qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm

|  |
| --- |
| Acceso y gestión de una base de datos SQLite  28/01/2014  Patricia Laiz Fernández |

Contenido

[QUE ES SQLITE 3](#_Toc378589969)

[COMO CREAR Y CONECTAR CON SQLITE 3](#_Toc378589970)

[COMO INTRODUCIR UN REGISTRO EN LA BASE DE DATOS 6](#_Toc378589971)

[COMO MODIFICAR UN REGISTRO DE LA BASE DE DATOS 7](#_Toc378589972)

[COMO LEER LOS REGISTROS DE LA BASE DE DATOS 8](#_Toc378589973)

[COMO BORRAR UN REGISTRO DE LA BASE DE DATOS 8](#_Toc378589974)

[BIBLIOGRAFIA 9](#_Toc378589975)

Creación y Acceso a una base de datos SQLite.

## QUE ES SQLITE

SQLite es un motor de base de datos relacional de código abierto y muy potente, eso hace que actualmente sea muy usado por los desarrolladores. Sus principales características son que precisa de poca configuración, **no necesita** ningún **servidor** ya que directamente lee y escribe en archivos, ocupa muy poco tamaño (la longitud máxima de la consulta SQL es de 1 MB) en el almacenamiento y es multiplataforma.

Combina el motor y el interfaz de la base de datos en una única biblioteca, y almacena los datos en un único archivo de texto plano. Esto hace que cada usuario pueda crear tantas bases de datos como desee sin la necesidad de la intervención de un administrador de bases de datos que gestione los espacios de trabajo, usuarios y permisos de acceso. El hecho de almacenar toda la base de datos en un único archivo, facilita la **portabilidad** de los **datos**.

Android ofrece de serie soporte total para la creación y administración de base de datos SQLite a través del paquete **"android.database.sqlite"**. Solo tendremos que definir las sentencias SQL para crear y gestionar la base de datos.

## COMO CREAR Y CONECTAR CON SQLITE

En Android, la forma típica para crear, actualizar, y conectar con una base de datos SQLite será a través de una clase auxiliar llamada **SQLiteOpenHelper**, o para ser más exactos, de una clase propia que derive de ella y que debemos personalizar para adaptarnos a las necesidades concretas de nuestra aplicación.

La clase SQLiteOpenHelper tiene tan sólo un constructor, que normalmente no necesitaremos sobrescribir, y dos métodos abstractos, **onCreate(SQLiteDatabasedb)** y **onUpgrade(SQLiteDatabasedb, intoldVersion, intnewVersion),** que deberemos personalizar con el código necesario ya que onCreate(SQLiteDatabasedb**)** se encargara de crear las tablas si no existeny onUpgrade(SQLiteDatabasedb, intoldVersion, intnewVersion)**,** nos la actualiza si en un futuro necesitamos introducir más columnas, pasándole como parámetro la versión de la tabla antigua y la versión de la nueva tabla.

En nuestro ejemplo vamos a crear una Agenda Personal, donde podremos incluir los datos siguientes:

* El Id no lo introducimos nosotros ya que lo generará automáticamente.
* Nombre del contacto
* Teléfono
* E-mail
* Notas

Ejemplo de la tabla para la base de datos:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **NOMBRE** | **TELEFONO** | **E-MAIL** | **NOTAS** |
| 1 | Papá | 987987987 | papa@agenper.es | Ir a comer a casa |
| 2 | Mamá | 999888777 | mama@agenper.es | Llamar |
| 3 | Pepe | 987888777 | pepe@agenper.es |  |
| 4 | Daniel | 987777999 | daniel@agenper.es | Cita 5:15 |

Lo primero que haremos será nuestra clase Persona que será la encargada de mostrarnos los datos de los objetos que crearemos.

packagecom**.**patricialaiz**.**agendatelefonica**;**

**import**java**.**io**.**Serializable**;**

/\*\*

\*

\* **@author** Patricia Laiz

\*

\*será la encargada de mostrarnos los datos.

