**Как добавить готовую базу SQLite в Android-приложение**

Сразу предупрежу, что материал не основан на моём опыте. Несколько раз встречал описание данной задачи и решил сохранить себе на всякий случай. Буду рад любым замечаниям, если использовали подобный способ.

Как известно, обычно мы в коде создаём новую пустую базу и пользователь начинает её заполнять. Если база небольшая, то также в коде можно программно добавить нужные записи. Но если требуется большая база данных на несколько мегабайт, то невольно задумаешься о решении проблемы.

Одним из способов является копирование заранее подготовленного файла базы данных в папку **assets** на этапе разработки, а далее программно можно скопировать данный файл в нужную системную папку базы данных вашего приложения. После этой операции можно работать с базой данных обычным способом.

Подготовить файл базы данных можно на рабочем компьютере при помощи различных программ для работы с SQLite, например, [SQLite Database Browser](http://sourceforge.net/projects/sqlitebrowser/).

Есть небольшая тонкость - кроме таблицы, которую вы создаёте для приложения, Android также создаёт для своих целей новую таблицу **android\_metadata** в вашей базе. Поэтому при ручном создании базы вам необходимо создать как минимум две таблицы: системную и свою рабочую.

Откройте вашу базу и добавьте новую таблицу под названием «android\_metadata»:

CREATE TABLE "android\_metadata" ("locale" TEXT DEFAULT 'en\_US')

Добавим в таблицу одну строку:

INSERT INTO "android\_metadata" VALUES ('en\_US')

Далее можете создать свои таблицы для работы.

Переименуйте первичное поле id на «\_id», как требует Android. В SQLite Database Browser это можно сделать, нажав на кнопку **Редактировать**, потом выбрав таблицу, которую вы хотите изменить, и, наконец, выбрав поле для переименования.

После выполнения указанных операций база данных готова для использования в вашем приложении.

Поместите ваш файл базы данных в папку **assets** вашего проекта и создайте новый класс, наследующий от SQLiteOpenHelper.

public class DataBaseHelper extends SQLiteOpenHelper{

// путь к базе данных вашего приложения

private static String DB\_PATH = "/data/data/YOUR\_PACKAGE/databases/";

private static String DB\_NAME = "myDBName";

private SQLiteDatabase myDataBase;

private final Context mContext;

/\*\*

\* Конструктор

\* Принимает и сохраняет ссылку на переданный контекст для доступа к ресурсам приложения

\* @param context

\*/

public DataBaseHelper(Context context) {

super(context, DB\_NAME, null, 1);

this.mContext = context;

}

/\*\*

\* Создает пустую базу данных и перезаписывает ее нашей собственной базой

\* \*/

public void createDataBase() throws IOException{

boolean dbExist = checkDataBase();

if(dbExist){

//ничего не делать - база уже есть

}else{

//вызывая этот метод создаем пустую базу, позже она будет перезаписана

this.getReadableDatabase();

try {

copyDataBase();

} catch (IOException e) {

throw new Error("Error copying database");

}

}

}

/\*\*

\* Проверяет, существует ли уже эта база, чтобы не копировать каждый раз при запуске приложения

\* @return true если существует, false если не существует

\*/

private boolean checkDataBase(){

SQLiteDatabase checkDB = null;

try{

String myPath = DB\_PATH + DB\_NAME;

checkDB = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN\_READONLY);

}catch(SQLiteException e){

//база еще не существует

}

if(checkDB != null){

checkDB.close();

}

return checkDB != null ? true : false;

}

/\*\*

\* Копирует базу из папки assets заместо созданной локальной БД

\* Выполняется путем копирования потока байтов.

