Zoe Jimenez

DH  CTD

Base de datos ii

Índice

* [Stored Procedures](#SP)
* Funciones almacenadas
* [Funciones MYSQL](#FuncionesMYSQL)

Funciones

A la hora de consultar la base de datos, solo se ve el nombre de la función —o SP—, si necesita parámetros de entrada —input— o no, y el resultado —output—. No vemos qué sucede internamente, ya que se encuentran encapsuladas.

**Función almacenada**

Es una rutina creada para tomar uno o más parámetros, realizarles un procedimiento y retornar los resultados en un output.

* Pueden incluir parámetros solamente de entrada.
* Deben retornar un valor con algún tipo de dato definido.
* Solo retornan un valor individual, no un conjunto de registros. A esto se le llaman resultados escalares.

Por lo general se usan para hacer cálculos sobre los datos, obteniendo **datos derivados.**

**Estructura**

|  |  |
| --- | --- |
| CREATE FUNCTION | Seguido el nombre de la función |
| RETURN | Se indica el tipo de dato retornado, y el tipo de función |
| BEGIN | Indica el inicio del código SQL |
| RETURN | Retorna el bloque de instrucciones SQL |
| END | Indica el final del código SQL |
| DROP FUNCTION [IF EXIST] nombre() | Elimina una función, se requiere privilegio de ALTER ROUTINE |

**CREATE FUNCTION** nombre()

**RETURNS** [tipo de dato] [característica]

**BEGIN**

**RETURN** – instrucciones SQL;

**END**

**Características**

Las características disponibles son:

|  |  |
| --- | --- |
| DETERMINISTIC | Indica que siempre devuelve el mismo resultado cuando se usan los mismos parámetros de entrada. |
| NOT DETERM… | Indica que no siempre devuelve el mismo resultado, aunque tengo los mismos parámetros. |
| CONTAINS SQL | Indica que contiene sentencias SQL, pero no sentencias de manipulación de datos. |
| NO SQL | Indica que no contiene sentencias SQL. |
| READS SQL DATA | Indica que contiene sentencias de lectura de datos (ej: SELECT). |
| MODIFIES SQL DATA | Indica que modifica los datos de la base y que contiene sentencias como INSERT, UPDATE O DELETE. |

**Aclaraciones**

Si no queremos especificar una característica a la función usar comando:

**SET GLOBAL log\_bin\_trust\_function\_creators = 1;**

Trabajando sobre query de trabajo usar:

**DELIMITER**

**Variables**

**Declaración**

* Dentro de una F se permite declarar variables (elementos que almacenan datos y pueden ir cambiando a lo largo de la ejecución).
* Se declara después de la cláusula BEGIN y antes del bloque de instrucciones SQL.
* Opcional; definir un valor inicial mediante clausula DEFAULT.

Sintaxis:

**DECLARE nombre TIPO\_DE\_DATO [DEFAULT valor];**

**Ej:**

**DECLARE apellido STRING DEFAULT Jimenez;**

**Asignación de valores**

Se utiliza la cláusula SET, y solo pueden contener valores escalares (un solo valor).

Sintaxis:

SET nombre = expresión;

Ej:

**CREATE FUNCTION** agregar-IVA(precio-sin-impuestos **DOUBLE**(10,12))

**RETURNS DOUBLE**(10,12) **DETERMINISTIC**

**BEGIN**

**DECLARE** **IVA** **INT DEFAULT** 21;

**RETURN** ((precio-sin-impuestos \* **IVA** ) / 100) + precio-sin-impuestos;

**END**

**Parámetros**

* Son variables por donde se envían y reciben datos de programas clientes.
* Se definen dentro de la cláusula CREATE.
* Las F pueden tener uno, varios o ningún parámetro de entrada.
* No pueden ingresarse parámetro del tipo OUT o INOUT

|  |  |  |
| --- | --- | --- |
| Parámetro | Tipo | Función |
| IN | Entrada | Recibe datos |

**IN**

Es un parámetro de entrada de datos y se usa para recibir valores. Es definido por defecto cuando no se especifica su tipo.

Sintaxis:

CREATE FUNCTION nombre(IN param1 TIPO, IN param2 TIPO)

Ej:

CREATE FUNCTION nombre(IN id-usuario INT)

BEGIN

-- Instrucciones SQL

**END**

Ejecución:

SELECT \*, nombre(idUsuario) FROM usuarios;

Ejemplos:

CREATE FUNCTION nombre\_completo(nombre VARCHAR(45), apellido VARCHAR(45))

RETURNS VARCHAR(90) DETERMINISTIC

BEGIN

RETURN CONCAT(nombre, ‘ ’, apellido);

**END**

SELECT idUsuario, nombre\_completo(nombre, apellido) FROM usuarios;

SELECT legajo, nombre\_completo(nombre, apellido) FROM empleados;

**CREATE FUNCTION** categoria\_sueldo(sueldo **DOUBLE**)

**RETURNS VARCHAR(15) DETERMINISTIC**

**BEGIN**

**DECLARE** categoria **VARCHAR(15)**;

**CASE WHEN** sueldo < 200 **THEN SET** categoria = ‘Bajo’;

**WHEN** sueldo < 1000 **THEN SET** categoria = ‘Promedio’;

**ELSE SET** categoria = ‘Alto’; **END** **CASE;**

**RETURN** categoría;

**END**

SELECT legajo, sueldo, categoria\_sueldo(sueldo) FROM empleados;

**Stored Procedure**

Son un conjunto de instrucciones SQL que se almacenan, compilan y ejecutan dentro del servidor de bases de datos.

