**INSTRUCCIONES REPETITIVAS**

Los loops permiten iterar un conjunto de instrucciones un número determinado de veces. Para ello, MySQL dispone de tres tipos de instrucciones:

* **LOOP**
* **REPEAT-UNTIL**
* **WHILE**

**LOOP**

Esta estructura repite una o varias operaciones y, después de hacerlas, utiliza una sentencia condicional para comprobar si se cumple una condición que es la que nos va a indicar si se abandona el bucle o no.

Esta instrucción es la menos utilizada puesto que hay que abandonar el bucle con una sentencia LEAVE. Prefiero que uséis las otras. Os pongo un ejemplo por si os la encontráis en internet que sepáis cómo funciona.

Su sintaxis básica es:

[etiqueta:] LOOP

Instrucciones;

END LOOP;

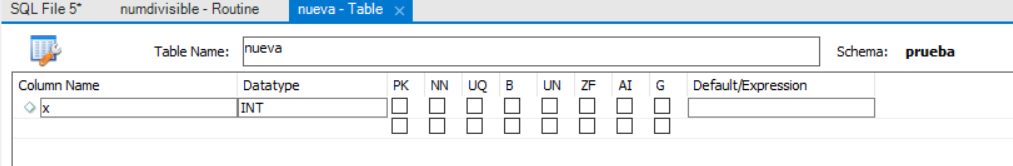
[etiqueta];

Donde la palabra opcional *etiqueta* permite etiquetar el loop para podernos referir a él dentro del bloque.

**EJEMPLO:**

Vamos a crear un procedimiento que va a insertar en una tabla llamada *nueva* el valor de un contador, que va a comenzar con el valor 1 y va a incrementarse en 1 en cada pasada por el bucle. Insertaremos registros del 1 al 10. El contador empezará en uno, lo insertamos en la tabla, toma el valor 2, insertamos….. y así hasta que inserte el 10 y termine.

La tabla *nueva* la tienes que crear tú en el esquema prueba: contendrá un único campo de tipo int. No haces falta que pongas clave.



El procedimiento a escribir será el siguiente:

CREATE PROCEDURE pcrear ()

BEGIN

DECLARE cont INT;

SET cont=1;

BUCLE1: LOOP

INSERT INTO nueva VALUES (cont);

SET cont=cont+1;

IF cont>10 THEN

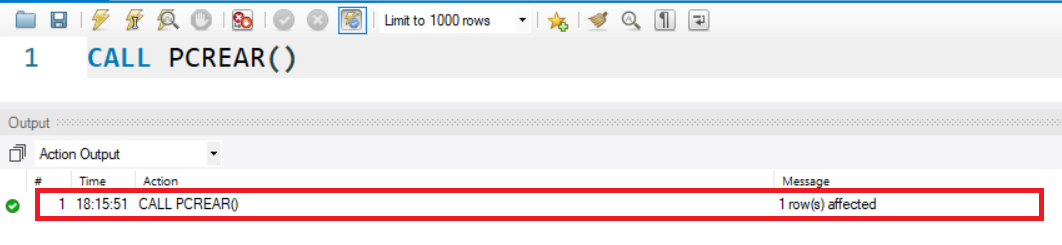
LEAVE BUCLE1;

END IF;

END LOOP;

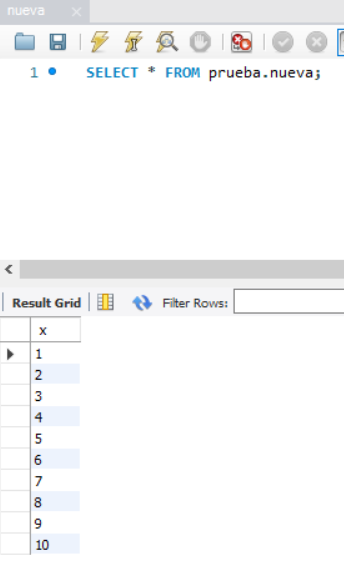
END;

Para ejecutarlo, pondré lo siguiente:



En el caso de los procedimientos puedo poner CALL PCREAR() o sin paréntesis, CALL PCREAR.

