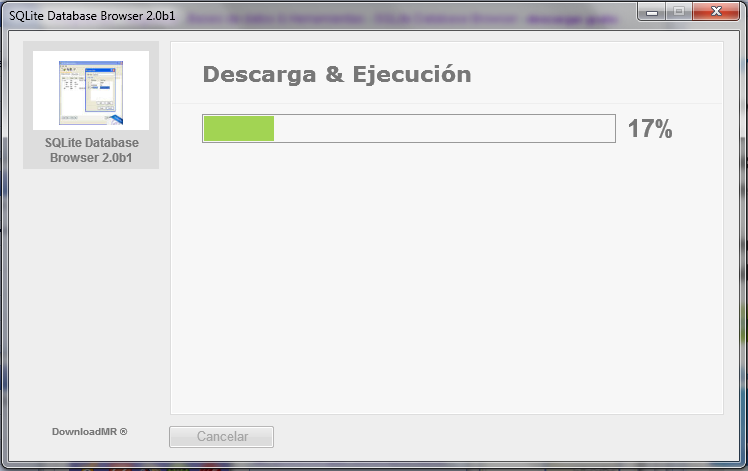
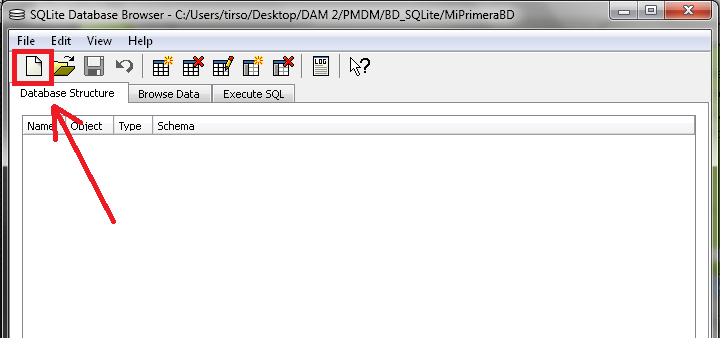
# **CREAR Y COPIAR UNA BASE DE DATOS EXTERNA (SQLite)**

**I. Pasos para crear una base de datos externa**

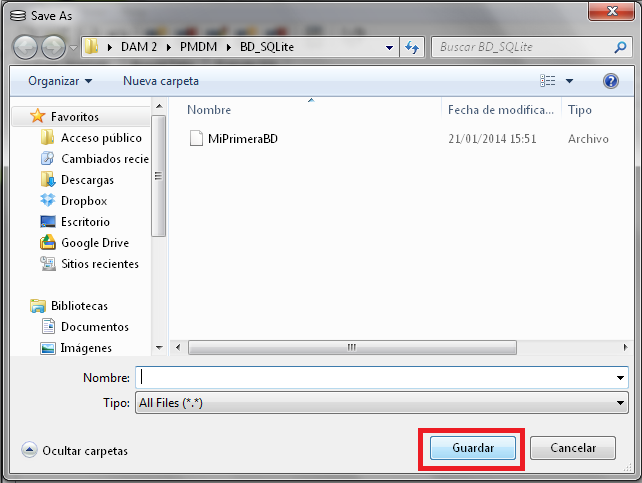
Lo primero que necesitamos es un software de código abierto **SQLite Database Browser** (disponible para Windows y Linux).

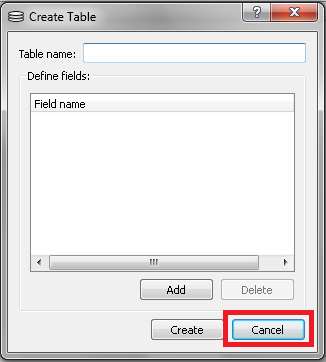


Terminada la descarga, ejecutamos el fichero SQLite Database Browser.



Pulsa sobre nuevo, te preguntara donde quieres guardar la base de datos, selecciona tu ruta (una que te sea sencilla acceder, porque luego tendremos que copiarla a nuestra aplicación de Android en Eclipse) y pulsa sobre **Guardar**.



