('Santos', 'Montes Valenzuela', 24, 2832119, 'santos@gmail.com', 'Av. 6 de Agosto', 'masculino');
```

# PARTE PRÁCTICA

```
CREATE TABLE materias
(
  id_mat INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
  nombre_mat VARCHAR(100),
  cod_mat VARCHAR(100)
);
INSERT INTO materias (nombre_mat, cod_mat) VALUES ('Introduccion a la Arquitectura','ARQ-101');
INSERT INTO materias (nombre_mat, cod_mat) VALUES ('Urbanismo y Diseno','ARQ-102');
INSERT INTO materias (nombre_mat, cod_mat) VALUES ('Dibujo y Pintura Arquitectonico','ARQ-103');
INSERT INTO materias (nombre_mat, cod_mat) VALUES ('Matematica discreta','ARQ-104');
INSERT INTO materias (nombre_mat, cod_mat) VALUES ('Fisica Basica','ARQ-105');
```

```
CREATE TABLE inscripcion
(
  id_ins INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
  semestre VARCHAR(20),
  gestion INTEGER,
  id_est INT NOT NULL,
  id_mat INT NOT NULL,
  FOREIGN KEY (id_est) REFERENCES estudiantes (id_est),
  FOREIGN KEY (id_mat) REFERENCES materias (id_mat)
);
```

# PARTE PRÁCTICA

```
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion) VALUES (1, 1, '1er Semestre', 2018);  
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion) VALUES (1, 2, '2do Semestre', 2018);  
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion) VALUES (2, 4, '1er Semestre', 2019);  
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion) VALUES (2, 3, '2do Semestre', 2019);  
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion) VALUES (3, 3, '2do Semestre', 2020);  
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion) VALUES (3, 1, '3er Semestre', 2020);  
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion) VALUES (4, 4, '4to Semestre', 2021);  
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion) VALUES (5, 5, '5to Semestre', 2021);
```

# PARTE PRÁCTICA

## 12. Crear una función que genere la serie Fibonacci.

```
CREATE FUNCTION seriefibonacci(limite INT)
RETURNS TEXT
BEGIN DECLARE J TEXT DEFAULT "";
DECLARE suma INT DEFAULT 0;
DECLARE a INT DEFAULT 0;
DECLARE b INT DEFAULT 1;
DECLARE cont INT DEFAULT 0;
WHILE cont < limite DO
SET J = CONCAT(J,a,',');
SET b=a+b;
SET a=b-a;
SET cont=cont+1;
END WHILE ;
RETURN J;
END;
SELECT seriefibonacci(7);
```

```
63 DECLARE cont INT DEFAULT 0;
64 WHILE cont < limite DO
65     SET J = CONCAT(J,a,',');
66     SET b=a+b;
67     SET a=b-a;
68     SET cont=cont+1;
69 END WHILE ;
70 RETURN J;
71 END;
72
73 ✓ SELECT seriefibonacci( limite: 8);
74
```

Output seriefibonacci(8):text x

1 row

1 `seriefibonacci(8)`  
0,1,1,2,3,5,8,13,

# PARTE PRÁCTICA

## 13. Crear una variable global a nivel BASE DE DATOS.

```
SET @LIMIT = 2;
```

```
CREATE FUNCTION serie_fibonacci_vg() RETURNS TEXT  
BEGIN
```

```
    DECLARE J TEXT DEFAULT " ";  
    DECLARE a INT DEFAULT 0;  
    DECLARE b INT DEFAULT 1;  
    DECLARE cont INT DEFAULT 0;
```

```
    WHILE cont<@LIMIT DO
```

```
        SET J = CONCAT(J,a,',');
```

```
        SET b=a+b;
```

```
        SET a=b-a;
```

```
        SET cont=cont+1;
```

```
    END WHILE ;
```

```
    RETURN J;
```

```
END;
```

```
SELECT serie_fibonacci_vg();
```

```
85 DECLARE a INT DEFAULT 0;  
86 DECLARE b INT DEFAULT 1;  
87 DECLARE cont INT DEFAULT 0;  
88 WHILE cont<@LIMIT DO  
89     SET J = CONCAT(J,a,',');  
90     SET b=a+b;  
91     SET a=b-a;  
92     SET cont=cont+1;  
93 END WHILE ;  
94 RETURN J;  
95 END;  
96  
97 ✓ SELECT serie_fibonacci_vg();  
98
```

Output serie\_fibonacci\_vg():text x

1 row

serie\_fibonacci\_vg()

1 0,1,1,2,3,5,8,13,

# PARTE PRÁCTICA

## 14. Crear una función no recibe parámetros (Utilizar WHILE, REPEAT o LOOP).