\*El id sera lo unico que no podremos modificar ya que los generamos automaticamente

\*/

publicclass Persona **implements**Serializable**{**

privateint id**;**

private String nombre**;**

privateinttelefono**;**

private String email**;**

private String notas**;**

/\*\*

\* constructores

\*/

public Persona**(){**

**super();**

**}**

public Persona**(**int id**,** String nombre**,**inttelefono**,** String email**,**

String notas**){**

**super();**

**this.**id **=** id**;**

**this.**nombre**=** nombre**;**

**this.**telefono**=**telefono**;**

**this.**email**=** email**;**

**this.**notas**=**notas**;**

**}**

Generando sus métodos de acceso y modificación de atributos (getters y setters).

Como segundo paso, crearemos nuestra clase derivada de **SQLiteOpenHelper** que llamaremos **BaseDatosPersonas**, donde sobrescribiremos los métodos onCreate() y onUpgrade() con el código de lo que queremos que hagan.

Lo primero que tenemos que hacer es crear las **variables**:

**VERSION\_BASEDATOS**: Es la versión de nuestra base de datos que será Int.

**NOMBRE\_BASEDATOS**: Es el nombre que le pondremos a nuestra base de datos que será String.

**TABLA\_PERSONAS:** Es el String que contiene la sentencia "CREATE TABLE contactos (\_id INT PRIMARY KEY, nombre TEXT, teléfono INT, email TEXT, notas TEXT)" para la creación de una tabla llamada “Personas”. En la sentencia hay que especificar el tipo de datos que van a soportar las columnas.

packagecom**.**patricialaiz**.**agendatelefonica**;**

**import**android**.**content**.**Context**;**

**import**android**.**database**.**sqlite**.**SQLiteDatabase**;**

**import**android**.**database**.**sqlite**.**SQLiteDatabase**.**CursorFactory**;**

**import**android**.**database**.**sqlite**.**SQLiteOpenHelper**;**

publicclassBaseDatosPersonas**extends**SQLiteOpenHelper**{**

// version de nuestra vase de datos

privatestaticfinalint VERSION\_BASEDATOS **=**1**;**

// Nombre de nuestro archivo de base de datos

privatestaticfinalString NOMBRE\_BASEDATOS **=**"basedatospersonas.db"**;**

//Sentencia SQL para crear la tabla de Personas

privatestaticfinal String TABLA\_PERSONAS **=**"CREATE TABLE Personas (id INTEGER PRIMARY KEY AUTOINCREMENT, nombre TEXT,"**+**

" telefono INTEGER, email TEXT, notas TEXT)"**;**

**}**

Seguimos con el **constructor** de la clase que nos pide como parámetros un **context**, el nombre del archivo de la base de datos, un **CursorFactory** que no lo vamos a necesitar y por lo tanto lo ponemos a null y la **versión** de nuestra base de datos.

Para gestionar la base de datos y la clase que estamos creando, vamos a usar métodos de la clase **"SQLiteDatabase".**

// CONSTRUCTOR de la clase

public BaseDatosPersonas**(**Context context**,** String name**,**CursorFactory factory**,**int version**){**

**super(**context**,** NOMBRE\_BASEDATOS**,null,** VERSION\_BASEDATOS**);**

**}**

Y para terminar la clase sobrescribimos los métodos **onCreate** y **onUpgrade** usando el método **execSQL()** que nos pide como parámetro una única sentencia SQL que **no sea** un **SELECT** o cualquier otra sentencia que devuelva datos. Entonces en el onCreate simplemente creamos la tabla si no existe y en el onUpgrade borramos la tabla si existe y creamos una nueva.

