**Clase 15**

**Funciones**

**FUNCIONES**

**Concepto general**

**Definición**

Las funciones customizadas permiten procesar y manipular datos de forma procedural y eficiente. Dichos datos son enviados a través de uno o más parámetros, al momento de invocar la función y retornando un único resultado.

Podemos crear funciones, a la medida de nuestra necesidad, combinando las mismas con funciones ya existentes del lenguaje Mysql, para así obtener los resultados deseados tal como necesitamos.

Para crear funciones personalizadas, necesitamos conocer el lenguaje de programación **SQL**, el cual nos permitirá definir una estructura básica que toda función personalizada debe cumplir para operar como tal.

**Beneficios**

Algunos de los beneficios a destacar:

* mejoran la integridad y seguridad de los datos
* optimizan el rendimiento de la base de datos
* otorgan una mejor lectura del código

**Mejoran la integridad y seguridad de los datos:** crear funciones en el servidor, garantiza que, los resultados de los datos procesados serán iguales para cualquier Stack de Programación. Evitando el error humano programando del lado del cliente cuya lógica pueda diferir entre diferentes lenguajes de programación.

**Optimizan el rendimiento de la base de datos:** para complementar el punto anterior, las funciones customizadas de Mysql envían los datos procesados por el servidor al cliente, minimizando el tráfico de información.

**Otorgan una mejor lectura del código:** cuando creas tus propias funciones puedes normalizar o personalizar su nombre, de acuerdo a la convención de nombres de tu país o de la empresa en donde trabajas, haciendo así más fácil la lectura e interpretación del código SQL que utiliza las mismas.

**SINTAXIS**

Como mencionamos anteriormente, las funciones personalizadas se crean utilizando el lenguaje *de programación* *SQL*.

Este es un lenguaje de programación casi tan poderoso como cualquier otro lenguaje, y cuenta con una sintaxis básica común a todas las funciones personalizadas y/o procedimientos almacenados.

**Sintaxis básicas**

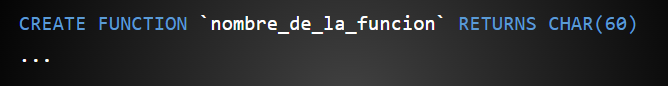
**CREATE FUNCTION `nombre\_de\_la\_funcion`**: esta es la sentencia DDL que nos permite crear una función partiendo desde una ventana de *Scripting*. El nombre de la función va especificado entre comillas simples del tipo **back tick** o, **acento grave**.

**CREATE FUNCTION `nombre\_de\_la\_funcion`**: esta es la sentencia DDL que nos permite crear una función desde una ventana de *Scripting*. El nombre de la misma se especifica entre comillas simples del tipo **back tick** o, **acento grave**.

**RETURNS tipoDeDato**: Las funciones devuelven o retornan usualmente algún tipo de dato. Éste puede ser **boolean**, **char** o **number**, entre otros, según lo que estamos procesando.

Junto a la sentencia **CREATE FUNCTION** debemos especificar qué tipo de datos retornará la misma.

Estructura básica de una función.

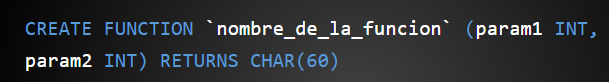
****

**Recepción de parámetros**

**(param1 INT, param2 INT)**: las funciones suelen recibir uno o más parámetros de entrada, los cuales provienen de las columnas de una tabla.

Debemos especificar el nombre de cada parámetro, junto al tipo de dato de éste y, si son más de uno, debemos separarlos por una coma.

Estructura básica de una función con más de un parámetro.



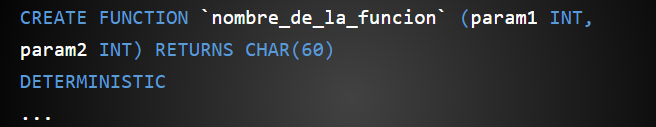
**Tipos de funciones**

Cada función que creamos suele tener diferentes fines. Para optimizar el comportamiento de Mysql al momento de ejecutarla, debemos indicarle qué tipo de función estará ejecutando.