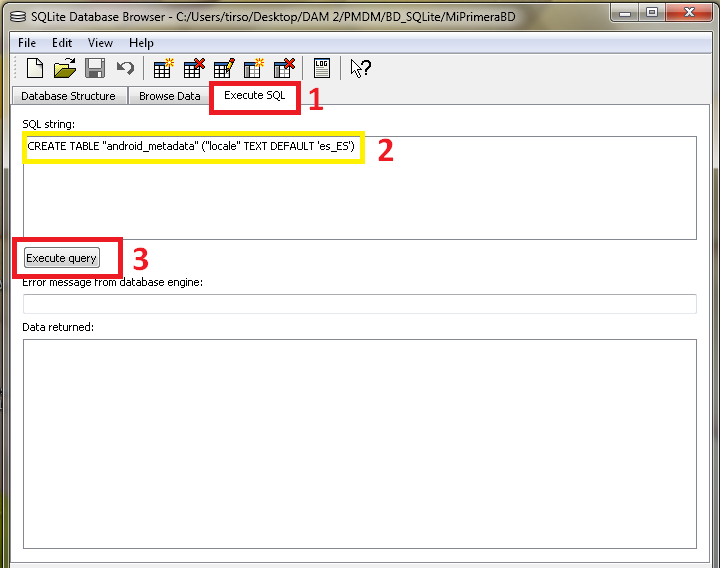
 Saldrá un cuadro de diálogo preguntándote si quieres crear una tabla, pulsa **Cancelar**.

Nuestra base de datos SQLite para Android tiene que cumplir unos requisitos para que nuestra aplicación la pueda utilizar.

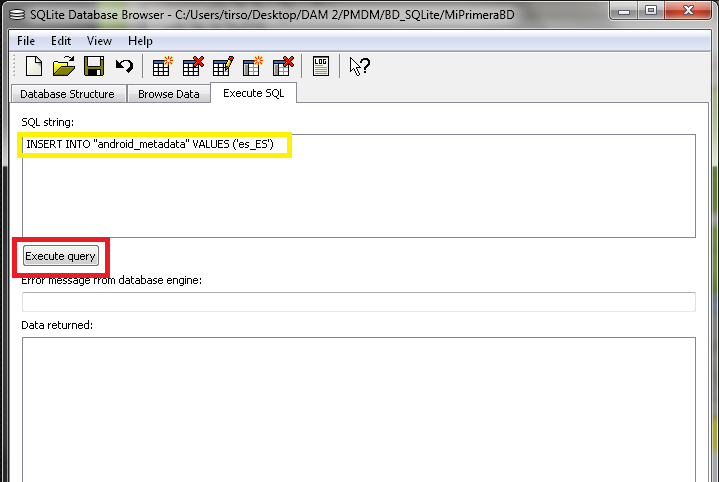
La base de datos debe tener una tabla llamada android\_metadata que tenga una sola columna "locale" de tipo TEXT. Dicha tabla debe tener al menos un valor, por ejemplo es\_ES.

Pulsa sobre la pestaña "**Execute SQL**", escribe la siguiente instrucción SQL: ***CREATE TABLE "android\_metadata" ("locale" TEXT DEFAULT 'es\_ES')*** y después pulsa sobre el botón "**Execute query**".

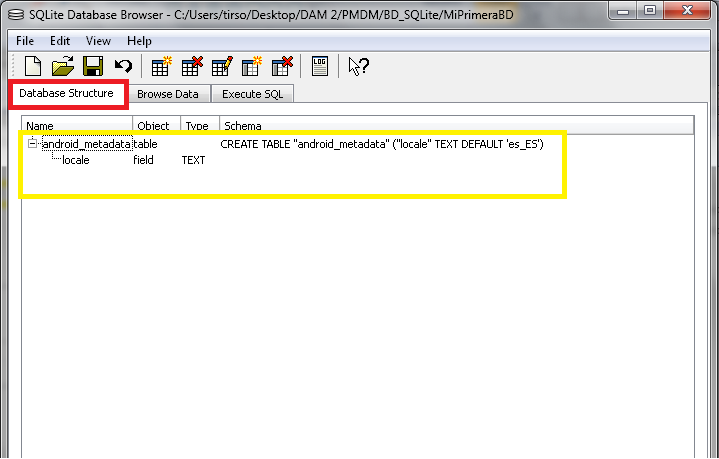
Si todo ha salido bien, verás un mensaje de “No error”.



Ahora escribe ***INSERT INTO "android\_metadata" VALUES ('es\_ES')*** para crear un registro con el dato ‘es\_ES’ en la tabla recién creada.