@Override

publicvoidonCreate**(**SQLiteDatabasedb**){**

//Se ejecuta la sentencia SQL de creación de la tabla

db**.**execSQL**(**TABLA\_PERSONAS**);**

**}**

@Override

publicvoidonUpgrade**(**SQLiteDatabasedb**,**intoldVersion**,**intnewVersion**){**

//Borramos la tabla si existe y creamos una nueva.

db**.**execSQL**(**"DROP TABLE IF EXISTS "**+** TABLA\_PERSONAS**);**

onCreate**(**db**);**

**}**

**}**

Ahora que ya tenemos nuestras clases principales, tenemos que crear una Activityprincipal que se encargue de darnos las opciones de ver los contactos creados, o nos de paso a la pantalla para ir introduciendo los contactos.

### COMO INTRODUCIR UN REGISTRO EN LA BASE DE DATOS

En nuestra Activity de crear contactos, creamos el método **insertarContacto()** en el que capturaremos los datos introducidos por el usuario y generaremos un registro nuevo en la base de datos.

Antes de nada tenemos que crear la base y utilizar el método **getWritableDatabase()** para poder escribir en ella.

// Creamos la base de datos de personas y la abrimos para leer y escribir

databasePersonas**=new**BaseDatosPersonas**(this,**"personas.db"**,null,**1**);**

SQLitedb**=**databasePersonas**.**getWritableDatabase**();**

Lo Metemos todo en un if por si acaso la base de datos no existe y ya dentro del if creamos una instancia de la clase **"ContentValues"** que como su nombre indica es un contenedor de un conjunto de datos. Usamos el método **put(key, value)**que nos pide como primer parámetro **"key"** el nombre donde establecer el valor almacenado y como segundo parámetro el valor que queremos almacenar. Una vez almacenamos los datos insertamos una fila en la tabla usando el método **insert(table, nullColumnHack, values)**que nos pide el nombre de la tabla **"table"**, un segundo parámetro en caso de que necesitemos insertar valores nulos en la tabla **"nullColumnHack"** en este caso lo dejaremos a null ya que no lo vamos a usar y como tercer parámetro nos pide un **"ContentValues"**, que será nuestro contenedor.

publicvoidinsertarContacto**(**View v**){**

**if(**SQLitedb**!=null){**

// utilizamos un contentValues como contenedor de los datos a meter en la base de datos

ContentValues valores **=new**ContentValues**();**

//metemos los datos en nuestro contenedor

valores**.**put**(**"nombre"**,**nomTemp**);**

valores**.**put**(**"telefono"**,**telTemp**);**

valores**.**put**(**"email"**,**emailTemp**);**

valores**.**put**(**"notas"**,**nomTemp**);**

//insertamos los datos en la base

SQLitedb**.**insert**(**"Personas"**,null,** valores**)**

**}**

### COMO MODIFICAR UN REGISTRO DE LA BASE DE DATOS

Prácticamente es igual que el anterior método pero con la excepción de que aquí estamos usando el método **"update(table, values, whereClause, whereArgs)"** para actualizar/modificar registros de nuestra tabla. Este método nos pide el nombre de la tabla **"table"**, los valores a modificar/actualizar **"values"** (ContentValues), una condición WHERE **"whereClause"** que es la condición donde queremos que actualice los datos (en este caso cogemos como referencia el id de nuestro contacto) y como último parámetro **"whereArgs"** podemos pasarle los valores nuevos a insertar, en este caso no lo vamos a necesitar por lo tanto lo ponemos a null.

Lo primero es abrir la base de datos en modo lectura y escritura:

// Creamos la base de datos de personas y la abrimos para leer y escribir

databasePersonas**=new**BaseDatosPersonas**(this,**"personas.db"**,null,**1**);**