Las principales opciones que permiten definir el tipo de función, son:

* **DETERMINISTIC>** Se considera a una rutina como **DETERMINISTIC** (*determinista*) si esta produce el mismo tipo de resultado que el de sus parámetros de entrada. Se utiliza comúnmente con cadena del texto (*string*) o procesamiento matemáticos, aunque no se limita solo a estos resultados.
* **NO SQL>** Se considera a una rutina como **NO SQL**, cuando esta no utiliza ningún tipo de llamada o invocación de datos a través del lenguaje SQL. O sea, no se llama una sentencia **SELECT**, **UPDATE**, ni cualquier otra sentencia del tipo DML.
* **READS SQL DATA>**Se considera a una rutina como **READS SQL DATA**, cuando esta función sólo leerá datos de una base de datos. No modificará datos a través de las cláusulas **INSERT**, **UPDATE** o **DELETE**, solo leerá información a través de la sentencia **SELECT**.

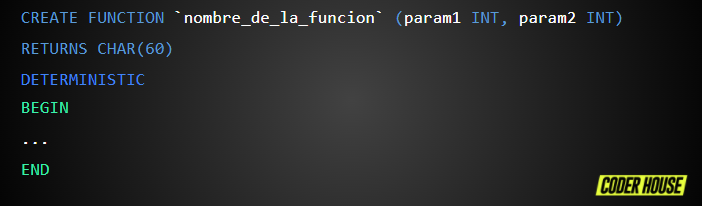
Estructura básica de una función almacenada con más de un parámetro y la implementación del tipo de función que manejará.

****

**Cuerpo de la función**

**BEGIN...END**: El cuerpo de la función se define a través de una estructura de inicio y fin. Y es allí, dentro de este cuerpo donde definiremos, a posteriori, el código que le dará vida a la misma.

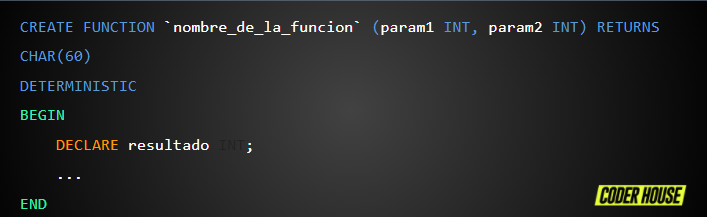
Estructura básica de una función con parámetros y cuerpo principal.

****

**Definir una variable**

**DECLARE resultado\_id INT**: para manejar los diferentes valores dentro de una función, podemos definir una o más variables. Éstas se definen a través de la palabra reservada **DECLARE** seguido del **nombre de la variable** y el tipo de dato que manejará.

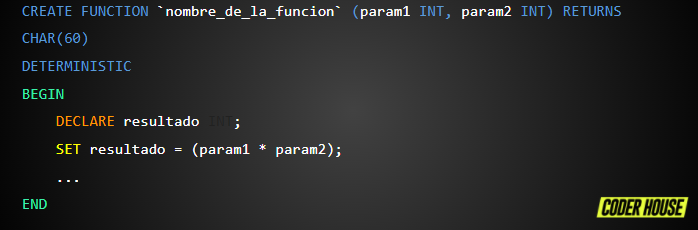
Estructura básica de una función que contiene una variable interna definida.



**Establecer valor de una variable**

**SET resultado\_id = 1 + 1**: cuando deseamos establecer un valor en una variable, debemos utilizar la palabra reservada **SET** seguida de la variable, el operador de definición, y el valor que se le asignará.

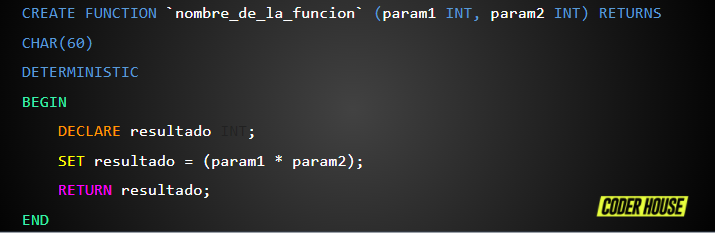
Estructura básica de una función que contiene una variable interna definida y con un valor asignado.