\* \*/

private void copyDataBase() throws IOException{

//Открываем локальную БД как входящий поток

InputStream myInput = mContext.getAssets().open(DB\_NAME);

//Путь ко вновь созданной БД

String outFileName = DB\_PATH + DB\_NAME;

//Открываем пустую базу данных как исходящий поток

OutputStream myOutput = new FileOutputStream(outFileName);

//перемещаем байты из входящего файла в исходящий

byte[] buffer = new byte[1024];

int length;

while ((length = myInput.read(buffer))>0){

myOutput.write(buffer, 0, length);

}

//закрываем потоки

myOutput.flush();

myOutput.close();

myInput.close();

}

public void openDataBase() throws SQLException{

//открываем БД

String myPath = DB\_PATH + DB\_NAME;

myDataBase = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN\_READONLY);

}

@Override

public synchronized void close() {

if(myDataBase != null)

myDataBase.close();

super.close();

}

@Override

public void onCreate(SQLiteDatabase db) {

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

}

// Здесь можно добавить вспомогательные методы для доступа и получения данных из БД

// вы можете возвращать курсоры через "return myDataBase.query(....)", это облегчит их использование

// в создании адаптеров для ваших view

}

Теперь вы можете создать новый экземпляр класса DataBaseHelper и вызвать методы createDataBase() и openDataBase(). Не забудьте изменить YOUR\_PACKAGE на имя пакета в вашем приложении в строке DB\_PATH.

DataBaseHelper myDbHelper = new DataBaseHelper(this);

myDbHelper = new DataBaseHelper(this);

try {

myDbHelper.createDataBase();

} catch (IOException ioe) {

throw new Error("Unable to create database");

}

try {

myDbHelper.openDataBase();

}catch(SQLException sqle){

throw sqle;

}

...

**Библиотека для работы с готовой базой**

На Github есть проект в виде отдельной библиотеки, позволяющая упростить работу по переносу готовой базы данных с компьютера на устройство. Как я понял из описания, базу нужно заархивировать и положить в папку **assets/databases/**.

[Проект на Гитхабре](https://github.com/jgilfelt/android-sqlite-asset-helper).

**Другой способ**

Ещё один способ, найденный в сети.

Поместим файл базы данных в zip-архив и переместим его в папку **res/raw**.

В методе **onCreate()** проверим - запускается ли приложение первый раз. Распакуем БД и переместим её на SD-карту. Установим соединение с базой. Проверим, существует ли файл БД, если не существует, то нужно его развернуть из файла.

Данный метод поместим в **onCreate()**.

public void createDataBase() throws IOException{

//получаем путь к SD-карте.

File DB\_PATH = myContext.getExternalCacheDir();

//создаём каталог для нашей базы данных

DB\_PATH.mkdirs();

//проверяем есть ли уже файл БД на карте

File db = new File(DB\_PATH, DB\_NAME);

if(!db.exists()) {

//если файла нет, то попытаемся его создать

db.createNewFile();

try {

copyFromZipFile();

} catch (IOException e) {

throw new Error("Error copying database",e);

}

}

Этот метод копирует файл БД из **res/raw** на SD-карту

private void copyFromZipFile() throws IOException{

InputStream is = myContext.getResources().openRawResource(R.raw.database);

File outFile = new File(DB\_PATH ,DB\_NAME);

OutputStream myOutput = new FileOutputStream(outFile.getAbsolutePath());

ZipInputStream zis = new ZipInputStream(new BufferedInputStream(is));

try {

ZipEntry ze;

while ((ze = zis.getNextEntry()) != null) {

ByteArrayOutputStream baos = new ByteArrayOutputStream();

byte[] buffer = new byte[1024];

int count;

while ((count = zis.read(buffer)) != -1) {

baos.write(buffer, 0, count);

}

baos.writeTo(myOutput);

}

} finally {

zis.close();

myOutput.flush();

myOutput.close();

is.close();

}

}

Устанавливаем соединение с БД.

public SQLiteDatabase openDataBase() throws SQLException{

File DB\_PATH = myContext.getExternalCacheDir();

File dbFile = new File (DB\_PATH,DB\_NAME);

myDataBase = SQLiteDatabase.openDatabase(dbFile.getAbsolutePath(), null, SQLiteDatabase.OPEN\_READWRITE);

return myDataBase;

}