A RESEARCH PAPERS FOR COPYING LARGER DATABASE FILE IN ANDROID APPLICATION

Ms. Sneha PravinChandra Mandavia

Abstract- this paper on Android deals with copy larger data base from assets/raw folder to Android application.

We can copy a database file or any other file from assets/raw folder to our android application by reading byte from file and writing byte to the application, it will work if our database size in assets/raw folder is less than 1MB.

But if database file more than 1Mb in size then you will certainly get the following error when you are trying to copy the database. D/asset (909): Data exceeds UNCOMPRESS_DATA_MAX (1424000 vs 1048576)

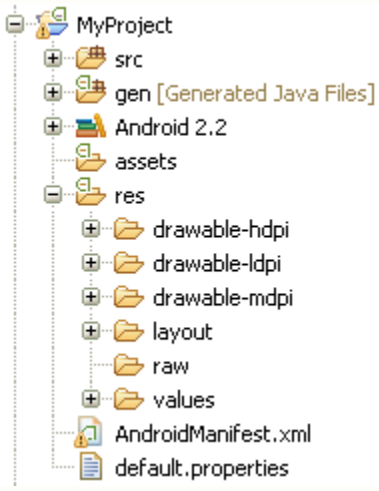
For that we have to split binary database file into a maximum of 1048576 bytes that is Mb and then we have to read those chunks one by one.

So by this way we can get final full size database in our application. This solution is very fast and work perfectly.

1. INTRODUCTION

Android is the world's most popular mobile platform. It is an open source operating system. [1]

Android projects are the projects that eventually get built into an .apk file that you install onto a device. They contain things such as application source code and resource files.[6]



As per Android documentation assets folder can use it to store raw asset files. Files those are save here are compiled into an .apk file as-is, and the original filename is preserved. We can navigate this directory in the same way as a typical file system using URIs and read files as a stream of bytes using the the AssetManager. [6]

As per android documentation raw folder saves assets file similar to assets folder. But only differs in the way that we access them. These files must be referenced from the application using a resource identifier in the R class. [6]

We can read byte from assets folder or raw folder using **InputStream** class which is a reads the data from a source in a byte-wise manner.

Fetching file from assets folder: InputStream obj = getApplicationContext().getAssets().open("myfile");

open() method open an asset using ACCESS_STREAMING mode. This provides access to files which are placed in to the "assets" directory.

Fetching file from raw folder: InputStream obj = getApplicationContext().getResources().openRawResource(R.raw.myfile);

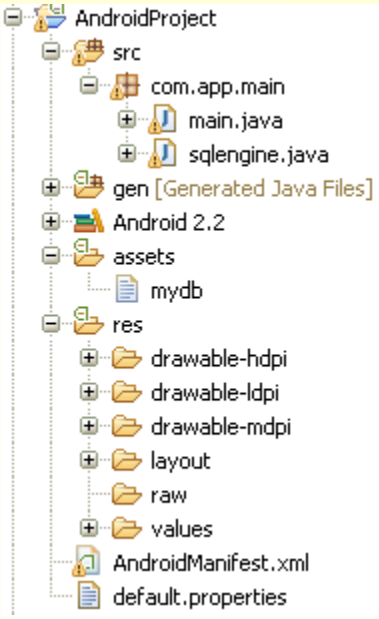
openRawResource() method open a data stream for reading a raw resource. This can only be used with resources whose value is the name of an asset files -- that is, it can be used to open drawable, sound, and raw resources. it will fail on string and color resources.

2. COPY DATABASE FROM assets/raw Folder

Step 1: Create a one android project from Eclipse.

Package name: suppose com.app.main

Step 2: Copy database file in assets folder. Database file name is mydb and size is 3MB. [2]



Step 3: Create a new class called sqlengine and derive SQLiteOpenHelper class. A helper class to manage database creation and version management.

```
package com.app.main;

import android.content.Context;

public class sqlengine extends SQLiteOpenHelper{

    private SQLiteDatabase myDataBase;
    private final Context myContext;
    sqlengine sql;
    /**
     * Constructor Takes and keeps a reference of the passed context in order to
     * access to the application assets and resources.
     */
    public sqlengine(Context context) {
        super(context, "mydb", null, 1);
        this.myContext = context;
    }

    @Override
    public synchronized void close() {
        if (myDataBase != null)
            myDataBase.close();
        super.close();
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // TODO Auto-generated method stub
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // TODO Auto-generated method stub
    }
}
```

Step 4: Create a method in class called createDataBase() which will check whether the particular database is exits or not, if it is not exits than it will copy the database from assets/raw folder to our application.

```
public void createDataBase() throws IOException {
    boolean dbExist = checkDataBase();
    if (dbExist) {
        // do nothing - database already exist
    } else {
        // By calling this method and empty database will be created into
        // the default system path
        this.getReadableDatabase();
        copyDataBase();
    }
}
```