**Retornar un resultado**

**RETURN resultado\_id**: Finalmente, cuando la función realice la o las operaciones pertinentes, devolverá el resultado de las mismas a través de la palabra reservada **RETURN** seguido de la variable que almacena dicho resultado.

Estructura básica de una función que contiene una variable interna definida, con un valor asignado y retornando un resultado.



**Implementación de funciones**

Vamos a generar una función que realice un cálculo matemático, a partir de una superficie que se requiere pintar. Sabemos que un metro cuadrado de pared requiere 100 cm3 de pintura.

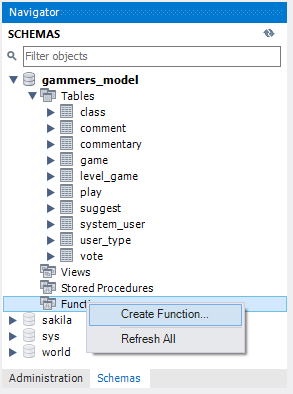
Al invocar a la función, debemos pasarle:

* largo de la pared (*en metros*)
* alto de la pared (*en metros*)
* cantidad de manos de pintura

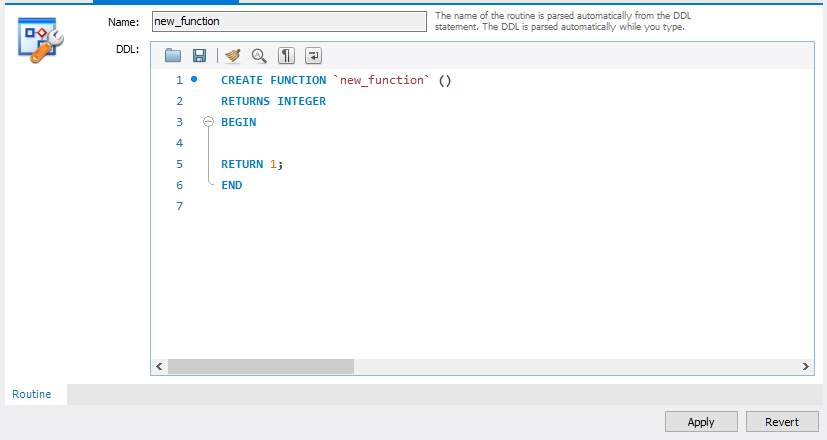
**Crear la función almacenada**

Desde **Mysql Workbench** podemos crear una nueva función simplemente ubicando el cursor **Objeto Functions** del esquema SQL con el cual deseamos trabajar.

Pulsamos el botón secundario del mouse y seleccionamos la opción **Create Function...** en el menú contextual.

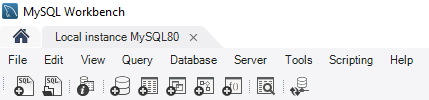


**Mysql Workbench** nos mostrará una ventana para crear una función customizada:



Otra opción desde **Mysql Workbench** es identificando el botón **Create a new function...**, ubicado en la Barra de Herramientas superior.

Al pulsarlo, se abrirá una ventana con la estructura base de una función.

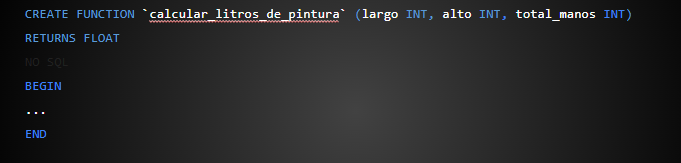


**Definir estructura de la función**

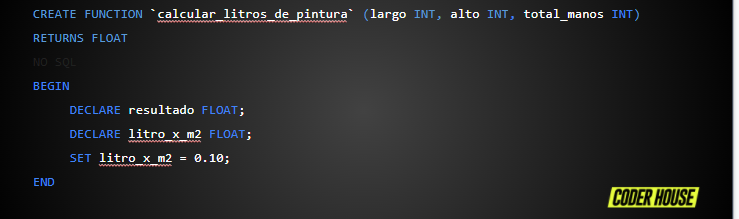
Creamos la función con su nombre, los parámetros que recibirá, y el valor que retornará.