```
CREATE FUNCTION min_edad_estudiantes() RETURNS INT
BEGIN
RETURN
(
    SELECT min(est.edad)
    FROM estudiantes AS est
);
END;
```

```
CREATE FUNCTION par_impar() RETURNS VARCHAR(150)
BEGIN
    DECLARE str VARCHAR(150) DEFAULT "";
    DECLARE x INTEGER DEFAULT 0;
    DECLARE y INTEGER DEFAULT 0;
    SET y = min_edad_estudiantes();
    WHILE x <= y DO
    IF (y % 2 = 0)
    THEN
        SET str = concat(str, x, ',');
        SET x = x + 2;
```

# PARTE PRÁCTICA

```
ELSE
    SET str = concat(str, y, ',');
    SET y = y - 2;
END IF;
END WHILE ;
RETURN str;
END;

SELECT par_impar();
```

```
22 IF (y % 2 = 0)
23 THEN
24     SET str = concat(str, x, ',');
25     SET x = x + 2;
26 ELSE
27     SET str = concat(str, y, ',');
28     SET y = y - 2;
29 END IF;
30 END WHILE ;
31 RETURN str;
32 END;
33
34 ✓ SELECT par_impar();
35
```

Output par\_impar():varchar(150) ×

1 row

`par\_impar()`

1 0,2,4,6,8,10,12,14,16,18,20,

# PARTE PRÁCTICA

## 15. Crear una función que determina cuantas veces se repite las vocales.

```
CREATE FUNCTION vocales(par1 text) RETURNS TEXT
```

```
BEGIN
```

```
    DECLARE x INT DEFAULT 1;
```

```
    DECLARE aVeces INT DEFAULT 0;
```

```
    DECLARE eVeces INT DEFAULT 0;
```

```
    DECLARE iVeces INT DEFAULT 0;
```

```
    DECLARE oVeces INT DEFAULT 0;
```

```
    DECLARE uVeces INT DEFAULT 0;
```

```
    DECLARE response TEXT DEFAULT '';
```

```
    DECLARE letra CHAR DEFAULT '';
```

```
    DECLARE limite INT DEFAULT char_length(par1);
```

```
    DECLARE a VARCHAR(5) DEFAULT 'a';
```

```
    DECLARE e VARCHAR(5) DEFAULT 'e';
```

```
    DECLARE i VARCHAR(5) DEFAULT 'i';
```

```
    DECLARE o VARCHAR(5) DEFAULT 'o';
```

```
    DECLARE u VARCHAR(5) DEFAULT 'u';
```

```
        SET uVeces = uVeces + 1;
    END IF;
END IF;
END IF;
END IF;
END IF;
    SET x = x + 1;
END WHILE;
    SET response = concat(a, ': ', aVeces, ', ', e, ': ', eVeces, ', ', i, ': ', iVeces, ', ', o, ': ', oVeces, ', ', u, ': ', uVeces);
    RETURN response;
END;
```

```
SELECT vocales(par1: 'taller de base de datos');
```

Output: vocales('taller de base de datos'):text

1	a:3,e:4,i:0,o:1,u:0



# PARTE PRÁCTICA

```
WHILE X <= limite DO
  SET letra = SUBSTRING(parl, x, 1);
  IF letra = a
  THEN
    SET aVeces = aVeces + 1;
  ELSE IF letra = e
  THEN
    SET eVeces = eVeces + 1;
  ELSE IF letra = i
  THEN
    SET iVeces = iVeces + 1;
  ELSE IF letra = o
  THEN
    SET oVeces = oVeces + 1;
  ELSE IF letra = u
  THEN
    SET uVeces = uVeces + 1;
  END IF;
END IF;
END IF;
END IF;
END IF;
END IF;
```

# PARTE PRÁCTICA

