Sentencias preparadas (*prepared statements*)

Este tipo de sentencias se basan en el concepto de que cuando un programa cliente (aplicación de escritorio, página web con PHP, ASP.NET, JSP,...) va a realizar una operación SQL (SELECT, INSERT,....) normalmente siempre va a ejecutar la misma sentencia cambiando los datos que se le envían al servidor.

Imaginemos que damos de alta a 10 empleados. Tendríamos que enviar 10 sentencias INSERT con los datos de cada uno de los empleados. El gestor tendrá que analizar esa sentencia por cada empleado, comprobar si está bien escrita y ejecutarla...

Con las sentencias 'prepared', lo que hacemos en enviar al servidor la sentencia SQL sin datos, enviamos la sentencia con parámetros. El gestor la analiza una sola vez para comprobar que no tiene problemas sintácticos y una vez hecho, el cliente sólo tiene que enviar los datos que van a ser ejecutados por la sentencia.

* El uso de sentencias 'prepared' nos va a permitir:
* Una mayor velocidad de ejecución (en el caso de usar procedimientos almacenados, ya no tenemos esta ventaja) porque las sentencias están pre-compiladas.
* Una mayor seguridad, ya que impide la inyección de código (en el caso de usar procedimientos almacenados, ya no tenemos esta ventaja, ya comenté antes este tipo de ataque).
* La posibilidad de crear sentencias SQL dinámicas, es decir, que podemos construir la sentencia como si fuera una cadena a la que vamos añadiendo partes hasta tenerla completa y ejecutarla (esta es la mayor ventaja si estamos usando procedimientos almacenados y necesitamos este tipo de sentencias, pero debemos de tener cuidado con la inyección de código).

Por lo tanto, si estamos haciendo uso de procedimientos, el único sentido de utilizar sentencias prepared es el de crear sentencias dinámicas dentro del mismo.

La sentencia PREPARE es muy recomendada si estamos programando un programa cliente y llamamos múltiples veces al servidor enviando la misma sentencia SQL, cambiando sólo los datos entre cada llamada o si queremos, desde el lado cliente, crear una sentencia SQL dinámicamente.

Proceso para emplear estas sentencias

El proceso para poder emplear este tipo de sentencias es el siguiente (tanto si lo hacemos desde el cliente (php, java,...) como desde un procedimiento almacenado:

* PREPARE: Donde definimos la sentencia SQL con parámetros que queremos ejecutar (por ejemplo, una sentencia INSERT lleva como parámetros los valores a añadir, o una sentencia SELECT lleva parámetros en la parte WHERE).
* Paso de datos a los parámetros que están definidos en la sentencia anterior.
* EXECUTE: Para ejecutar la sentencia.
* DEALLOCATE PREPARE: Para liberar la memoria del servidor de la sentencia preparada.

Ejemplo genérico

**SET** @sentencia = 'SELECT \* FROM TABLA WHERE col1=? AND col2=?';

**PREPARE** stPrepared **FROM** @sentencia; *-- Preparamos la sentencia. El nombre stPrepared se lo damos nosotros. Podemos poner lo que queramos.*

**SET** @valorParam1 = 2; *-- Estamos usando una variable para establecer el valor que se le va a enviar al parámetro. Podríamos pasarlo directamente en la siguiente intrucción.*

**SET** @valorParam2 = 'Angel'; *-- Estamos usando una variable para establecer el valor que se le va a enviar al parámetro. Podríamos pasarlo directamente en la siguiente intrucción.*

**EXECUTE** stPrepared **USING** @valorParam1,@valorParam2;

**DEALLOCATE** **PREPARE** stPrepared; *-- Liberamos la memoria*

Importante

En la sentencia PREPARE sólo podemos emplear variables de sesión o literales (es decir, entre comillas si son cadenas o directamente el valor si son numéricos). No podemos emplear variables locales si estamos dentro de un procedimiento. En la sentencia EXECUTE ... USING sólo podremos emplear variables de sesión.

La sentencia 'PREPARE' es visible a nivel de sesión. Quiere esto decir que si no hacemos un DEALLOCATE, podríamos emplearla en cualquier pestaña SQL de la conexión activa.

Ejemplo aplicado a nuestra base de datos

Veamos un ejemplo aplicado a nuestra base de datos.

Crea un procedimiento de nombre animales\_getListPorFiltro que devuelva la lista de animales filtrados por columnas dinámicamente. Los datos a enviar será el nombre de la columna, el tipo de operación y el valor que debe cumplir. Por ejemplo, 'anhos','=','2'.

USE CIRCO;

**DELIMITER** $$

**CREATE** **PROCEDURE** animales\_getListPorFiltro (p\_nombreColum varchar(20),p\_operacion char(2),p\_valorParam varchar(10))

**COMMENT** 'Devuelve todos los animales que cumplan la condición que se la va pasar como parámetro.'

**BEGIN**

**SET** @v\_consulta = CONCAT('SELECT \* FROM ANIMALES WHERE ',p\_nombreColum,p\_operacion,'?'); *-- Podríamos poner como tercer parámetro directamente el parámetro p\_valorParam*

**SET** @v\_valor = p\_valorParam;

**PREPARE** prepConsulta **FROM** @v\_consulta;

**EXECUTE** prepConsulta **USING** @v\_valor;

**DEALLOCATE** **PREPARE** prepConsulta;

**END** $$

**DELIMITER** ;

**Ejemplo de uso:**

**CALL** animales\_getListPorFiltro('tipo','=','Jirafa');

**CALL** animales\_getListPorFiltro('anhos','>=','3');