So to check existence we have to make checkDataBase() method and to copy database we have to create copyDataBase() method in sqlengine class.

```
private boolean checkDataBase() {
    SQLiteDatabase checkDB = null;
    try {
        String myPath = "/data/data/com.app.main/databases/" + "mydb";
        checkDB = SQLiteDatabase.openDatabase(myPath, null,
                SQLiteDatabase.OPEN_READONLY);
    } catch (SQLiteException e) {
        // database does't exist yet.
    }
    if (checkDB != null) {
        checkDB.close();
    }
    return checkDB != null ? true : false;
}
```

```
private void copyDataBase() {
    // Open your local db as the input stream
    try {
        OutputStream databaseOutputStream = new FileOutputStream(
                "/data/data/com.app.main/databases/" + "mydb");
        InputStream databaseInputStream;
        byte[] buffer = new byte[1024];
        int length;

        databaseInputStream = myContext.getResources().getAssets().open("mydb");
        while ((length = databaseInputStream.read(buffer)) > 0) {
            databaseOutputStream.write(buffer);
        }
        databaseOutputStream.flush();
        databaseOutputStream.close();

    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

So now sqlengine class is ready having three extra method - createDataBase() ,checkDataBase(), copyDataBase().

Step 5: call this class from our activity.

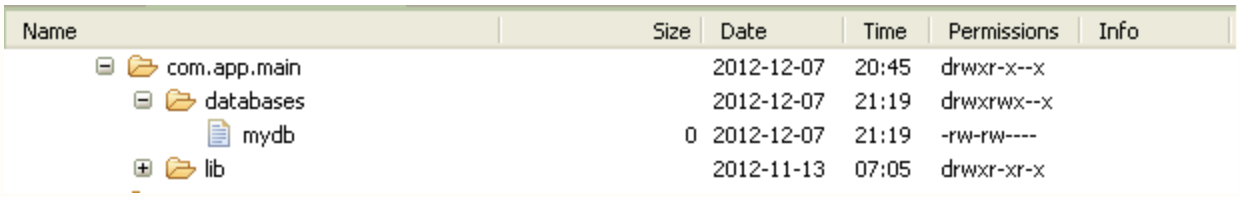
```
public class main extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        sqlengine objSqlEngine = new sqlengine(getApplicationContext());
        try {
            objSqlEngine.createDataBase();
        } catch (IOException e) {
            e.printStackTrace();
        }
        objSqlEngine.close();
    }
}
```

So on call createDataBase() method from activity, it will call checkDataBase() method to check whether mydb database is exits or not and if the method returns false then it will call copyDataBase() method to copy file from assets folder and call read() method to read byte, but it will certainly throws the following error when it tries to copy the database.