```
    SET x = x + 1;  
END WHILE;  
SET response = concat(a, ':', aVeces, ' ', e, ':', eVeces, ' ', i, ':', iVeces, ' ', o, ':', oVeces, ' ', u, ':', uVeces);  
RETURN response;  
END;
```

```
SELECT vocales('taller de base de datos');
```

# PARTE PRÁCTICA

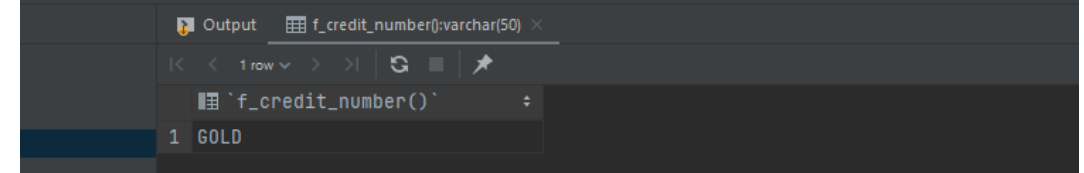
## 16. Crear una función que recibe un parámetro INTEGER.

```
SET @credit_number = 10000;
```

```
CREATE FUNCTION f_credit_number()  
RETURNS VARCHAR(50)  
BEGIN  
    DECLARE respuesta VARCHAR(50) DEFAULT '';  
    CASE  
        WHEN @credit_number > 50000 THEN SET respuesta = 'PLATINIUM';  
        WHEN @credit_number >= 10000 AND @credit_number <= 50000  
        THEN SET respuesta = 'GOLD';  
        WHEN @credit_number < 10000 THEN SET respuesta = 'SILVER';  
        ELSE SET respuesta = 'ERROR';  
    END CASE;  
    RETURN respuesta;  
END;
```

```
SELECT f_credit_number();
```

```
SET @credit_number = 10000;  
  
CREATE FUNCTION f_credit_number()  
RETURNS VARCHAR(50)  
BEGIN  
    DECLARE respuesta VARCHAR(50) DEFAULT '';  
    CASE  
        WHEN @credit_number > 50000 THEN SET respuesta = 'PLATINIUM';  
        WHEN @credit_number >= 10000 AND @credit_number <= 50000 THEN SET respuesta = 'GOLD';  
        WHEN @credit_number < 10000 THEN SET respuesta = 'SILVER';  
        ELSE SET respuesta = 'ERROR';  
    END CASE;  
    RETURN respuesta;  
END;  
  
SELECT f_credit_number();
```



	f_credit_number():varchar(50)
1	GOLD

# PARTE PRÁCTICA

## 17. Crear una función que reciba un parámetro TEXT

CREATE FUNCTION paramettex(cadena VARCHAR(20), position INTEGER)

RETURNS TEXT

BEGIN

DECLARE subCadena TEXT DEFAULT '';

DECLARE limite INT DEFAULT char\_length(cadena);

SET position = 1;

REPEAT

SET subCadena = concat(subCadena,substring(cadena , position, limite),',');

SET position = position + 1;

UNTIL position -1 = limite END REPEAT;

RETURN subCadena;

END;

SELECT paramettex('dbaii', 1);A

```
4 RETURNS TEXT
5 BEGIN
6     DECLARE subCadena TEXT DEFAULT '';
7     DECLARE limite INT DEFAULT char_length(cadena);
8     SET position = 1;
9     REPEAT
10        SET subCadena = concat(subCadena,substring(cadena , position, limite),',');
11        SET position = position + 1;
12    UNTIL position -1 = limite END REPEAT;
13    RETURN subCadena;
14 END;
15
16 SELECT paramettex( cadena: 'dbaii', position: 5);
```

Output paramettex('dbaii', 5);text

	paramettex('dbaii', 5)
1	dbaii,baii,aii,ii,i,



GRACIAS