4. Rutinas almacenadas

4.1. Rutinas almacenadas en MySQL

Una rutina almacenada es un conjunto de sentencias que pueden ser almacenadas en el servidor con un nombre que se le asigna en el momento de la creación. Las aplicaciones cliente pueden ejecutar las rutinas almacenadas, indicando el nombre de la rutina y pasándole, opcionalmente, los parámetros necesarios. El término rutinas almacenadas hace referencia tanto a los procedimientos almacenados como a las funciones definidas por el usuario.

Cuando se crea una rutina almacenada, el conjunto de instrucciones almacenadas en el servidor queda enlazado y optimizado para su ejecución, lo que supone una mejora en el rendimiento y una reducción del tráfico en la red, aunque a cambio supone un aumento de carga de trabajo en un servidor de bases de datos.

4.1.1. Creación de rutinas almacenadas

El proceso de creación de una rutina almacenada hace que quede asociado a la base de datos que esté activa en ese momento o a la que se indica explícitamente con un nombre calificado como *nombreBD.nombreRutina*. Esto tiene varias implicaciones:

- Cuando se quieran utilizar el procedimiento almacenado o la función, ha de estar activa la base de datos asociada o bien hay que calificar el nombre de la rutina con el nombre de la batallade datos.
- En una base de datos no puede haber dos rutinas almacenadas que tengan el mismo nombre.
- Cuando se borra una base de datos, se borran todos los procedimientos almacenados y todas las funciones asociadas a esa base de datos.

En el caso de existir un conjunto de procedimientos almacenados y de funciones definidas por el usuario que puedan ser de utilidad en más de una base de datos, puede ser útil tener creada una base de datos con una librería de procedimientos almacenados y de funciones definidas por el usuario, en lugar de crear esos procedimientos o funciones en cada una de las bases de datos en las que se va a utilizar.

La información de las rutinas creadas se guarda en el diccionario de datos, al igual que el resto de objetos de las bases de datos. En el caso de MySQL, la información sobre las rutinas almacenadas se puede consultar en las tablas *mysql.proc* y *information_schema.routines*.

Sentencias CREATE PROCEDURE y CREATE FUNCTION

Estas sentencias permiten crear procedimientos almacenados y funciones definidas por el usuario. Sintaxis:

```
CREATE [DEFINER = { usuario |  CURRENT_USER }]
  PROCEDURE nome_procedemento ( [parámetro_procedimiento [,...]])
  [característica ...]  cuerpo_rutina
CREATE [DEFINER = { usuario | CURRENT_USER }]
FUNCTION  nombre_función ([parámetro_función [,...]])
```

```
RETURNS tipo_dato
[característica ...]  cuerpo_rutina
```

- Las cláusulas DEFINER y SQL SECURITY (forma parte de *característica*) especifican el contexto de seguridad en el momento de la ejecución de la rutina. Con DEFINER se puede indicar el nombre del usuario que va a ser considerado el creador de la rutina. Si no se especifica nada, se toma CURRENT_USER, que hace referencia al usuario actual que está creando la rutina.
- Los paréntesis que van después del nombre del procedimiento son obligatorios y contienen la lista de parámetros. No se escribe nada dentro de ellos si el procedimiento no tiene parámetros.
- *parámetro_procedimiento* tiene la forma: [IN | OUT | INOUT] nombre_parámetro tipo_dato
  - IN, OUT e INOUT es el tipo de parámetro.

    Un parámetro de entrada (IN - input) indica que cuando se llame al procedimiento almacenado, hay que escribir en esa posición un valor que se corresponda con el tipo de dato asociado a ese parámetro. El parámetro puede cambiar de valor durante la ejecución del procedimiento, pero su valor no es visible para quien lo llama.

    Un parámetro de salida (OUT - output) indica que en esa posición hay que poner una variable de usuario que almacenará el resultado que devuelve el parámetro, para poder manejarlo posteriormente. Su valor inicial es null.

    Un parámetro de entrada-salida (INOUT) indica que en esa posición hay que poner una variable de usuario que tiene un valor inicial que se pasa como entrada cuando se llama al proceso, y en el que almacenará el resultado que devolverá el parámetro después de ejecutarse el procedimiento.

    En el caso de llamar a un procedimiento desde otro procedimiento o función, también se pueden utilizar variables locales para los parámetros OUT e INOUT, en lugar de variables de usuario.

    El procedimiento almacenado puede devolver resultados mediante los parámetros OUT e INOUT.

    Si no se indica IN, OUT o INOUT, se declara que es de entrada (IN).

  - El nombre del parámetro se puede escribir en minúsculas o mayúsculas indistintamente.
  - El tipo de dato asociado del parámetro puede ser cualquier tipo de dato válido en MySQL.
- *parámetro_función* tiene la forma: *nombre_parámetro tipo_dato*

  El tipo de dato puede ser cualquier tipo de dato válido en MySQL.
- La cláusula RETURNS sólo puede utilizarse en la creación de funciones, donde es obligatoria. Indica el tipo de dato que retorna la función. En el cuerpo de la función tiene que existir una sentencia RETURN para indicar el valor que retorna la función.

  Una función solo puede retornar un valor. En el caso de necesitar una rutina que de-

vuelva más de un valor, se utilizará un procedimiento almacenado con más de un parámetro de salida (OUT).

- *cuerpo_rutina* puede ser una instrucción o un conjunto de instrucciones SQL en forma de bloque de programación empezando por BEGIN y terminando en END. El cuerpo de la rutina puede incluir sentencias de declaración de variables, de asignación de valores, de control de flujo, de manipulación de datos y la mayoría de las sentencias del lenguaje SQL, y además, puede hacer llamadas a otras rutinas almacenadas (procedimientos y funciones definidas por el usuario).

4.1.2. Uso de rutinas almacenadas

Para llamar a un procedimiento almacenado hay que utilizar la sentencia CALL, mientras que una función se utiliza como cualquier otra función de las que ya tiene definidas MySQL, añadiéndola a una expresión que puede utilizarse dentro de una sentencia SQL, como por ejemplo, SELECT, INSERT, UPDATE o DELETE. La función devuelve siempre un valor en el momento de evaluar la expresión.

4.1.3. Modificación de procedimientos almacenados y funciones

Los procedimientos y funciones existentes se pueden modificar empleando las sentencias ALTER y DROP.

Sentencias ALTER PROCEDURE y ALTER FUNCTION

Esta sentencia permite cambiar las características de una rutina almacenada (procedimiento o función). En la misma sentencia se puede cambiar más de una característica. Sin embargo, no se permite hacer cambios en los parámetros ni en el cuerpo de la rutina, para hacer estos cambios es necesario borrar la rutina almacenada y volver a crearla.

Sentencias DROP PROCEDURE y DROP FUNCTION

Estas sentencias permiten borrar procedimientos almacenados.