Si visualizo el contenido de la tabla, aparecerá lo siguiente:



**REPEAT UNTIL**

Esta estructura se refiere a que vamos a estar repitiendo una o varias operaciones hasta que se cumpla una condición. Si te fijas luego en el ejemplo, la condición se evalúa al final, después de realizar la operación

Su sintaxis general es:

[etiqueta:] REPEAT

Instrucciones;

UNTIL expresión;

END REPEAT [etiqueta]

**EJEMPLO:**

Vamos a crear un procedimiento que haga que se muestren en pantalla los números impares de 0 a 10, utilizando el REPEAT UNTIL.

Crearemos el procedimiento en el esquema *prueba:*

El procedimiento nos quedará de la siguiente forma

CREATE PROCEDURE `numerosimpares`()

BEGIN

DECLARE a int default 0; **🡪 declaro a y la inicializo a 0**

REPEAT

SET a=a+1; **🡪 a, la primera vez, vale 1 porque partía de 0 y le sumo 1**

if MOD(a,2)<>0 THEN **🡪 esta función devuelve el resto. Si es <>0, impar**

SELECT a; **🡪 como es impar, lo muestro**

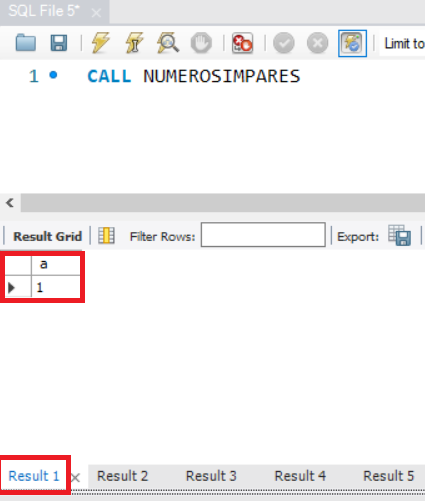
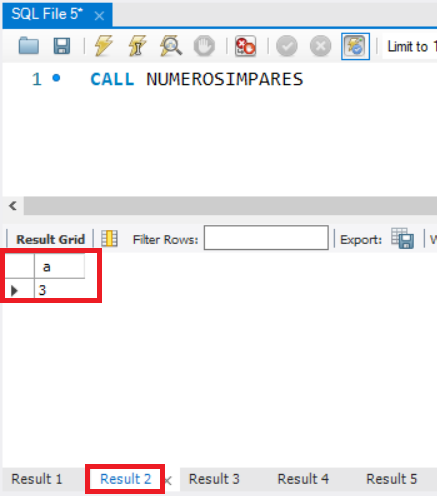
END IF;

UNTIL a>=10 **🡪 aquí comprueba si hemos llegado a 10**

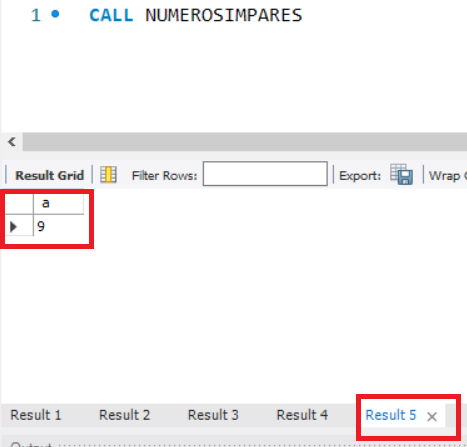
END REPEAT;

END;

El resultado al ejecutar es el siguiente:



Y así sucesivamente hasta el Result 5 que es el 9.



**Los resultados salen en pestañas distintas debido a que cada pasada del bucle hace que se ejecute una sentencia SELECT diferente.**

**WHILE**

Esta estructura se refiere a que, mientras se cumpla una condición, se van a repetir una o varias operaciones. En este caso, la condición se evalúa al principio, antes de entrar en el bucle.

