Consultar y Recuperar registros

En este documento vamos a describir la última de las tareas importantes de tratamiento de datos que nos queda por ver, la selección y recuperación de datos.
De forma análoga a lo que vimos para las sentencias de modificación de datos, vamos a tener dos opciones principales para recuperar registros de una base de datos SQLite en Android.

- La primera de ellas utilizando directamente un comando de selección SQL,
- Como segunda opción utilizando un método específico donde *parametrizaremos* la consulta a la base de datos.

Para la primera opción utilizaremos el método `rawQuery()` de la clase `SQLiteDatabase`. Este método recibe directamente como parámetro un comando SQL completo, donde indicamos los campos a recuperar y los criterios de selección. El resultado de la consulta lo obtendremos en forma de cursor, que posteriormente podremos recorrer para procesar los registros recuperados. Sirva la siguiente consulta a modo de ejemplo:

```
Cursor c = db.rawQuery(" SELECT usuario, email FROM Usuarios WHERE usuario='usu1' ");
```

Como en el caso de los métodos de modificación de datos, también podemos añadir a este método una lista de argumentos variables que hayamos indicado en el comando SQL con el símbolo '?', por ejemplo:

```
String[] args = new String[] {"usu1"};
Cursor c = db.rawQuery(" SELECT usuario, email FROM Usuarios WHERE usuario=? ", args);
```

Como segunda opción para recuperar datos podemos utilizar el método `query()` de la clase `SQLiteDatabase`. Este método recibe varios parámetros: el nombre de la tabla, un array con los nombre de campos a recuperar, la cláusula *WHERE*, un array con los argumentos variables incluidos en el *WHERE* (si los hay, `null` en caso contrario), la cláusula *GROUP BY* si existe, la cláusula *HAVING* si existe, y por último la cláusula *ORDER BY* si existe. Opcionalmente, se puede incluir un parámetro al final más indicando el número máximo de registros que queremos que nos devuelva la consulta.
Veamos el mismo ejemplo anterior utilizando el método `query()`:

```
String[] campos = new String[] {"usuario", "email"};
String[] args = new String[] {"usu1"};

Cursor c = db.query("Usuarios", campos, "usuario=?", args, null, null, null);
```

Como vemos, los resultados se devuelven nuevamente en un objeto Cursor que deberemos recorrer para procesar los datos obtenidos.
Para recorrer y manipular el cursor devuelto por cualquiera de los dos métodos mencionados tenemos a nuestra disposición varios métodos de la clase Cursor, entre los que destacamos dos de los dedicados a recorrer el cursor de forma secuencial y en orden natural:

- `moveToFirst()`: mueve el puntero del cursor al primer registro devuelto.
- `moveToNext()`: mueve el puntero del cursor al siguiente registro devuelto.

Los métodos `moveToFirst()` y `moveToNext()` devuelven `TRUE` en caso de haber realizado el movimiento correspondiente del puntero sin errores, es decir, siempre que exista un primer registro o un registro siguiente, respectivamente.
Una vez posicionados en cada registro podremos utilizar cualquiera de los métodos `getXXX(índice_columna)` existentes para cada tipo de dato para recuperar el dato de cada campo del registro actual del cursor. Así, si queremos recuperar por ejemplo la segunda columna del registro actual, y ésta contiene un campo alfanumérico, haremos la llamada `getString(1)`, en caso de contener un dato de tipo real llamaríamos a `getDouble(1)`, y de forma análoga para todos los tipos de datos existentes.
NOTA: Los índices comienzan por 0, por lo que la segunda columna tiene índice 1

Con todo esto en cuenta, veamos cómo podríamos recorrer el cursor devuelto por el ejemplo anterior:

```
String[] campos = new String[] {"usuario", "email"};
String[] args = new String[] {"usu1"};

Cursor c = db.query("Usuarios", campos, "usuario=?", args, null, null, null);

//Nos aseguramos de que existe al menos un registro
if (c.moveToFirst()) {
     //Recorremos el cursor hasta que no haya más registros
     do {
          String usuario = c.getString(0);
          String email = c.getString(1);
     } while(c.moveToNext());
}
```

Además de los métodos comentados de la clase `Cursor` existen muchos más que nos pueden ser útiles en muchas ocasiones.
Por ejemplo, `getCount()` te dirá el número total de registros devueltos en el cursor, `getColumnName(i)` devuelve el nombre de la columna con índice `i`, `moveToPosition(i)` mueve el puntero del cursor al registro con índice `i`, etc.
Se puede consultar la lista completa de métodos disponibles en la clase `Cursor` en la documentación oficial de Android.

Nota: Dejamos pendiente algunos temas algo más avanzados, como por ejemplo el uso de *transacciones*, pero con los métodos descritos podremos realizar un porcentaje bastante alto de todas las tareas necesarias relativas al tratamiento de datos estructurados en aplicaciones Android.