****

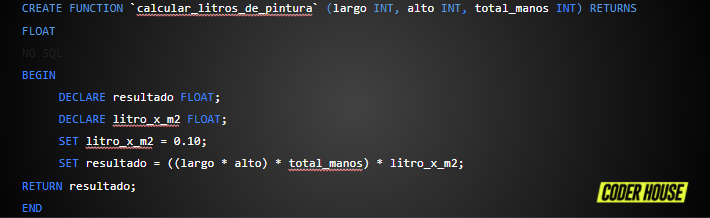
Como no usaremos SQL, definimos el tipo de función como **NO SQL**, y le agregamos el cuerpo a través de **BEGIN** y **END**.

****

Definimos las variables resultado y litro\_x\_m2. A esta última le asignamos el valor 0.10, correspondiente a los **Lts** de pintura por **M2**.

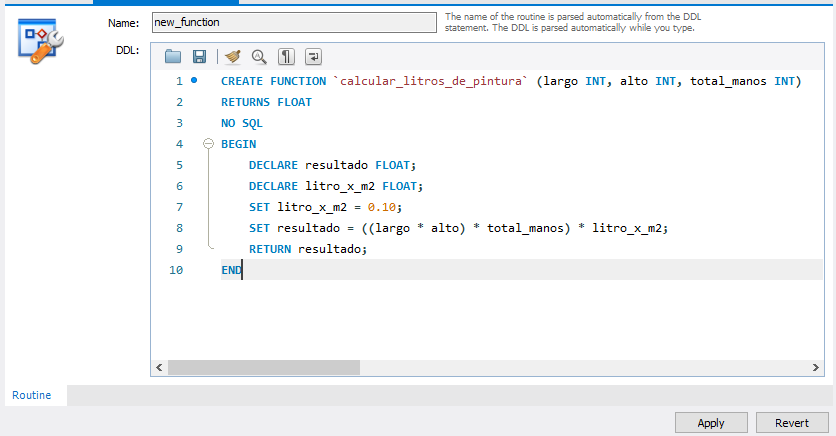


Multiplicamos largo \* alto. Luego, lo multiplicamos por el total de manos de pintura y, finalmente, por el valor del litro x m2. Retornamos el resultado almacenado en la variable homónima, mediante RETURN.

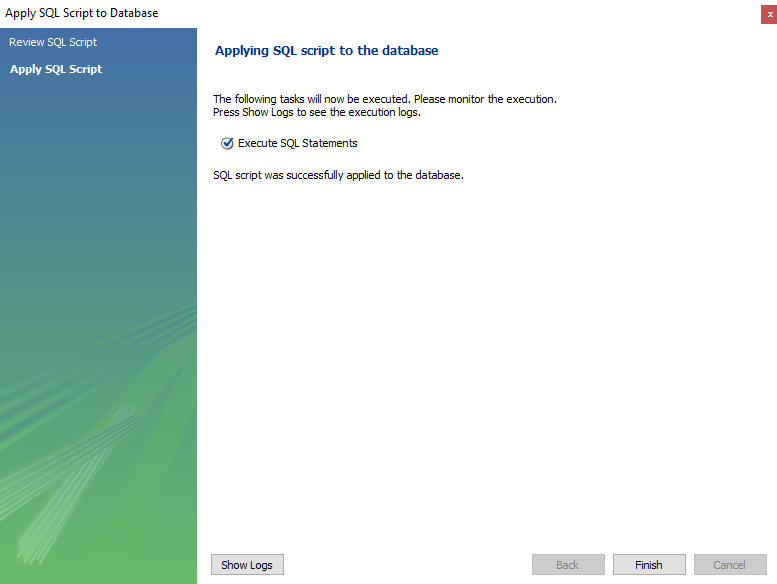
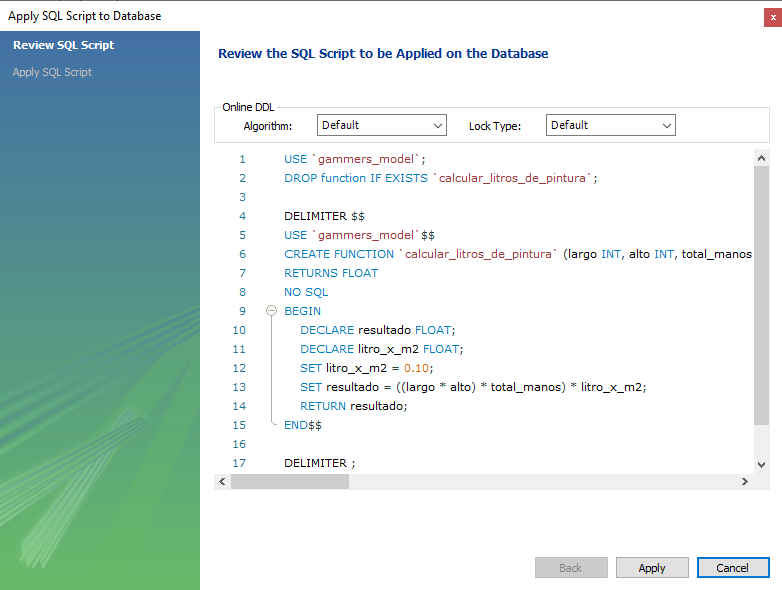