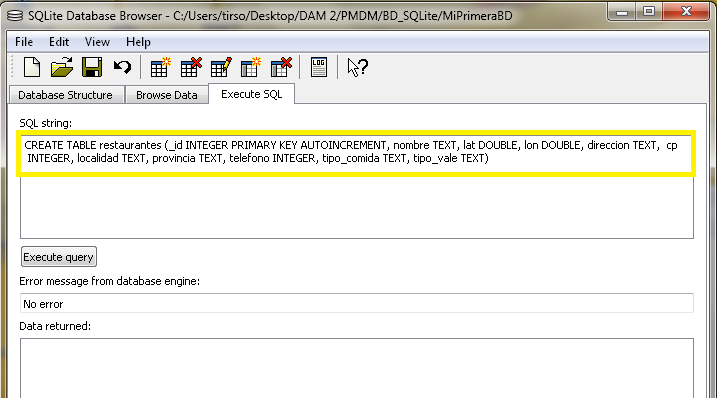
Vuelve a la pestaña de “**Database Structure**” para comprobar que este la tabla, fíjate que aquí aparecen las tablas, y si le damos al + aparecen los campos.



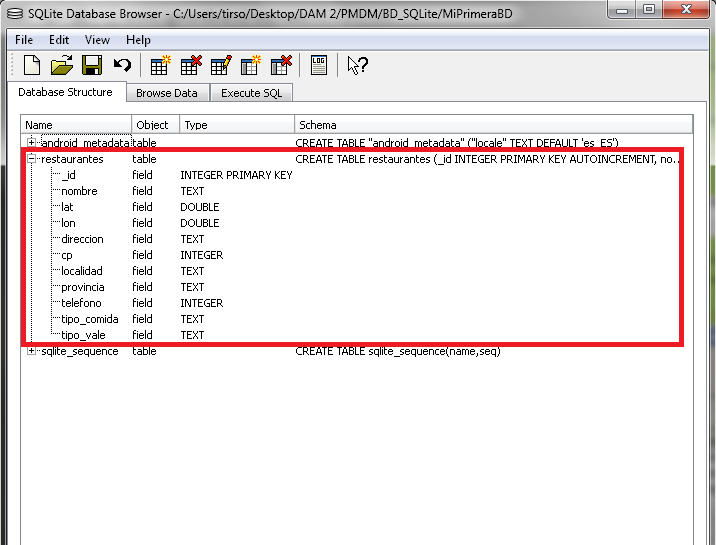
Ahora debes de crear otra tabla con el nombre que quieras para introducir los datos que necesites, para esto puedes usar otra sentencia de SQL o utilizar el asistente que tiene la aplicación (los iconos superiores de la aplicación).

Todas las tablas que crees en la BD (a excepción de android\_metadata) deben tener la misma clave primaria:

* Nombre del campo: **\_id**
* Tipo de campo: **INTEGER PRIMARY KEY AUTOINCREMENT**