```
I    5886  Database         sqlite returned: error code = 14, msg = cannot open file at source line 25467
E    5886  Database         sqlite3_open_v2("/data/data/com.app.main/databases/mydb", &handle, 1, NULL) failed
D    5886  asset            Data exceeds UNCOMPRESS_DATA_MAX (7989248 vs 1048576)
W    5886  System.err       java.io.IOException
W    5886  System.err           at android.content.res.AssetManager.readAsset(Native Method)
W    5886  System.err           at android.content.res.AssetManager.access$700(AssetManager.java:36)
W    5886  System.err           at android.content.res.AssetManager$AssetInputStream.read(AssetManager.java:571)
W    5886  System.err           at com.app.main.sqlengine.copyDataBase(sqlengine.java:90)
W    5886  System.err           at com.app.main.sqlengine.createDataBase(sqlengine.java:60)
W    5886  System.err           at com.app.main.main.onCreate(main.java:17)
W    5886  System.err           at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1047)
W    5886  System.err           at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2627)
W    5886  System.err           at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2679)
W    5886  System.err           at android.app.ActivityThread.access$2300(ActivityThread.java:125)
W    5886  System.err           at android.app.ActivityThread$H.handleMessage(ActivityThread.java:2033)
W    5886  System.err           at android.os.Handler.dispatchMessage(Handler.java:99)
W    5886  System.err           at android.os.Looper.loop(Looper.java:123)
W    5886  System.err           at android.app.ActivityThread.main(ActivityThread.java:4627)
W    5886  System.err           at java.lang.reflect.Method.invokeNative(Native Method)
W    5886  System.err           at java.lang.reflect.Method.invoke(Method.java:521)
W    5886  System.err           at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:868)
W    5886  System.err           at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:626)
W    5886  System.err           at dalvik.system.NativeStart.main(Native Method)
I    61    ActivityManager  Displayed activity com.app.main/.main: 3810 ms (total 3810 ms)
```

Due to the fact that there is a file size limit (upto 1 MB) on resources in the raw or assets folders. Finally it indicates that you cannot copy the file which is >1 MB into your application at runtime.

And on seeing DDMS - > File Explorer->data->data->com.app.main->database we can see mydbfile has been created but its size is zero.



So we can copy a file larger than 1MB from assets folder to android application.

Step 4: Remove the mydb file from assets folder and copy that file in raw/ folder. Do minor change in copyDataBase() function i.e. replace InputStream obj = getApplicationContext().getAssets().open("mydb"); with InputStream obj = getApplicationContext().getResources().openRawResource(R.raw.mydb);

Now during runtime on call this method it will similarly throws an runtime error and on seeing DDMS - > File Explorer->data->data->com.app.main->database we can see mydbfile has been created but its size is zero.

So we cannot copy the file from assets or raw folder whose size is larger than 1MB.

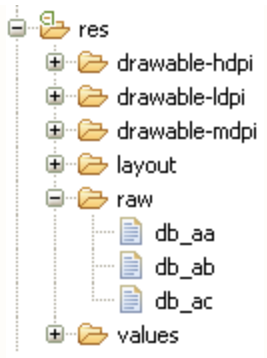
So the solution is to split the file in to 1MB's file chunks and read those chunks one by one.

3. COPYING LARGER THAN ONE 1MB FILE FROM assets/raw Folder

Step 1. Split the file: used the Linux split command to split the binary database file into a maximum of 1048576 bytes. [3]

- The command is: **split mydb -b 1048576 db_**
  - Here mydb is the database file which we want to split in to 1MB chunks.
  - 1048576 is the size of the single chunk. 1048576 byte = 1MB
  - And db_ is the output file prefix.
  - So in this example our mydb file's size is 3MB, so command will split the file in to three chunks, each of them has 1MB file size and their name would be db_aa, db_ab, db_ac respectively.

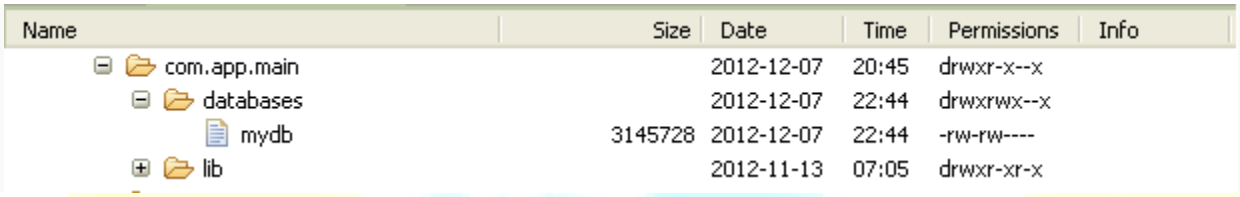
Step 2. Copy these files into raw resource folder.[3]



Step 3. And do minor changes in copyDataBase() method in sqlengine class, as now we have the chunks of file we have to read one by one each chunks and write them into application database.

```
private void copyDataBase() {
    // Open your local db as the input stream
    try {
        OutputStream databaseOutputStream = new FileOutputStream(
                "/data/data/com.app.main/databases/" + "mydb");
        InputStream databaseInputStream;

        byte[] buffer = new byte[1024];
        int length;

        databaseInputStream = myContext.getResources().openRawResource(R.raw.db_aa);
        while ((length = databaseInputStream.read(buffer)) > 0) {
            databaseOutputStream.write(buffer);
        }
        databaseInputStream.close();

        databaseInputStream = myContext.getResources().openRawResource(R.raw.db_ab);
        while ((length = databaseInputStream.read(buffer)) > 0) {
            databaseOutputStream.write(buffer);
        }
        databaseInputStream.close();

        databaseInputStream = myContext.getResources().openRawResource(R.raw.db_ac);
        while ((length = databaseInputStream.read(buffer)) > 0) {
            databaseOutputStream.write(buffer);
        }
        databaseInputStream.close();
        databaseOutputStream.flush();
        databaseOutputStream.close();

    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

So on call copyDataBase() method by createDataBase() method it will read files in to raw folder and copy to application database that is in /data/data/com.app.main/databases/ folder. And we can get fully copied database.



4. CONCLUSION

The solution works perfectly and it's very fast. Final database size on the emulator is 3,072 KB. It copies almost instantly. There is no delay when the application first runs.

5. REFERENCE

[1] Home page for android introduction retrieve on 8/12/12 from IBM website: http://www.ibm.com/developerworks/opensource/library/os-android-devel/

[2] Page for to learn how to create android application retrieved 4/12/12 from android developer site: http://developer.android.com/training/basics/firstapp/creating-project.html

[3] Page for to review how split file and read one by one. retrieved 8/12/2012 from androidsnips blogspot: http://androidsnips.blogspot.in/2010/10/copying-larger-database-files-in.html

[4] Page to review how to use raw folder to retrieve file retrieved 8/12/2012: http://stackoverflow.com/questions/2860157/load-files-bigger-than-1m-from-assets-folder

[5] http://stackoverflow.com/questions/10738623/copy-database-from-assets-folder-in-unrooted-device

[6] Page to review android directories retrived 6/ 12/12: http://developer.android.com/tools/projects/index.html