Su sintaxis general es:

[etiqueta:] WHILE condición DO

Instrucciones;

END WHILE [etiqueta];

**EJEMPLO:**

Este ejemplo es igual que el anterior, pero en este caso mostraremos los números pares entre 0 y 10, usando WHILE:

Crearemos el procedimiento también en el esquema *prueba.*

El procedimiento nos quedará de la siguiente forma:

CREATE PROCEDURE pares()

BEGIN

DECLARE i int DEFAULT 1; **🡪 declaro I y lo inicializo a 1**

WHILE i<=10 DO **🡪 mientras i sea menor o igual a 10**

IF MOD(i,2)=0 THEN**🡪 esta función devuelve el resto. Si es =0, par**

SELECT i as RESULTADO; **🡪 como es par, lo muestro**

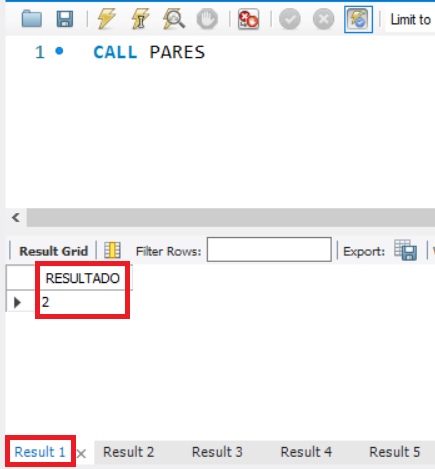
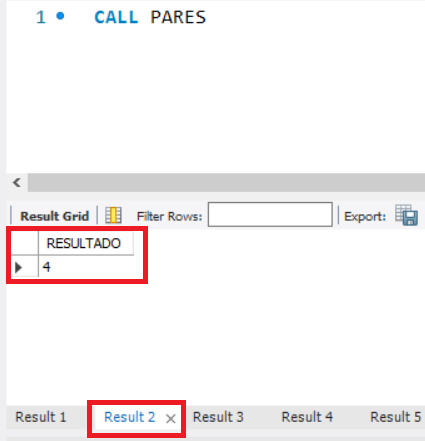
END IF;

SET i=i+1; **🡪 incremento i en 1 y vuelve arriba a ver si ha llegado a 10**

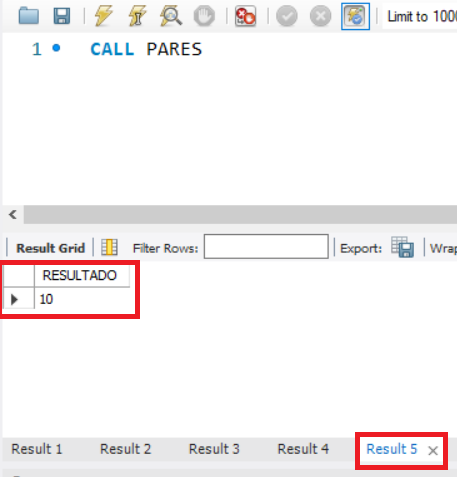
END WHILE;

END

El resultado será el siguiente:



Y así hasta la pestaña Result5 que mostrará el número 10:



**EJERCICIOS:**

1. **Sobre el esquema prueba, crea una función que muestre la suma de los primeros n números enteros, siendo n un parámetro de entrada.**

Es decir, que si llamo a la función SUMA(x int), obtendré:

SELECT SUMA(4) el resultado a obtener será 1+2+3+4=10

SELECT SUMA(6) 🡪 1+2+3+4+5+6

Es decir, que el parámetro x es el que te va a decir cuando tienes que parar de sumar.

Necesitarás dos variables: un contador para ir incrementando de 1 en 1 y otra donde irás sumando dichos valores.

1. **Sobre el esquema prueba, crea un procedimiento que muestre la suma de los términos 1/n , es decir, 1+1/2+1/3+…+1/n, siendo n el parámetro de entrada de entrada del procedimiento. Ten en cuenta que n no puede ser 0. Tienes que hacer esta comprobación.**

En este caso, si creo el procedimiento SUMAFRAC(x int), obtendré:

CALL SUMAFRAC(4) 🡪1+1/2+1/3+1/4=2,083333

CALL SUMAFRAC(6) 🡪1+1/2+1/3+1/4+1/5+1/6=2,45

1. **Sobre el esquema prueba, crea una función que calcule el factorial de un número que introduciremos como parámetro.**

El factorial de un número se calcula de la siguiente manera:

Para calcular el factorial de 3 🡪 3\*2\*1=6

Para calcular el factorial de 5 🡪 5\*4\*3\*2\*1=120

Es decir, si creo la función FACT(x INT) y la ejecuto SELECT FACT(3) el resultado sería 6.