PROCEDIMIENTOS Y FUNCIONES

Un **procedimiento** o **función** es un programa que se almacena físicamente en una tabla dentro del sistema de bases de datos. Este programa está hecho con un lenguaje propio de cada SGBD y esta compilado, por lo que la velocidad de ejecución será muy rápida.

Ventajas:

- **Seguridad:** Cuando llamamos a un procedimiento almacenado, este deberá realizar todas las comprobaciones pertinentes de seguridad y seleccionará la información lo más precisamente posible, para enviar de vuelta la información justa y necesaria y que por la red corra el mínimo de información, consiguiendo así un aumento del rendimiento de la red considerable.
- **Rendimiento:** el SGBD, en este caso MySQL, es capaz de trabajar más rápido con los datos que cualquier lenguaje del lado del servidor, y llevará a cabo las tareas con más eficiencia. Solo realizamos una conexión al servidor y este ya es capaz de realizar todas las comprobaciones sin tener que volver a establecer una conexión.
- Otra ventaja es la posibilidad de **separar la carga del servidor**, ya que si disponemos de un servidor de base de datos externo estaremos descargando al servidor web de la carga de procesamiento de los datos.
- **Reutilización:** el procedimiento almacenado podrá ser invocado desde cualquier parte del programa, y no tendremos que volver a armar la consulta a la BD cada qué vez que queramos obtener unos datos.

Desventajas:

- El **programa** se guarda en la BD, por lo tanto si se corrompe y perdemos la información también perderemos **nuestros procedimientos**. Esto es fácilmente subsanable llevando a cabo una buena política de respaldos de la BD.

Por lo tanto es **recomendable usar procedimientos almacenados** siempre que se vaya a hacer una aplicación grande, ya que nos facilitará la tarea bastante y nuestra aplicación será más rápida.

Más información acerca en:

http://dev.mysql.com/doc/refman/5.7/en/create-procedure.html

SINTAXIS PROCEDURE

Para la **creación de un procedimiento almacenado**:

```
DELIMITER //

CREATE PROCEDURE nombre ([parametro[,...]])
-- Opciones
BEGIN

-- Declaración de variables


-- Código Fuente


END//

DELIMITER ;
```

Los **parámetros** de los procedimientos pueden ser de entrada, salida o ambos:

```
parámetro_proc:
        [IN | OUT | INOUT] nombre_parametro tipo_dato
```

Para la **llamada a ese procedimiento almacenado**:

```
CALL nombre(parametro1, @parametro2, ...);

SELECT @parametro2;
```

Las **opciones** son idénticas en los procedimientos y en las funciones:

- **LANGUAGE SQL:** el único valor posible de momento. Se prevé que en el futuro se puedan usar otros lenguajes de programación.
- **[NOT] DETERMINISTIC:** un SP está considerado determinista cuando siempre retorna el mismo resultado con los mismos parámetros. Los SP que dependen de una tabla de la base de datos son no deterministas obviamente. Por defecto los SP son no deterministas. Sin embargo, los SP que sí lo son pueden ejecutarse de una manera mucho más eficiente ya que, por ejemplo, los resultados se pueden almacenar en una cache. De cualquier manera, actualmente esta opción es ignorada por MySQL.
- **SQL SECURITY DEFINER o INVOKER:** el modo SQL SECURITY especifica los privilegios de acceso al SP.
- **COMMENT 'texto':** el comentario se almacena con el SP a modo de documentación.

SINTAXIS FUNCTION

Para la **creación de una función**:

```
DELIMITER //

CREATE FUNCTION nombre ([parametro[,...]]) RETURNS tipo_dato
-- Opciones
BEGIN

-- Declaración de variables


-- Código Fuente


END//

DELIMITER ;
```

Los **parámetros** de las funciones son siempre de entrada:

```
parámetro_fun:
        nombre_parametro tipo_dato
```

Para la **llamada a esa función**:

```
SELECT nombre(parametro);
```

SINTAXIS COMÚN

Las **Store Procedure** (***SP***) están compuestas principalmente de un conjunto de instrucciones SQL ordinarias. Sin embargo hay una serie de instrucciones adicionales que permiten control de flujo, alternativas, cursores... Las principales diferencias entre procedimientos y funciones son:

	PROCEDURE	FUNCTION
Llamada	Solo con CALL	Posible en todas las instrucciones SQL (SELECT, UPDATE, ...)
Retorno	Puede retornar uno o más SELECT	Retorna un valor único de un tipo determinado.
Parámetros	Permite por valor y por referencia (IN, OUT, INOUT)	Solo parámetros por valor.
Instrucciones permitidas	Todas las SQL	No se puede acceder a tablas
Llamadas a otras funciones o procedimientos	Puede llamar a otros procedimientos y/o funciones	Solo puede llamar otras funciones

Las reglas generales para escribir SP son:

- **Punto y coma.** Para separar las diferentes instrucciones se utiliza el carácter ';'.
- **BEGIN-END.** Las instrucciones que no se encuentran entre dos palabras reservadas (ej. IF, THEN, ELSE) han de ponerse entre un BEGIN y un END.
- **Salto de línea.** Los saltos de línea son considerados como espacios en blanco.
- **Variables.** Las variables locales y los parámetros se acceden sin un carácter @ al principio. El resto de variables SQL deben comenzar con @.
- **Mayúsculas/minúsculas.** MySQL no las distingue al definir el nombre del SP.
- **Caracteres especiales.** Evitar el uso de caracteres fuera del código ASCII (*acentos, ñ,...*). A pesar de estar permitidos, pueden aparecer problemas al administrar la base de datos.
- **Comentarios.** Los comentarios se introducen con '--' y se extienden hasta el final de la línea.

VARIABLES

Se pueden **declarar variables** locales dentro de los procedimientos y funciones. La sintaxis es:

```
DECLARE nombre_var1, nombre_var2, .... tipo_dato [DEFAULT valor];
```

Se debe definir el tipo de datos para todas las variables. Se inicializan con NULL, a menos que se les dé un valor explícito. Hay que tener cuidado de no dar a las variables los mismos nombres que columnas o tablas de la base de datos ya que, a pesar de estar permitido, puede dar lugar a errores muy difíciles de rastrear.

La **asignación de variables** la podemos hacer de dos formas diferentes. Utilizando SET o SELECT…INTO:

```
SET cuenta=10;

SELECT 2*7 INTO cuenta;

SELECT COUNT(*) INTO cuenta
FROM articulos;

SELECT titulo, subtitulo INTO mititulo, misubtitulo
FROM libros
WHERE tituloID = 3;
```

CONTROL DEL FLUJO

Existen instrucciones para **controlar el flujo** del programa.

Entre las que destacamos:

- IF-THEN-ELSE
- CASE
- WHILE-DO
- REPEAT-UNTIL

La sintaxis de cada una de ellas la podemos encontrar en la referencia de MySQL:

http://dev.mysql.com/doc/refman/5.7/en/flow-control-statements.html

FUNCIONES PREDEFINIDAS EN EL SISTEMA

Hay una serie de **funciones predefinidas** en MySQL que podemos utilizar para nuestros procedimientos o funciones que serán de gran utilidad. Las funciones se clasifican en:

- Funciones para cadenas de caracteres
- Funciones numéricas
- Funciones de fecha y hora
- …

La sintaxis de cada una de ellas la podemos encontrar en la referencia de MySQL:

http://dev.mysql.com/doc/refman/5.7/en/functions.html