Pueden incluir parámetros de entrada y salida, devolver resultados tabulares o escalares, mensajes para el cliente e invocar instrucciones DDL y DML.

Se los suele utilizar para definir la lógica del negocio dentro de la base de datos y reducir la necesidad de codear la lógica en programas clientes.

Estructura

|  |  |
| --- | --- |
| DELIMITER | Se escribe seguida de una combinación de símbolos que no serán utilizados dentro del SP. |
| CREATE PROCEDURE | Seguido el nombre del procedimiento. |
| BEGIN | Indica el inicio del código SQL |
| END | Indica el final del código SQL, y seguido debe escribirse la combinación de símbolos definidos en DELIMITER. |
| DROP PROCEDURE [IF EXISTS] | Elimina un SP. Requiere privilegio ALTER ROUTINE. |

**Variables**

[Más información](#Variables)

**Parámetros**

* Son variables por donde se envían y reciben datos de programas clientes.
* Se definen dentro de la cláusula CREATE.
* Los SP pueden tener uno, varios o ningún parámetro de entrada, igualmente para los parámetros de salida.

|  |  |  |
| --- | --- | --- |
| Parámetro | Tipo | Función |
| IN | Entrada | Recibe datos |
| OUT | Salida | Devuelve datos |
| INOUT | Entrada-Salida | Recibe y devuelve datos |

* **IN**

Es un parámetro de entrada de datos y se usa para recibir valores. Es definido por defecto cuando no se especifica su tipo.

Sintaxis:

CREATE PROCEDURE nombre(IN param1 TIPO, IN param2 TIPO)

Ej:

DELIMITER $$

CREATE PROCEDURE nombre(IN id-usuario INT)

BEGIN

-- Instrucciones SQL

END $$

Ejecución:

CALL nombre(2)

* **OUT**

Es un parámetro de salida de datos y se usa para devolver valores.

Sintaxis:

CREATE PROCEDURE nombre(OUT param1 TIPO, OUT param2 TIPO)

Ej:

DELIMITER $$

CREATE PROCEDURE nombre(OUT salario FLOAT)

BEGIN

SET salario = 25700.50;

END $$

Ejecución:

CALL nombre(@salario);

SELECT @salario; -- Instrucciones SQL;

* **INOUT**

Es un parámetro de entrada y salida de datos. Puede recibir valores y devolver los resultados en la misma variable.

Sintaxis:

CREATE PROCEDURE nombre(INOUT param1 TIPO, INOUT param2 TIPO)

Ej:

DELIMITER $$

CREATE PROCEDURE nombre(INOUT aumento FLOAT)

BEGIN

SET aumento = aumento + 25700.50;

END $$

Ejecución:

SET @salario = 2000.0;

CALL nombre(@salario);

SELECT @salario; -- Instrucciones SQL;

Bloque de sentencias

Dentro del bloque BEING… END, además de definir variables su pueden escribir distintas sentencias para los SP:

* Sentencias de control de flujo
* Cursores
* Manejo de errores
* **Sentencias de control de flujo**

Cuando necesitamos aplicar condiciones y/o bucles en nuestro programa, MYSQL nos brinda las siguientes sentencias:

|  |  |
| --- | --- |
| **CASE** | Se utiliza para condicionales complejas. |
| **IF** | Se utiliza para condicionales simples. |
| **ITERATE** | Vuelve a empezar una iteración. Solo se utiliza en LOOP, REPEAT y WHILE. |
| **LEAVE** | Se utiliza para salir de iteraciones. |
| **LOOP** | Bucle que se ejecuta X cantidad de veces. |
| **REPEAT** | Bucle que se ejecuta hasta que se cumple una condición. |
| **RETURN** | Se utiliza para retornar un valor en una función o procedimiento. |
| **WHILE** | Bucle que se ejecuta mientras se cumpla la condición. |

Ej:

**DELIMITER $$**

**CREATE FUNCTION** sp\_nombre\_funcion(**IN** id\_usuario INT) **RETURNS** INT

**BEGIN**

**DECLARE** **v** INT;

**SET** v =id\_usuario;

**CASE** v

**WHEN** v =1 **THEN SET** v =2;

**WHEN** v =3 **SET** v =4;

**END CASE**;

nombreLoop1: **LOOP**

**SET** v = v +1;

**IF** v <=3 **THEN ITERATE** nombreLoop1;

**ELSEIF** v > 3 **LEAVE** nombreLoop1;

**END IF**;

**END LOOP** nombreLoop1;

**RETURN** v ;

**END $$**

**CREATE FUNCTION** sp\_nombre\_funcion() **RETURNS** INT

**BEGIN**

**DECLARE v** INT**;**

**SET** v = 1;

**REPEAT**

**SET** v = v +1;

**UNTIL** v = 1000 **END REPEAT;**

**WHILE v > 0 DO**

**SET** v = v -1;

**END WHILE;**

**RETURN** v**;**

**END $$**

**Cursor**

**-------------------------------------------------------------------**

**Funciones MYSQL**

* Funciones de cadenas o textos
* Funciones numéricas
* Funciones de fecha
* Funciones de ventana
* Funciones de información
* Funciones almacenadas
* **Funciones de cadenas o textos**

|  |  |
| --- | --- |
| **CHARACTER\_LENGTH** | Devuelve la longitud de la cadena en caracteres. |
| **CONCAT** | Junta 2 o más cadenas de texto. |
| **FORMAT** | Formatea un numero a un formato como ”XXX,XXX.XX”, redondeado a un número especifico de posiciones decimales. |
| **INSERT** | Inserta una cadena dentro de una cadena en la posición especificada y para un cierto número de caracteres. |
| **LOCATE** | Devuelve la posición de la primera aparición de una subcadena en una cadena. |
| **LOWER** | Convierte una cadena en minúsculas. |
| **UPPER** | convierte una cadenas en mayúsculas. |
| **REPLACE** | Reemplaza todas las apariciones de una subcadena dentro de una cadena, con una nueva subcadena. |
| **RIGHT** | Extrae varios caracteres de una cadena comenzando desde la derecha. |
| **LEFT** | Extrae varios caracteres de una cadena comenzando desde la izquierda. |
| **SUBSTR** | Extrae una subcadena de una cadena comenzando en cualquier posición. |

* **Funciones numéricas**

|  |  |
| --- | --- |
| **GREATEST** | Devuelve el mayor valor de la lista de argumentos. |
| **LEAST** | Devuelve el valor mas pequeño de la lista de argumentos. |
| **MOD** | Devuelve el resto de un numero dividido por otro número. |
| **POW** | Devuelve el valor de un numero elevado a la potencia de otro número. |
| **ROUND** | Redondea un numero a un número especifico de decimales. |
| **SQRT** | Devuelve la raíz cuadrada de un número. |
| **TRUNCATE** | Trunca un numero al numero especificado de posiciones decimales. |

* **Funciones de fechas**

|  |  |
| --- | --- |
| **ADDDATE** | Agrega un intervalo de hora/fecha a una fecha y luego la devuelve. |
| **CURDATE** | Devuelve la fecha actual. |
| **DATEDIFF** | Devuelve el número de días entre dos valores de fecha. |
| **DATE\_FORMAT** | Formatea una fecha. |
| **EXTRACT** | Extra una parte de una fecha determinada. |
| **LAST\_DAY** | Extrae el ultimo día del mes para una fecha determinada. |
| **SEC\_TO\_TIME** | Devuelve un valor de tiempo basado en los segundos especificados. |

* **Funciones de ventana**

Realiza un calculo en un conjunto de filas de la tablas que de alguna manera están relacionadas con la fila actual. Comparable con AVG, COUNT, etc.

No hace que las filas se agrupen en una sola fila de salidas. Detrás de escena la función de ventana puede acceder a mas que solo la fila actual del resultado de la consulta.

|  |  |
| --- | --- |
| **FUNCION** | **DEVUELVE** |
| **FIRST\_VALUE** | Primer valor del primer registro de nuestra consulta. |
| **LAG** | Valor anterior del registro actual que estamos mostrando. |
| **LAST\_VALUE** | Valor del último registro de nuestra consulta. |
| **LEAD** | valor siguiente del registro actual que estamos mostrando |
| **NTILE** | Divide la cantidad de resultados por el parámetro que recibe y asigna un grupo a cada uno de los registros |
| **ROW\_NUMBER** | Número de la fila actual dentro de la cantidad de resultados. Los números de filas van desde 1 hasta el número de filas de resultados. |

* **Función de información**

Se utilizan para consultar la información del sistema de la base de datos MYSQL. También para obtener información sobre las conexiones a nuestra base.

|  |  |
| --- | --- |
| **FUNCION** | **DEVUELVE** |
| **CURRENT\_ROLE** | Roles activos actuales del usuario conectado. |
| **CURRENT\_USER** | Nombre de usuario y el nombre de host autenticados. |
| **DATABASE** | Nombre de la base de datos actual. |
| **LAST\_INSERT\_ID** | Valor de la columna AUTOINCREMENT para el último INSERT. |
| **ROW\_COUNT** | Numero de filas actualizadas. |
| **VERSION** | Cadena que indica la versión del servidor MYSQL. |