**Guardar la función creada**

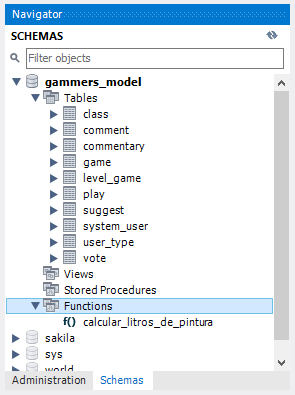
Dentro del editor de funciones del **MySQL Workbench:**

****

Al pulsar el botón **Apply**, **Mysql Workbench** nos pedirá confirmar la estructura del algoritmo, revisando por última vez los parámetros y código de éste.

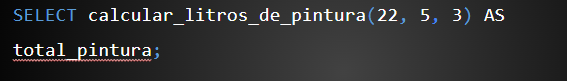
****

Nueva función creada **👇**

****

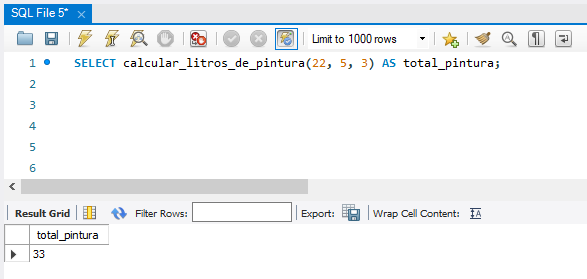
**PROBAR LA FUNCIÓN**

Abramos a continuación una ventana de Script en **Mysql Workbench**. Luego, escribimos la siguiente sentencia:

****

Finalmente, vemos nuestra función personalizada en acción.

Probemos de cambiar los parámetros numéricos, para ir obteniendo otros resultados.



**BUENAS PRÁCTICAS PERFORMANCE TUNING**

1. **Utilizar la función EXPLAIN**

Agregar la expresión EXPLAIN **al comienzo de una consulta leerá y evaluará la consulta**. Si hay expresiones ineficientes o estructuras confusas, EXPLAIN puede ayudarnos a encontrarlas. Luego, puedes ajustar la redacción de la consulta para evitar exploraciones de tablas no intencionadas u otros impactos en el rendimiento.

1. **Evitar usar una función después del FROM en el WHERE**

Por ejemplo:

**SELECT\* FROM TABLA WHERE UPPER(COL1) = '123'**

La notación UPPER crea una función, que debe operar durante la operación SELECT. Esto duplica el trabajo que está haciendo la consulta y es mejor evitarlo si es posible.

**3. Evitar utilizar funciones sobre columnas indexadas**

El uso de funciones sobre columnas indexadas anula el propósito del índice. Supongamos que deseamos obtener datos donde los primeros dos caracteres del código del cliente son AK,

No escribir:

**SELECT** columnas **FROM** Tabla **WHERE** **LEFT** (id\_cliente,2)='AK'

Si podemos escribirlo de la siguiente forma:

**SELECT** columnas **FROM** Tabla **WHERE** id\_cliente **LIKE** 'AK%'

La cual hará uso del índice que da como resultado un tiempo de respuesta más rápido.