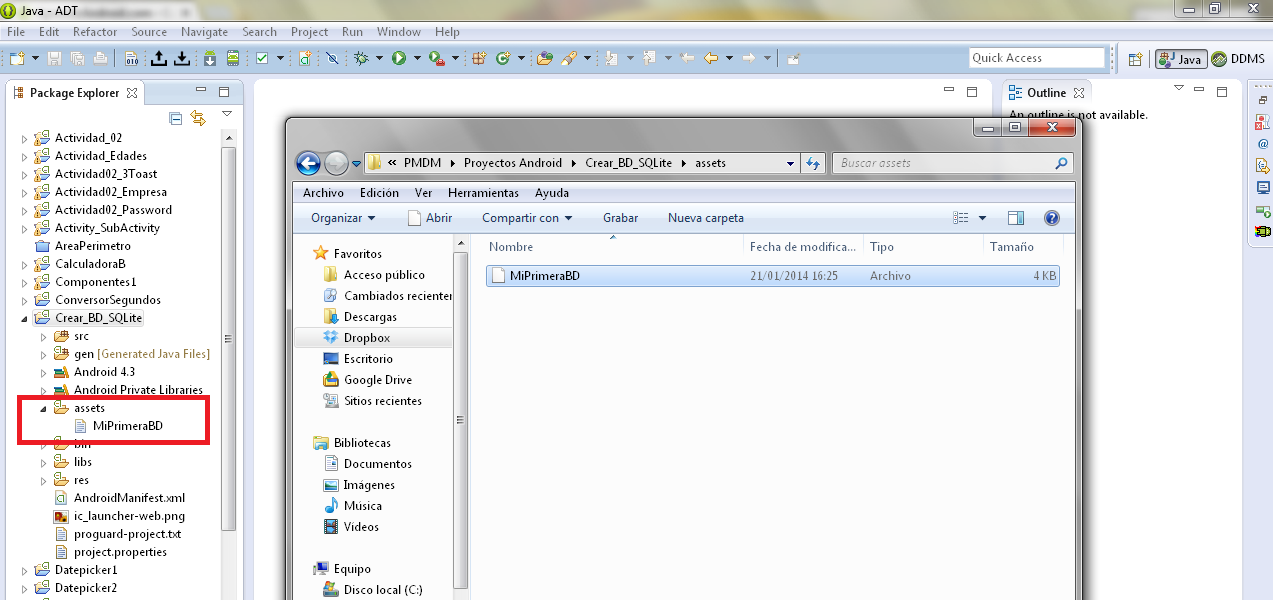


Es posible que la propia aplicación te añada una tercera tabla llamada “sqlite\_sequence”, en ella se guardan las transacciones que se van haciendo en la BD. **NO LA BORRES**.



**II. Copiar la BD externa a nuestra aplicación con SQLiteOpenHelper**

Lo primero es copiar el fichero resultante de crear nuestra BD a nuestro proyecto de Eclipse. Coge el fichero y arrástralo desde el explorador de Windows hasta la carpeta “**assets**” de tu proyecto. Si no existe esta carpeta la tendrás que crear primero. Has de asegurarte que tu BD no ocupa más de 1Mb, ya que Android tiene una limitación para copiar ficheros de más de 1Mb desde el directorio “**assets**” o “**raw**”. Si la BD es de más peso, hay que “partir” el fichero resultante en fragmentos de 1Mb para luego recomponerla en nuestra aplicación.



Para copiar una BD SQLite se necesita una clase auxiliar llamada **SQLiteOpenHelper**, o para ser más exactos, una clase propia que derive de ella y que debemos personalizar para adaptarnos a las necesidades concretas de nuestra aplicación.

**public class nombreDeTuClase extends SQLiteOpenHelper**

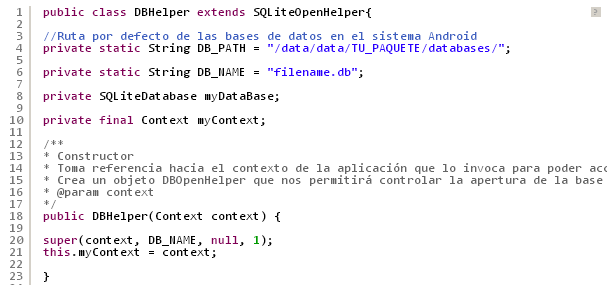
Lo primero que hacemos es crear dos variables:

- **DB\_PATH**: Es la ruta que tienen las bases de datos. Todas las bases de datos de cada aplicación se guardan en esa ruta.

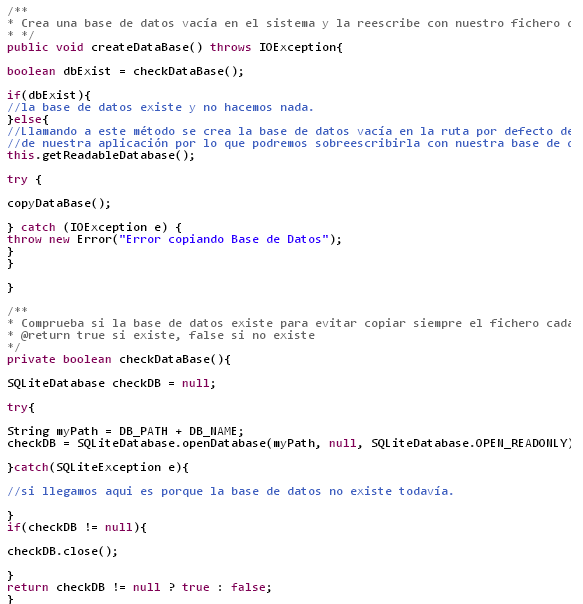
**private static final String DB\_PATH = "/data/data/nombreDelPaquete/databases/";**

- **DB\_NAME**: Es el nombre de la BD de tu aplicación, es decir, el nombre que utilizarás en tu aplicación para acceder a la BD.

