Licenciatura en Sistemas - Bases de Datos I – 2018

Titular: Ing. Federico Ribeiro federico.ribeiro@gmail.com
Ayudante: Lic. Leandro Ríos leandro.rios.unla@gmail.com

Resumen Stored Procedures MySQL

Manual de referencia de MySQL: https://dev.mysql.com/doc/refman/8.0/en/

Un Stored Procedure (SP) en MySQL es un procedimiento (análogo a los que definimos en los lenguajes de programación) que se almacena en la base de datos y podemos invocarlo desde un programa externo, un trigger o incluso desde otro SP.

Ventajas:
Mejora la performance de las aplicaciones: Al estar precompliados en el servidor, se evita toda la operación de parseo y preparación del query plan que ocurre con las sentencias enviadas como strings directamente al servidor.
Disminuye el tráfico de red: por las mismas causas que la ventaja anterior, al no ser necesario enviar sentecias SQL complejas a través de la red.
Son reutilizables y no dependen de la tecnología con la que se desarrolle la aplicación: nos permiten crear, si es necesario una API para acceder a nuestra base de datos. Detrás de dicha API podemos realizar validaciones que el modelo relacional no permite. Además, permiten una mejor granularidad en el manejo de la seguridad, ya que se puede asignar distintos permisos de ejecución a los distintos SP, sin dar acceso a las tablas de la base de datos si es necesario.

Desventajas:
Aumentan la carga de memoria y procesamiento del motor de BD.
No son tan flexibles como los procedimientos de los lenguajes de programación.
El debugging de los SP en MySQL es complicado por la escasa infraestructura que ofrece el motor de BD para realizarlo. Se debe recurrir a herramientas externas (que no siempre ofrecen la calidad necesaria o no corren en todas las plataformas)

Resumen de sintaxis

Comenzamos con un SP sencillo: vamos a devolver todos los registros de la tabla clientes:

```
DELIMITER $$
CREATE PROCEDURE traerClientes()
BEGIN
  SELECT * FROM clientes;
```

```
END $$
DELIMITER ;
```

DELIMITER es necesario para poder incluir varios punto y coma (uno por cada sentencia SQL) dentro del cuerpo del SP .Cambiamos temporariamente el delimitador por defecto (el punto y coma) por cualquier otro a fin de que el motor entienda que el SP termina con el nuevo delimitador. De lo contrario dejaría de leer al encontrar el primer punto y coma, causando un error. Al finalizar, volvemos a reponer el delimitador por defecto.

CREATE PROCEDURE es el comando de creación del procedimiento, seguido por el nombre del mismo.

Para invocar un stored procedure utilizamos la sentencia CALL:

```
CALL traerClientes();
```

que nos devolverá un resultset con todos los clientes.

Variables

Podemos emplear variables dentro de los SP:

```
DECLARE nombre_variable tipo(tamaño) DEFAULT valor
```

El tipo de la variable puede ser cualquiera de los tipos primitivos que soporta MySQL, y entre paréntesis el tamaño de la misma. Al declararse la variable, MySQL le asigna el valor NULL, a menos que hayamos indicado otra cosa por medio de DEFAULT.

```
DECLARE codigo_cliente VARCHAR(6) default "000000";
```

Para asignarlas usamos SET:

```
SET codigo_cliente="123456";
```

o podemos asignarles el resultado de un select (usando INTO):

```
SELECT codigo INTO codigo_cliente from cliente where idCliente=1;
```

El ámbito y el alcance de las variables es local; se definen dentro del bloque BEGIN… END que delimita el SP y dejan de existir una vez alcanzada la sentencia END. una variable que comienza con el símbolo @ tiene alcance global para la sesión.

Parámetros de los SP

Los parámetros de los SP en MySQL se definen con uno de tres modos: IN, OUT e INOUT. El primero sirve para pasar datos al procedimiento, el segundo para devolverlos y el tercero

cumple ambas funciones. Tanto OUT como INOUT son similares al paso por referencia en los lenguajes convencionales. IN, en cambio es similar al pase por valor. Si no definimos el modo, se establece como IN por defecto.

Un parámetro se define de la siguiente manera

[IN / OUT / INOUT] nombre_parametro tipo(tamaño)

Tipo y tamaño siguen los mismos lineamientos que para las variables.

```
DELIMITER $$
CREATE PROCEDURE traerClientesPorGenero
(IN genero_cliente VARCHAR(1))
BEGIN
  SELECT * FROM clientes
  WHERE genero = genero_cliente;
END $$
DELIMITER ;
```

Tanto las variables. como los parametros no pueden llamarse igual que las tablas, campos o cualquier otro objeto de la base de datos. Los identificadores deben ser únicos.

Llamamos al SP de la siguiente manera:

```
CALL traerClientesPorGenero('F');
```

```
Supongamos que queremos contar los clientes según el género. Devolveríamos
el resultado como un parámetro de salida:
```

```
DELIMITER $$
CREATE PROCEDURE contarClientesPorGenero
(IN genero_cliente VARCHAR(1), OUT cantidad INT)
BEGIN
  SELECT count(*) INTO cantidad FROM clientes
  WHERE genero = genero_cliente;
END $$
DELIMITER ;
```

```
y para llamarlo hacemos uso de una variable de sesión:
```

```
CALL contarClientesPorGenero('M', @cantidad);
```

```
select @cantidad as cantidad_total; // mostramos
```

Cuidado con las variables globales: como ya sabemos, no se deben utilizar para pasar información entre procedimientos; se debe evitar su uso. Aquí se utiliza con propósito de

demostración, en un caso de uso real, este procedimiento podría ser llamado desde otro que almacenaría el resultado en una variable local y luego lo utilizaría para lo que fuera necesario.

Estructuras de control condicional en SP

Sentencia IF:

```
IF expresion THEN
  sentencias
[ELSEIF expresion THEN
   sentencias]
[ELSE
  sentencias]
END IF;
```

Sentencia CASE:

```
CASE
WHEN expresion1 THEN
  sentencias;
WHEN expresion2 THEN
  sentencias;
...
WHEN expresion3 THEN
  sentencias;
ELSE
  sentencias;
END CASE;
```

Iteración:

Sentencia WHILE:

```
WHILE expresion DO
  sentencias;
END WHILE;
```

Sentencia REPEAT … UNTIL

```
REPEAT
  sentencias;
UNTIL expresion
END REPEAT;
```

Stored Functions

Las Stored Functions (SF) son funciones que definimos y se almacenan en el esquema de la base de datos. Las mismas pueden emplearse como cualquier otra función SQL.

Supongamos que queremos devolver un saludo para cada uno de nuestros clientes. A tal fin definimos una función:

```
DELIMITER $$
CREATE FUNCTION saludo(nombre_cliente TEXT, apellido_cliente TEXT)
RETURNS TEXT
BEGIN
  RETURN CONCAT('Estimado señor ',nombre_cliente,' ',apellido_cliente, ',
tenga Ud. buenos días');
END; $$
DELIMITER ;
```

La definición de funciones es muy similar a la de los stored procedures, con algunas salvedades:

Los parámetros de las funciones simpre se definen en modo IN, pero sus tipos siguen las mismas reglas que los de los SP.

Las funciones siempre retornan algún valor. El tipo del valor de retorno puede ser cualquier tipo primitivo de MySQL.

Las funciones no pueden ejecutar sentencias SQL que modifiquen datos (INSERT, UPDATE, DELETE). Sí pueden utilizar todos los select que sean necesarios.

Pueden utilizarse, igual que en los SP, construcciones de control, de iteración y variables.

Para utilizar la función:

```
select saludo(nombre,apellido) from cliente;
```