SQLitedb**=**databasePersonas**.**getWritableDatabase**();**

Después creamos nuestro método de modificar datos

public void guardarCambiosContacto**(**View v**){**

String nomTemp**=**modNom**.**getText**().**toString**();**

inttelTemp**=**0**;**

telTemp**=**Integer**.**parseInt**(**modTel**.**getText**().**toString**());**

String emailTemp**=**modEmail**.**getText**().**toString**();**

String notasTemp**=**modNotas**.**getText**().**toString**();**

**if(**SQLitedb**!=null){**

// utilizamos un contentValues como contenedor de los datos a meter

// en la base de datos

ContentValues valores **=new**ContentValues**();**

// metemos los datos en nuestro contenedor

valores**.**put**(**"nombre"**,**nomTemp**);**

valores**.**put**(**"telefono"**,**telTemp**);**

valores**.**put**(**"email"**,**emailTemp**);**

valores**.**put**(**"notas"**,**notasTemp**);**

Stringidmodi**=**"id == "**+**idAmodificar**;**

// Sentencia para modificar los valores

SQLitedb**.**update**(**"Personas"**,** valores**,**idmodi**,null);**

Toast**.**makeText**(this,**"Se guardaron los cambios"**,**Toast**.**LENGTH\_SHORT**)**

**.**show**();**

volver**();**

**}**

**}**

Si todo esta correcto se nos creará nuestra base de datos en:

/data/data/com.patricialaiz.agendatelefonica/databases/basedatospersona

### COMO LEER LOS REGISTROS DE LA BASE DE DATOS

Para acceder a los datos de la base de datos, utilizaremos el método **rawquery(sql, selectionArgs)**que directamente recibe una declaración SQL como primer parámetro y un segundo parámetro opcional.

// realizamos la consulta a la base de datos y la guardamos en un cursor

Cursor c **=**SQLitedb**.**rawQuery**(**"SELECT \* FROM Personas"**,null);**

A parte creamos una variable **"listaContactos"**que será un array, donde almacenaremos todos los registros de la tabla en objetos Persona. En el bucle do-while usamos el método **moveToNext()**como parámetro que se encargara de pasar al siguiente registro de la tabla y por lo tanto recorrer todos los registros de la tabla.

**do{**

// Sacamos la informacion de cada columna de un elemento

// La columna de 0 es el id

id**=**c**.**getInt**(**0**);**

// La columna de 1 es el nombre

nombre**=**c**.**getString**(**1**);**

// La columna 2 es el telefono

telefono**=**Integer**.**parseInt**(**c**.**getString**(**2**).**toString**());**

// La columna 3 es el email

email**=**c**.**getString**(**3**);**

// La columna de 4 son las notas

notas **=**c**.**getString**(**4**);**

// Llenamos el array con los contactos

listaContatcos**[**i**]=new** Persona**(**id**,** nombre**,**telefono**,** email**,** notas**);**

i**++;**

// movemos el cursor al siguiente registro

**}while(**c**.**moveToNext**());**

### COMO BORRAR UN REGISTRO DE LA BASE DE DATOS

Para borrar registros usaremos el método **"delete(table, whereClause, whereArgs)"** que nos pide el nombre de la tabla **"table"**, el registro a borrar **"whereClause"** que tomaremos como referencia su id y como último parámetro **"whereArgs"** los valores a borrar. Como en los casos anteriores tenemos que abrir la base de datos en modo lectura y escritura.

publicvoidborrar**(**View v**){**

StringidAeliminar**=**"id == "**+**contacto**.**getId**();**

//Sentencia para eliminar el objeto nombre de la tabla, registro a borrar (usamos el id) y valores a borrar

SQLitedb**.**delete**(**"Personas"**,**idAeliminar**,null);**

// Intent intent = new Intent(this, ListadoPersonas.class);

Toast**.**makeText**(this,**"Contactoeliminado"**,**Toast**.**LENGTH\_SHORT**).**show**();**

finish**();**

**}**

En este caso utilizamos el id para indicar el registro de la tabla que queremos borrar.

## BIBLIOGRAFIA

* <http://elbauldeandroid.blogspot.com.es/2013/02/base-de-datos-sqlite.html>
  + Creación y gestión de una base de datos (ejemplos)
* <http://www.sgoliver.net/blog/?p=1611>
  + Crear base de datos.
* <http://www.sgoliver.net/blog/?p=1632>
  + Insertar/Actualizar/Elimina
* <http://www.sgoliver.net/blog/?p=1632>
  + Consulta/Recuperar registros