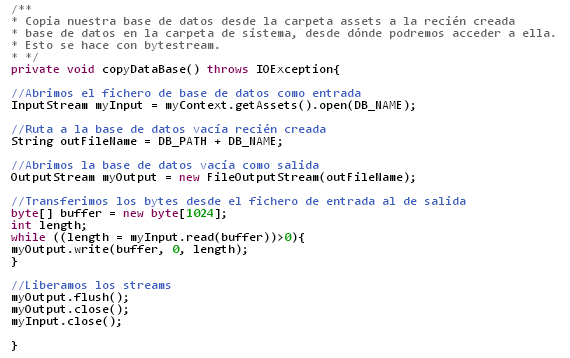
**private static final String DB\_NAME = “nombreBD.db”;**

****

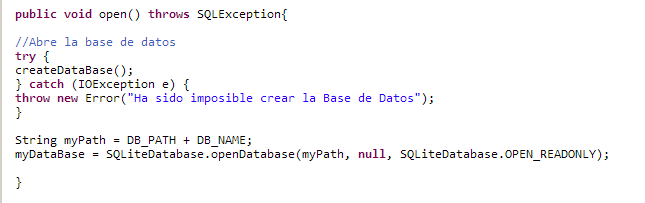
Para copiar el contenido de la BD externa, primero hay que crear una BD vacía en el DB\_PATH que hemos proporcionado al principio. Mediante una variable de tipo boolean se comprueba si ya existe esa BD. En caso de que no exista, se llama al método **getReadableDatabase()** que crea la BD vacía, para sobrescribirla con la BD externa mediante el método **copyDataBase()**, definido más abajo.



Método para sobrescribir la BD vacía que acabamos de crear:

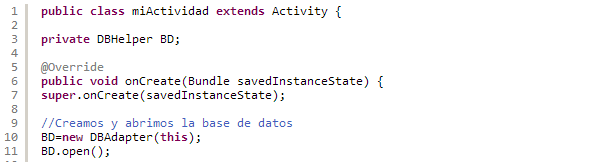


Método para abrir la BD:



**III. Usar la BD desde una Activity**

Una vez tenemos creada la clase DBHelper,  basta con crear un objeto DBHelper y llamar al método **open()** desde la Activity. Este código deberás ponerlo solo la primera vez que intentes acceder a la base de datos por primera vez, yo te aconsejo que lo pongas seguido al **OnCreate()** para que te asegures de que inicializas la base de datos antes de usarla más adelante cuando te interese.



**DIVIDIR UNA BD GRANDE EN TROZOS DE 1Mb**

El proceso de partir o dividir una base de datos es “complicado", tendremos que usar un comando "Split" que solo está en sistemas operativos Linux.

Requerimientos:

* **VMware Player.** Es el software que usaremos para la virtualización. ([**http://www.vmware.com/products/player/overview.html**](http://www.vmware.com/products/player/overview.html))
* **Ubuntu Linux.** Un ISO que usaremos para la instalación de Linux.

([**http://www.ubuntu.com/download/desktop**](http://www.ubuntu.com/download/desktop))

* Un **PenDrive USB**. Para intercambiar ficheros entre nuestro ordenador físico, que seguramente correrá bajo Windows y nuestra máquina virtual que corre bajo Linux.

Una vez tengas copiada la Base de datos en la memoria USB, ve a la barra de menú el VMware, pulsa sobre "Removable Devices" y busca en la lista tu dispositivo USB, después selecciona la opción "Connect (Disconect from host)".

Como ves, se te agregara la memoria USB en tu máquina virtual (Linux), y ya podremos dividir la base de datos, si instalas las WMware Tools (te aparecerá un aviso debajo de la pantalla), podrás arrastrar archivos entre la maquina física y la máquina virtual, y no necesitaras PenDrive, esto ya como tú quieras, el resultado.

Desde el PenDrive arrastro y copio la BD en la carpeta "Home", de esta forma en el Terminal no tendremos que movernos entre carpetas, por defecto, cuando entramos en el Terminal estaremos en esa carpeta.

En el terminal, ejecutas la sentencia: ***split -b 1m “nombreBD” “nombreNuevo\_”***

El resultado serán varios archivos de 1Mb con un sufijo: nombreNuevo\_aa, nombreNuevo\_ab, nombreNuevo\_ac…

Para recomponer el archivo de BD en la aplicación de Android en la clase extendida de SQLiteOpenHelper, es necesario copiar los ficheros resultantes del comando “Split” a la carpeta “raw”, que está dentro de la carpeta “res”.

El código para este caso, es casi similar al anterior, solamente cambia el contenido del método **copyDataBase()**.



**Bibliografía**

<http://blog.netrunners.es/usar-nuestra-propia-base-de-datos-sqlite-en-android/>

<http://www.aprendeandroid.com/l5/sql4.htm>

<http://elbauldeandroid.blogspot.com.es/2013/02/base-de-datos-sqlite.html>

<http://www.sgoliver.net/blog/?p=1611>

<http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html>

<http://www.aprendeandroid.com/l5/sql5.htm>