Insertar, Actualizar y Eliminar

En este documento vamos a describir las posibles alternativas que proporciona la API de Android a la hora de insertar, actualizar y eliminar registros de nuestra base de datos SQLite.

La API de SQLite de Android proporciona dos alternativas para realizar operaciones sobre la base de datos que no devuelven resultados (entre ellas la inserción, actualización y eliminación de registros, pero también la creación de tablas, de índices, etc.).

El primero de ellos, que ya comentamos brevemente en el artículo anterior, es el método `execSQL()` de la clase `SQLiteDatabase`. Este método permite ejecutar cualquier sentencia SQL sobre la base de datos, siempre que ésta no devuelva resultados. Para ello, simplemente aportaremos como parámetro de entrada de este método la cadena de texto correspondiente con la sentencia SQL. Por ejemplo:

```
//Insertar un registro
db.execSQL("INSERT INTO Usuarios (usuario, email) VALUES ('usu1','usu1@email.com') ");

//Eliminar un registro
db.execSQL("DELETE FROM Usuarios WHERE usuario='usu1' ");

//Actualizar un registro
db.execSQL("UPDATE Usuarios SET email='nuevo@email.com' WHERE usuario='usu1' ");
```

La segunda de las alternativas disponibles en la API de Android es utilizar los métodos `insert()`, `update()` y `delete()` proporcionados también con la clase `SQLiteDatabase`. Estos métodos permiten realizar las tareas de inserción, actualización y eliminación de registros de una forma algo más paramétrica que `execSQL()`, separando tablas, valores y condiciones en parámetros independientes de estos métodos.

Empecemos por el método `insert()` para insertar nuevos registros en la base de datos. Este método recibe tres parámetros, el primero de ellos será el nombre de la tabla, el tercero serán los valores del registro a insertar, y el segundo lo obviaremos por el momento ya que tan sólo se hace necesario en casos muy puntuales (por ejemplo para poder insertar registros completamente vacíos), en cualquier otro caso pasaremos con valor `null` este segundo parámetro.

Los valores a insertar los pasaremos como elementos de una colección de tipo `ContentValues`. Esta colección es de tipo *diccionario*, donde almacenaremos parejas de *clave-valor*, donde la *clave* será el nombre de cada campo y el *valor* será el dato correspondiente a insertar en dicho campo. Veamos un ejemplo:

```
//Creamos el registro a insertar como objeto ContentValues
ContentValues nuevoRegistro = new ContentValues();
nuevoRegistro.put("usuario", "usu10");
nuevoRegistro.put("email", "usu10@email.com");

//Insertamos el registro en la base de datos
db.insert("Usuarios", null, nuevoRegistro);
```

Los métodos `update()` y `delete()` se utilizarán de forma muy parecida a ésta, con la salvedad de que recibirán un parámetro adicional con la condición *WHERE* de la sentencia SQL.
Por ejemplo, para actualizar el email del usuario de nombre '*usu1*' haríamos lo siguiente:

```
//Establecemos los campos-valores a actualizar
ContentValues valores = new ContentValues();
valores.put("email","usu1_nuevo@email.com");

//Actualizamos el registro en la base de datos
db.update("Usuarios", valores, "usuario='usu1'");
```

Como podemos ver, como tercer parámetro del método `update()` pasamos directamente la condición del *UPDATE* tal como lo haríamos en la cláusula *WHERE* en una sentencia SQL normal.

El método `delete()` se utilizaría de forma análoga. Por ejemplo para eliminar el registro del usuario '*usu2*' haríamos lo siguiente:

```
//Eliminamos el registro del usuario 'usu2'
db.delete("Usuarios", "usuario='usu2'");
```

Como vemos, volvemos a pasar como primer parámetro el nombre de la tabla y en segundo lugar la condición *WHERE*. Por supuesto, si no necesitáramos ninguna condición, podríamos dejar como `null` en este parámetro.
Un último detalle sobre estos métodos. Tanto en el caso de `execSQL()` como en los casos de `update()` o `delete()` podemos utilizar argumentos dentro de las condiciones de la sentencia SQL.
Esto no es más que partes *variables* de la sentencia SQL que aportamos en un array de valores aparte, lo que nos evitará pasar por la situación típica en la que tenemos que construir una sentencia SQL concatenando cadenas de texto y variables para formar el comando SQL final.
Estos argumentos SQL se indicarán con el símbolo '?', y los valores de dichos argumentos deben pasarse en el array en el mismo orden que aparecen en la sentencia SQL. Así, por ejemplo, podemos escribir instrucciones como la siguiente:

```
//Eliminar un registro con execSQL(), utilizando argumentos
String[] args = new String[]{"usu1"};
db.execSQL("DELETE FROM Usuarios WHERE usuario=?", args);

//Actualizar dos registros con update(), utilizando argumentos
ContentValues valores = new ContentValues();
valores.put("email", "usu1_nuevo@email.com");

String[] args = new String[]{"usu1", "usu2"};
db.update("Usuarios", valores, "usuario=? OR usuario=?", args);
```

Esta forma de pasar a la sentencia SQL determinados datos variables puede ayudarnos además a escribir código más limpio y evitar posibles errores.